



الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific
Research
جامعة عين تموشنت بلحاج بوشعيب
Ain Temouchent University - Belhadj Bouchaib
Faculty of Science and Technology
Department of Mathematics and Computer
Science



End of Cycle Project
for obtaining Master's degree
Field: Mathematics and Computer Science
Sector: Computer Science
Specialty: Cyber Security And Artificial Intelligence

Theme

Identification Of An Email Author Using Machine Learning And Natural Language Processing

Presented on June 29, 2026, by:

- 1) ELMEGUENNI Hadjar
- 2) MOHAMMED SMAIN Bouchra

Before a jury composed of:

Mme. BARNOUSSI M.	MAA	UAT.B.B (Ain Temouchent) – President
M. BOUAFIA Z.	MAA	UAT.B.B (Ain Temouchent) – Examiner
M. BOUCHAKOUR E.	MCA	UAT.B.B (Ain Temouchent) – Supervisor

Academic Year 2025/2026

Appreciation

We would like to extend our deepest gratitude to our supervisor, Dr. BOUCHAKOUR for his insightful guidance and invaluable expertise. We also wish to express our sincere appreciation to all of our teachers. Each of them have played a pivotal role in shaping our academic experience. Their commitment to our education have been evident in every lecture, discussion, and piece of advice they have shared with us. The knowledge and skills we have gained under their guidance are immeasurable and will undoubtedly serve as a solid foundation for our future endeavors. This thesis is a testament to the collective effort of all those who have contributed to our education. We are incredibly thankful for the mentorship and support provided by Dr. BOUCHAKOUR and our teachers, which has been fundamental in achieving this milestone.

The greatest gratitude goes to God, our families, friends, and loved-ones for they are the reason we are here today.

Lastly, we would like to express our appreciation to ourselves for the dedication, perseverance, and effort invested throughout the completion of this work.

Abstract

Emails are often used in cybercrime today, so it is important to verify the identity of the email author. This paper proposes different Machine Learning models: Naive Bayes (NB), Logistic Regression (LR) and Support Vector Machine (SVM), to solve the problem of anonymous email author attribution. The main task is to find the author of an anonymous email among the many suspected targets, to verify if an email was in fact written by the sender.

In this project, the models were trained and tested using the same email dataset where we analyze writing style, vocabulary usage, and textual patterns, plus date and time information using supervised machine learning in addition to natural language processing techniques, with the intention of combining multiple factors to identify the author.

The tests proved that the accuracy varies from one model to another indicating that some are better than others. In email authorship verification experiments, usually the average accuracy reaches 89.9%. while our model's accuracy rate to a well distributed dataset is 89,06% for SVM , 81,77% for Naive Bays and 88.02% for Logistic Regression. And with a not so well distributed dataset the accuracy rate is 91,40% for SVM model, 76,74% for Naive Bays and for Logistic Regression 92.29%. Proving that a good identification system relies on two aspects, the model chosen for the task and the dataset veracity.

Key words: Identification, Machine Learning, NLP, text classification, TF-IDF, Naive Bays, SVM, LR, Email.

Résumé

Les courriels étant fréquemment utilisés dans la cybercriminalité, il est crucial de vérifier l'identité de leur auteur. Cet article propose différents modèles d'apprentissage automatique : Naive Bayes (NB), la régression logistique (LR) et les machines à vecteurs de support (SVM), afin de résoudre le problème de l'attribution de l'auteur d'un courriel anonyme. L'objectif principal est d'identifier l'auteur d'un courriel anonyme parmi les nombreuses cibles suspectes, afin de vérifier si le courriel a bien été écrit par l'expéditeur.

Dans le cadre de ce projet, les modèles ont été entraînés et testés sur un même ensemble de données de courriels, où nous avons analysé le style d'écriture, le vocabulaire et les structures textuelles, ainsi que les informations de date et d'heure, en utilisant l'apprentissage automatique supervisé et des techniques de traitement automatique du langage naturel, dans le but de combiner plusieurs facteurs pour identifier l'auteur.

Les tests ont démontré que la précision varie d'un modèle à l'autre, indiquant que certains sont plus performants que d'autres. Lors des expériences de vérification d'auteur de courriels, la précision moyenne atteint généralement 89,9%. Alors que le taux de précision de notre modèle sur un jeu de données bien distribué est de 89.06% pour SVM, 81,77% pour Naive Bays et 88.02 % pour la régression logistique, et avec un jeu de données moins bien distribué, le taux de précision est de 91,40 % pour le modèle SVM, 76,74% pour Naive Bays et pour la régression logistique 92.29%. Ceci prouve qu'un bon système d'identification repose sur deux aspects: le modèle choisi pour la tâche et la véracité du jeu de données.

Mots-clés: Identification, Apprentissage automatique, NLP, Classification de texte, TF-IDF, Naive Bays, SVM, Régression logistique, Email.

ملخص

تُستخدم رسائل البريد الإلكتروني بكثرة في الجرائم الإلكترونية اليوم، لذا من المهم التحقق من هوية كاتب البريد الإلكتروني. تقترح هذه الورقة نماذج مختلفة للتعلم الآلي: بايز الساذج، (NB) والانحدار اللوجستي، (LR) وآلة المتجهات الداعمة، (SVM) لحل مشكلة تحديد هوية كاتب البريد الإلكتروني المجهول. تتمثل المهمة الرئيسية في العثور على كاتب البريد الإلكتروني المجهول من بين العديد من الأهداف المشتبه بها، للتحقق مما إذا كان البريد الإلكتروني مكتوباً بالفعل من قبل المرسل. في هذا المشروع، تم تدريب النماذج واختبارها باستخدام نفس مجموعة بيانات البريد الإلكتروني، حيث قمنا بتحليل أسلوب الكتابة، واستخدام المفردات، والأنماط النصية، بالإضافة إلى معلومات التاريخ والوقت، باستخدام التعلم الآلي الخاضع للإشراف إلى جانب تقنيات معالجة اللغة الطبيعية، بهدف دمج عوامل متعددة لتحديد هوية الكاتب. أثبتت الاختبارات أن الدقة تختلف من نموذج لآخر، مما يشير إلى أن بعضها أفضل من غيرها. في تجارب التحقق من هوية كاتب البريد الإلكتروني، عادةً ما يصل متوسط الدقة إلى 89.9 في المئة. بينما بلغت دقة نموذجنا على مجموعة بيانات موزعة توزيعاً جيداً 89.06 في المئة لنموذج SVM، و81.77 في المئة لنموذج Bayes، Naive و88.02 في المئة لنموذج الانحدار اللوجستي.

ومع مجموعة بيانات غير موزعة توزيعاً جيداً، بلغت دقة النموذج 91.40 في المئة لنموذج SVM، و76.74 في المئة لنموذج Bayes، Naive و92.29 في المئة لنموذج الانحدار اللوجستي. وهذا يُثبت أن نظام التعرف الجيد يعتمد على جانبيين أساسيين: النموذج المختار للمهمة، ومصادقية مجموعة البيانات.

الكلمات المفتاحية: التعرف، التعلم الآلي، معالجة اللغة الطبيعية، تصنيف النصوص، الانحدار اللوجستي، البريد الإلكتروني، TF-IDF، Bayes، Naive، SVM.

Contents

General Introduction	10
1 Machine Learning	12
1.1 Introduction	12
1.2 What is Machine Learning	12
1.3 History of Machine Learning	13
1.4 Types of Machine Learning	14
1.4.1 Supervised Machine Learning	15
1.4.2 Unsupervised Machine Learning	17
1.4.3 Semi-supervised Machine Learning	18
1.4.4 Reinforcement Machine Learning	20
1.5 Components of Machine Learning	21
1.6 Relationship Between AI, ML, DL, and NLP	22
1.7 Overfitting vs Underfitting	23
1.7.1 Underfitting	23
1.7.2 Overfitting	24
1.7.3 Bias and Variance in Machine Learning	24
1.7.4 Reducing Underfitting	25
1.7.5 Reducing Overfitting	25
1.8 Evaluation Metrics in Machine Learning	25
1.8.1 Classification	25
1.8.2 Regression	27
1.8.3 Clustering	29
1.9 Real-World Applications of Machine Learning	30
1.10 Advantages of Machine Learning	30
1.11 Disadvantages of Machine Learning	31
1.12 Conclusion	31
2 TF-IDF	32
2.1 Introduction	32
2.2 What Is TF-IDF	32

2.3	Components Of TF-IDF	32
2.3.1	Term Frequency (TF)	32
2.3.2	Inverse Document Frequency (IDF)	33
2.3.3	TF-IDF Score	33
2.3.4	Example	33
2.4	Applications	35
2.5	Advantages	35
2.6	Limitations	35
2.7	Conclusion	35
3	State of the Art	37
3.1	Introduction	37
3.2	Articles Analysis	38
3.2.1	The Email Author Identification System (SVM and AHP)	38
3.2.2	Identifying the Authors of Suspect Email	38
3.2.3	Author Profiling for English Emails	39
3.2.4	Classification Approach to Email Authorship Identification	40
3.3	Comparison Tables	41
3.4	Conclusion	42
4	Contribution	43
4.1	Introduction	43
4.2	Dataset	43
4.3	Libraries Used	45
4.3.1	Flask (Web Framework)	45
4.3.2	Pandas (Data Handling)	45
4.3.3	NumPy (Numerical Computation)	46
4.3.4	Scikit-learn (Machine Learning)	46
4.3.5	spaCy (Word Embeddings)	46
4.3.6	SciPy (Sparse Matrix Handling)	46
4.3.7	Pickle (Model Persistence)	47
4.3.8	Regular Expressions (re module)	47
4.3.9	OS (Operating System Interface)	47
4.4	Summary of the System	47
4.4.1	Training Phase	48
4.4.2	Naive Bayes (NB) Model	51
4.4.3	Support Vector Machine (SVM)	51
4.4.4	Logistic Regression (LR)	52
4.4.5	Prediction Phase	52

4.5	Models Behavior Summary	55
4.6	Overview	55
4.6.1	Training Section	56
4.6.2	Prediction Section	59
4.7	Conclusion	60
	General Conclusion	61
	References	62

List of Figures

1.1	Types of Machine Learning [36]	15
1.2	Machine Learning Pipeline (Generated by AI)	22
1.3	Underfitting, Appropriate Fitting, and Overfitting [37]	24
4.1	Libraries	45
4.2	Train and Test(Generated by AI)	48
4.3	Models Accuracy (Generated by AI)	55
4.4	Main Interface	56
4.5	Training Section	56
4.6	Training Load	57
4.7	Evaluation Matrices	58
4.8	Evaluation Metrics For Each Author	58
4.9	Prediction Example	59
4.10	Prediction Result	60

List of Tables

3.1	Comparison between articles (basic information)	41
3.2	Accuracy, advantages and limitations comparison	42
4.1	Summary of the Datasets Used	44
4.2	Comparison of Machine Learning Models	55

General Introduction

With the rapid growth of digital communication, email has become one of the most widely used communication tools in both personal and professional environments. However, this expansion has also increased cybercrime activities such as phishing, email fraud, identity theft, impersonation, and malicious anonymous communications. In this context, identifying the real author of an email has become an important challenge in the fields of cybersecurity and digital forensics.

Email author identification belongs to the domains of Machine Learning, Natural Language Processing (NLP), and Cybersecurity. Specifically, it analyzes writing patterns, vocabulary usage, linguistic style, and textual features in order to automatically determine the author of a given email. Recent advances in machine learning have made it possible to build intelligent systems capable of learning these patterns and improving authorship prediction accuracy.

Traditional methods of cybercrime investigation often require significant manual analysis and expert intervention, which can be time-consuming and inefficient when dealing with large volumes of emails. In many situations, malicious users attempt to hide their identity by using fake accounts or anonymous messages, making the identification process even more difficult.

The main problem addressed in this project is how to accurately identify the author of an email using machine learning techniques. More specifically, this work aims to compare different machine learning models in order to determine which algorithm provides the best performance in terms of accuracy, precision, recall, and F1-score for email authorship identification.

Unlike previous works, this project develops an intelligent email author identification system based on machine learning using three different classification models:

- Naive Bayes (NB)
- Logistic Regression (LR)
- Support Vector Machine (SVM)

The system includes several important components, such as advanced text cleaning, TF-IDF vectorization, style feature extraction, and seman-

tic embeddings using spaCy. In addition, an AHP-inspired weighting approach was used to combine multiple textual features in order to improve classification performance.

A complete web application was also developed using Flask, providing functionalities such as upload dataset, prediction visualization, multilingual interface support, dark mode, and performance evaluation metrics including accuracy, precision, recall, F1-score, and classification reports.

This thesis is organized into several chapters:

First chapter is called Machine Learning which provides a general understanding of machine learning as a major field of artificial intelligence. It explains how machines learn from data, presents the main learning approaches, and discusses the fundamental concepts required to build and evaluate intelligent systems used in real-world applications.

Second chapter is about TF-IDF, it focuses on TF-IDF as an important text processing and feature extraction technique, explaining how textual data can be transformed into numerical representations that allow machine learning models to analyze and classify email content.

Third chapter is the State Of The Art which presents an overview of previous research studies related to the identification of authors of emails and compares the machine learning techniques and feature extraction methods used in existing approaches.

Lastly the Contribution chapter that discusses the implementation of the application, experimental results, comparison of models, and performance evaluation.

Finally, the dissertation concludes with a general summary of the obtained results and discusses possible future improvements and perspectives.

Chapter 1

Machine Learning

1.1 Introduction

Ever since computers were invented in 1945, scientists have wondered whether they might be able to learn. Fast forward in recent years, successful machine learning applications have been developed, from data-mining programs that detect fraud to medical diagnosis, to autonomous vehicles that learn to drive on public highways. Although, we do not yet know how to make computers learn nearly as well as people learn. However, algorithms have been invented that are effective for certain types of learning tasks.

For problems such as speech recognition, algorithms based on machine learning outperform all other approaches that have been attempted to date.[19] In the field known as data mining, machine learning algorithms are being used routinely to discover valuable knowledge from large databases to serve us in various fields. So let us now explore the fundamentals of machine learning.

1.2 What is Machine Learning

Machine learning (ML) is a branch or subfield of artificial intelligence (AI), that uses algorithms already trained on data sets to develop self-learning models able to predict results, improve from experience, identify patterns and categorize information independently without being explicitly programmed (zero human involvement).[1]

1.3 History of Machine Learning

The term machine learning was first used in 1959 by Arthur Samuel, an IBM worker and a trailblazer in computer gaming and artificial intelligence. The term self-teaching computers was likewise utilized in that era.

The first machine learning program was created in the 1950s when Arthur Samuel developed a computer application that assessed the probability of winning in checkers for both sides, yet the origins of machine learning trace back to years of human ambition and effort to understand cognitive functions. In 1949, Canadian psychologist Donald Hebb released the book *The Organization of Behavior*, where he presented a theoretical neural structure resulting from specific interactions among nerve cells[2, 5].

By the early 1960s, Raytheon Company had developed an experimental “learning machine” called Cybertron, which utilized punched tape memory to analyze sonar signals, electrocardiograms, and speech patterns through basic reinforcement learning. It was continually “trained” by a human instructor to identify patterns and had a “goof” button to reevaluate incorrect decisions.

A notable publication on machine learning research from the 1960s was Nils Nilsson’s work on *Learning Machines*, [11, 2] which primarily focused on machine learning for pattern classification.

Interest in pattern recognition persisted into the 1970s, as detailed by Duda and Hart in 1973[12].

In 1981, a study was presented on employing teaching methods so that an artificial neural network could learn to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.

Tom M. Mitchell offered a widely referenced, more formal definition of the algorithms used in the machine learning domain. He quotes:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .”

This definition of the tasks relevant to machine learning provides a fundamentally operational perspective rather than characterizing the field in cognitive terms. This stems from Alan Turing’s suggestion in his work *Computing Machinery and Intelligence*, where the inquiry “Can machines think?” is exchanged for the question, “Can machines perform tasks that we (as thinking entities) can accomplish?”[2, 8]

Machine learning gained prominence in the 1980s and 1990s with scientists constantly developing new algorithms such as artificial neural networks and support vector machines.

Data accessibility rose notably in the 2000s, alongside improvements in processing power and storage capabilities. Consequently, advanced machine learning techniques, such as deep learning and reinforcement learning, were developed.

Contemporary Machine Learning algorithms are categorized into three types: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

Current Supervised Learning Algorithms aim at classification and regression tasks[?].

Current Unsupervised Learning Algorithms focus on clustering, dimensionality reduction, and association rule mining.

Current Reinforcement Learning Algorithms emphasize decisions needing to be made concerning a prior, unknown moment and are classified as either model-based approaches or model-free approaches.

In 2014, Ian Goodfellow and his team presented generative adversarial networks (GANs) for realistic data generation, and in 2016 AlphaGo was able to defeat top human players by using reinforcement learning techniques[2, 8].

1.4 Types of Machine Learning

The field of machine learning includes different kinds of models, each designed for particular data contexts and business goals. For companies aiming to adopt scalable and trustworthy AI solutions, it is essential to grasp these different forms of machine learning.

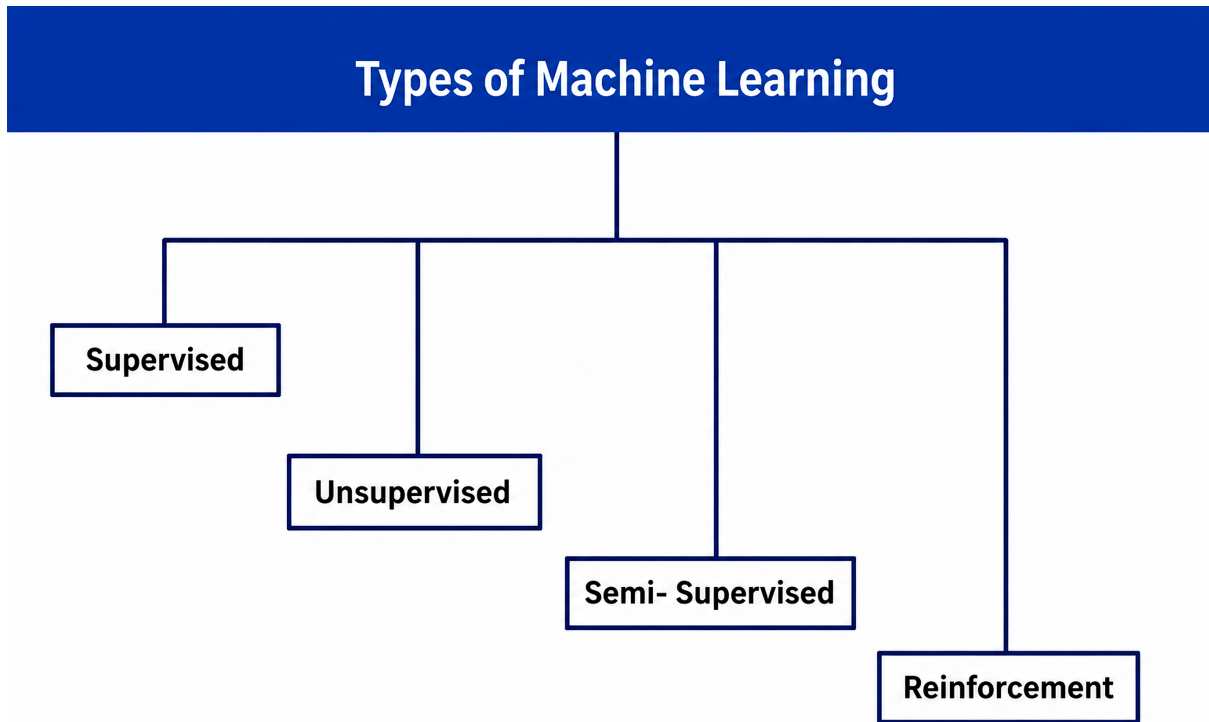


Figure 1.1: Types of Machine Learning [36]

These varied types of machine learning provide companies with an extensive array of tools to enhance workflows, boost predictive precision, and utilize data-informed intelligence.[3] By mastering these types of machine learning, data scientists and business leaders can create solutions that align with their strategic goals and the availability of data.

1.4.1 Supervised Machine Learning

Supervised learning is a machine learning approach where models are trained using labeled data, meaning each input is paired with a specific label. This is an attempt to understand a mapping that reduces the generalization error of the model on new data. Models adjust their parameters by utilizing loss functions such as mean squared error (for regression) or cross-entropy (for classification) to modify the parameters. Supervised learning is widely used in areas such as credit scoring and medical diagnosis[3, ?].

Categories of Supervised Machine Learning

- **Classification:** Allocates categorical tags to input data, such as spam identification or picture recognition.

- **Regression:** Anticipates continuous numerical results, like forecasting sales or predicting temperatures.
- **Decision Trees:** Structured models that divide data according to feature limits to reach conclusions.
- **Random Forests:** An ensemble technique that combines several decision trees to enhance generalization and lower variance[3].

Advantages of Supervised Machine Learning

- Generates extremely precise and dependable models (given sufficiently high-quality data).
- Robust performance in controlled environments (using labeled datasets and established results).
- Able to manage a diverse range of activities with classification and regression methods.
- Continuous Improvement Through Feedback.

Disadvantages of Supervised Machine Learning

- Depends significantly on labeled training data, making it potentially time-consuming or difficult to gather.
- Sensitivity to irrelevant features and noisy or inaccurate labels.
- Limited flexibility for unstructured or evolving data[4, 6].

Applications of Supervised Machine Learning

- Image recognition.
- Customer sentiment analysis.
- Spam detection.
- Predictive analysis.
- Data mining.
- Health monitoring[4].

1.4.2 Unsupervised Machine Learning

Unsupervised learning refers to a category of machine learning that utilizes unlabeled data to deduce the data's structure through comparable, yet dimensionally reduced, data points without depending on predefined labels. It utilizes algorithms to detect clusters or relationships among observations and to deduce hidden variables. Key techniques include k-means clustering, hierarchical clustering, and principal component analysis (PCA). This approach is necessary when labeled data is scarce for uncovering hidden patterns, segmenting customers, and detecting anomalies[3]. It often creates beneficial attributes for subsequent supervised learning.

Categories of Unsupervised Machine Learning

- **Clustering:** Organizes unlabeled data points into clusters according to their similarities or differences.
- **Density Estimation:** Calculates the probability distribution responsible for the data to recognize typical patterns within it.
- **Anomaly Detection:** Recognizes atypical or infrequent data points that deviate from the standard pattern of the data.
- **Dimensionality Reduction:** Decreases the feature count in the dataset while maintaining key information to streamline analysis.
- **Autoencoders:** Neural networks that learn efficient data representations for reconstruction purposes.

Advantages of Unsupervised Machine Learning

- Excels in examining raw, unstructured data and revealing concealed trends.
- Can manage large amounts of data and create representations swiftly.
- Capable of streamlining intricate, multi-dimensional information.
- Fast classification[4, 7].

Disadvantages of Unsupervised Machine Learning

- It may occasionally result in unforeseen or less-than-ideal outcomes because of untagged data.

- Measuring the effectiveness of unsupervised learning models can be difficult.
- Complexity in algorithm selection and parameter tuning.
- Exclusively carries out classification tasks[4, 7].

Applications of Unsupervised Machine Learning

- Exploratory data analysis.
- Image compression.
- Social network analysis.
- Detecting anomalies.
- Dimensionality reduction.
- Identifying customer personas.
- Determining market segmentation[4].

1.4.3 Semi-supervised Machine Learning

Semi-supervised learning leverages aspects of both supervised and unsupervised learning, using a small set of labeled data alongside a substantial amount of unlabeled data. This approach enhances both the efficiency of the model training process and the accuracy of the training, and it is especially valuable in industries where labeling is costly. Methods such as self-training, co-training, and graph-based approaches enable models to share label information and recognize patterns.[3] It is especially beneficial in natural language processing and computer vision, where there is a lack of labeled data but an abundance of unlabeled data. This approach improves generalization and reduces annotation expenses.

Categories of Semi-Supervised Machine Learning

- **Self-Training:** The model consecutively assigns labels to unlabeled data based on its own predictions.
- **Co-Training:** Two classifiers, each trained on distinct feature sets, educate one another[3].

- **Graph-Based Approaches:** Employ graph frameworks to transfer labels among similar data instances.
- **Multi-View Learning:** Integrates various data representations to enhance learning.

Advantages of Semi-Supervised Machine Learning

- Enhanced precision thanks to unlabeled data that allows the model to grasp fundamental data patterns, thus boosting prediction precision.
- Cost efficiency since annotating data can require significant time and expense.
- Flexible in a range of functions, such as spam detection, sentiment evaluation, and image categorization[9].

Disadvantages of Semi-Supervised Machine Learning

- The importance of selecting the right architecture and parameters.
- Unlabeled data might include noise or be unrelated to the task resulting in disrupted information that can cause overfitting.
- Error in consistency assumption resulting in mistakes.
- Computational intricacy, particularly with complex models or vast amounts of data.
- Ambiguity in performance evaluation as it depends on the quality of the unlabeled data[14].

Applications of Semi-Supervised Machine Learning

- Recommender systems.
- Document classification.
- Speech recognition.
- Face recognition.
- Handwritten text recognition.
- Image classification.

- Web content classification[3, 1].

1.4.4 Reinforcement Machine Learning

In Reinforcement Learning (RL), agents engage with their environment based on reward signals to discover the optimal strategies. Markov decision processes (MDPs) are utilized in reinforcement learning (RL) to represent decision-making amid uncertainty. Agents can employ different strategies,[3] such as actor-critic approaches, policy gradient techniques, and Q-learning, to enhance their total rewards. Reinforcement learning is particularly beneficial for adaptive control and real-time optimization in areas such as robotics and resource management.

Categories of Reinforcement Machine Learning

- **Value-Based Approaches:** Assesses the worth of actions to formulate optimal policies (e.g., Q-learning).
- **Policy-Based Approaches:** Optimizes the policy function directly, bypassing value estimation.
- **Model-Based Approaches:** Utilizes an environmental model for forecasting and strategy.
- **Actor-Critic Techniques:** Merges value and policy methods for enhanced learning consistency[3].

Advantages of Reinforcement Machine Learning

- Efficiency in handling long-term planning and sequential decision-making.
- Scalability to complex decision-making problems.
- Flexibility in adapting to new information.
- Ability to identify and refine strategies through trial and error.
- Useful in multi-agent environments[10].

Disadvantages of Reinforcement Machine Learning

- Heavy reliance on extensive interaction with a large amount of environment to learn effectively.
- Demands significant computational power and extensive resources.
- Hard to maintain balance between exploration and exploitation.
- Risk of learning unintended or dangerous behaviors[10].

Applications of Reinforcement Machine Learning

- Healthcare.
- Education.
- Natural language processing (NLP).
- Robotics.
- Chat-bots and dialogue systems[10, 2].

1.5 Components of Machine Learning

- **Dataset:** Collecting data is the first step and arguably the most important one in the process, an accurate and high quality data gives the system the facts and illustrations it needs so that the algorithm can learn the right patterns and generate insightful predictions.
- **Features:** Raw data cannot be used directly by the algorithms thus a precise and superior feature extraction is a necessary process of converting raw data into meaningful numerical features that a machine learning model can understand and learn from.
- **Algorithm:** Data-driven system learning is determined by algorithms, selecting the appropriate algorithm is crucial for effective pattern collection, helping the system to make accurate decisions and predictions.
- **Model:** The algorithm trains on the data to develop the model. It keeps track of patterns and rules, which enables the system to accurately and efficiently predict outcomes for new information.

- **Training:** Training is where the model is continuously adjusted using data by comparing its predictions with correct results to enhance accuracy in order to make the model ready for practical use.
- **Test:** After the training the model is put to action, it is now ready to be used on real-time data so it can finally be useful to people for daily work, providing information such as making forecasts, recommendations, or decisions depending on the task.
- **Evaluation:** Evaluation measures the model's performance. The feedback from training and testing evaluate the effectiveness of the model, this is done to enhance precision, correcting errors, and maintaining the system's dependability over times[1, 13].

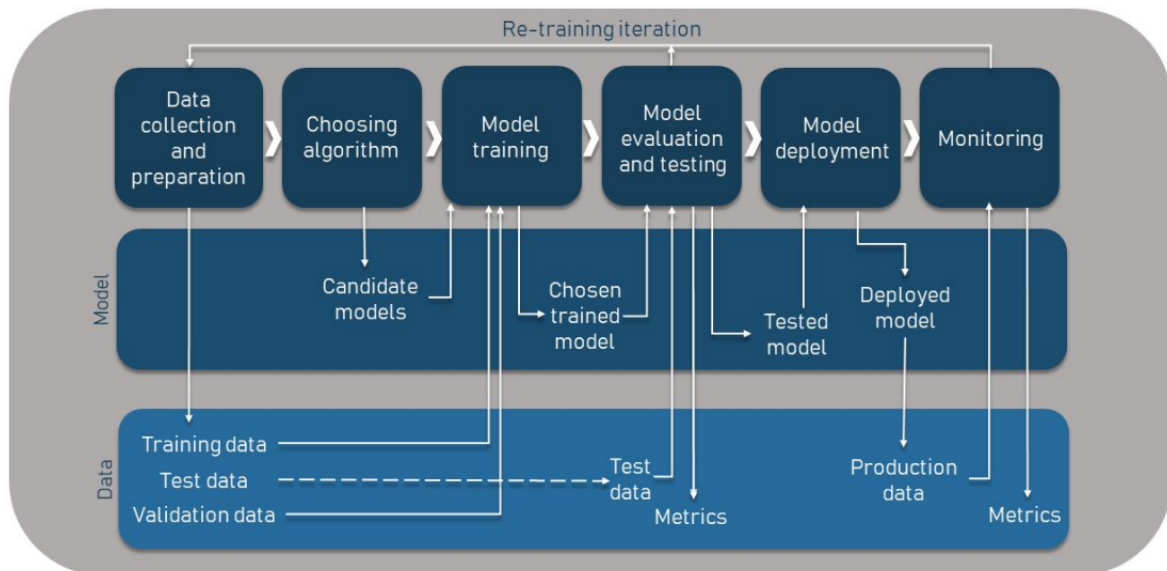


Figure 1.2: Machine Learning Pipeline (Generated by AI)

1.6 Relationship Between AI, ML, DL, and NLP

- **Artificial Intelligence (AI):** The term AI has been around since the 1950s, and it is the broadest field that refers to the development of computer systems simulating human intelligence in order to perform tasks that normally require a human brain . It covers several areas including science, psychology, philosophy, linguistics and others

- **Machine Learning (ML):** is a major subfield that allows systems to learn from data and improve their performance over time without being explicitly programmed for every task.
- **Deep Learning (DL):** is a branch of Machine Learning that uses artificial neural networks with many layers to automatically learn complex features from large amounts of data. Unlike traditional ML, in DL algorithms naturally extract features, this works wonderfully in images, audio, and text making the model faster and more adaptable.
- **Natural Language Processing (NLP):** in AI there is a field specializes in making machines like computers understand, interpret, and generate human language. It's the same concept that an adult uses with a child to help them understand the meaning of words. NLP systems are paired with ML and DL techniques to perform tasks such as translation, sentiment analysis, speech recognition, and chat-bots [3].

In summary, AI is the vast field that aims to make machines intelligent. Machine Learning (ML) is a segment of AI that allows systems to learn from data. Deep Learning (DL) is a more advanced part of ML that uses neural networks to learn complex patterns automatically making the system operate smarter not harder. And in between all of this exists NLP, an area of AI that focuses on enabling the machine to understand languages.

1.7 Overfitting vs Underfitting

1.7.1 Underfitting

Occurs when the model fails to learn important patterns, resulting in high errors across training and testing datasets. Causes include:

- Model too simple.
- Very high regularization.
- Weak or missing feature.
- Not enough training.
- High bias[4].

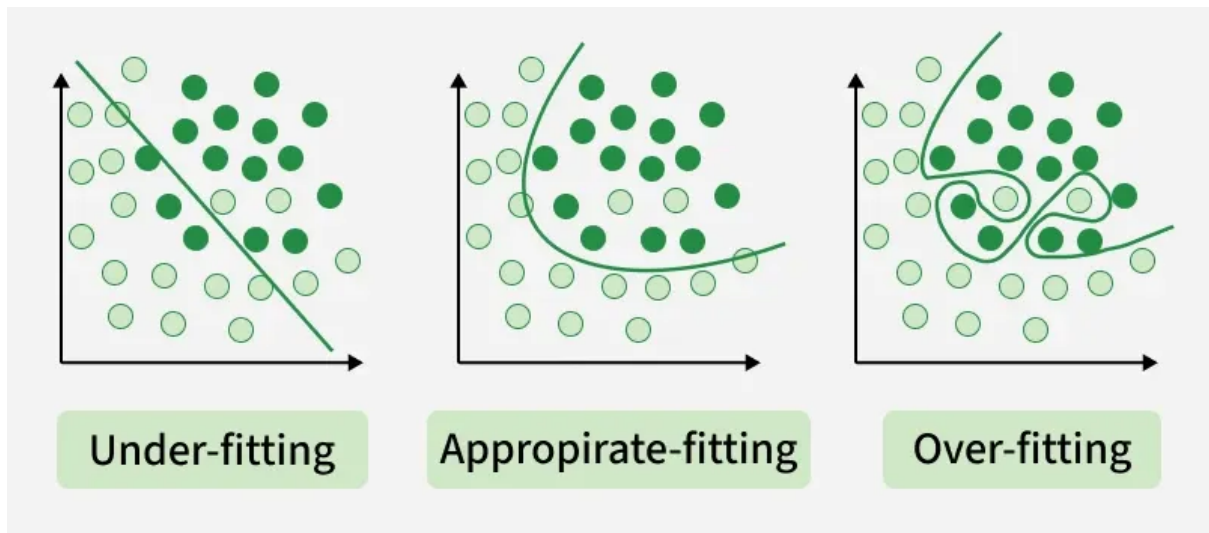


Figure 1.3: Underfitting, Appropriate Fitting, and Overfitting [37]

1.7.2 Overfitting

Occurs when the model learns too much from training data including noise, causing low training error but high testing error. Causes include:

- Model too complex.
- Too many features.
- Very little data.
- No regularization.
- High variance[4].

1.7.3 Bias and Variance in Machine Learning

- **Bias:** Strong assumptions that ignore complexities of data, leading to systematic errors.
- **Variance:** Model captures noise and random details, leading to poor generalization.

Overfitting = Low Bias + High Variance
Underfitting = High Bias + Low Variance

1.7.4 Reducing Underfitting

- Employ a more complex model.
- Lower regularization.
- Incorporate additional features.
- Clean data to avoid outliers[4].

1.7.5 Reducing Overfitting

- Minimize model complexity.
- Use regularization.
- Collect more training data.
- Remove noise from data[4].

1.8 Evaluation Metrics in Machine Learning

When developers build machine learning models, they try to understand and evaluate their performance to help them measure the effectiveness of the models. Thus comes the implementation of evaluation metrics:

1.8.1 Classification

In classification, the object is for the model to classify input data into specific predefined categories or classes.

Accuracy

Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Total Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

While accuracy [7] does provide evaluation of performance, it can be misleading in cases of imbalanced datasets. For example: If 90% of data represent one class, predicting that class alone will result in 90% accuracy but completely dismisses the minority class, leading to a false sense of achievement.

Precision

Precision measures how many of the positive predictions made by the model are actually correct[2]. It is useful in cases like medical diagnoses where predicting a disease when it's not present can have serious consequences:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- TP : True Positives (predicted Yes and actual Yes)
- FP : False Positives (predicted Yes but actual No)

Recall

Recall[?] calculates how many of the positive cases were correctly identified by the model:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- FN : False Negatives (predicted No but actual Yes)

Recall is important in cases when missing a positive case is more costly than false positives. Example: disease detection.

F1 Score

F1 Score combines precision and recall into a single metric[2]:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Log Loss

Logarithmic loss calculates the uncertainty of the model's predictions[1]. For N samples belonging to M classes:

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

- y_{ij} : Actual class (0 or 1) for sample i and class j
- p_{ij} : Predicted probability for sample i and class j

Area Under Curve (AUC)

AUC measures the model's ability to separate positive and negative classes[15]:

$$\text{AUC} = \frac{d}{P \times N}$$

- d = number of times the positive score $>$ negative score
- P : number of positive samples
- N : number of negative samples

Additional metrics:

- True Positive Rate(TPR) : $\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$
- True Negative Rate(TNR) : $\text{TNR} = \text{TN} / (\text{TN} + \text{FP})$
- False Positive Rate(FPR) : $\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$
- TN : True Negative (It is the case where we predicted No and the real output was also No)
- False Negative Rate(FNR) : $\text{FNR} = \text{FN} / (\text{FN} + \text{TP})$

Confusion Matrix

Confusion matrix[2] creates an $N \times N$ matrix, where N is the number of classes. Example: binary classification ($N = 2$):

- $n = 160$

	Predicted NO	Predicted YES
Actual NO	55	10
Actual YES	15	80

1.8.2 Regression

In regression, the target variable is continuous values. To evaluate the performance of such a model below metrics are used:

Mean Absolute Error (MAE)

MAE calculates the average absolute difference between actual and predicted values[16]. It does not heavily penalize large errors.

$$\text{MAE} = \frac{1}{N} \sum_{j=1}^N |y_j - \hat{y}_j|$$

- y_j : Actual value
- $\hat{y}_j$: Predicted value

Mean Squared Error (MSE)

MSE calculates the average of the squared difference between the predicted and actual values[2]. It penalizes large errors more strongly.

$$\text{MSE} = \frac{1}{N} \sum_{j=1}^N (y_j - \hat{y}_j)^2$$

Root Mean Squared Error (RMSE)

RMSE is the square root of MSE, and easier to interpret than MSE.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{j=1}^N (y_j - \hat{y}_j)^2}$$

Root Mean Squared Logarithmic Error (RMSLE)

RMSLE is useful when target values vary widely[16]. Unlike RMSE, it penalizes underestimations more than over-estimations, and is more common in price or population prediction.

$$\text{RMSLE} = \sqrt{\frac{1}{N} \sum_{j=1}^N (\log(y_j + 1) - \log(\hat{y}_j + 1))^2}$$

R² (R-squared)

The R² score represents how much variance is explained by the model[17]. The range is $(-\infty, 1]$; an R^2 value close to 1 shows a model that explains most of the variance while a value close to 0 shows that the model does not explain much of the variability in the data, and in extreme cases it can reach very large negative numbers.

$$R^2 = 1 - \frac{\sum_{j=1}^N (y_j - \hat{y}_j)^2}{\sum_{j=1}^N (y_j - \bar{y})^2}$$

- y_j : Actual value
- $\hat{y}_j$: Predicted value
- $\bar{y}$: Mean of actual values

1.8.3 Clustering

In unsupervised learning tasks like clustering, the aim is to group similar data points together.

Silhouette Score

Silhouette Score measure cohesion (how well a data point fits within its assigned cluster) and separation (how far it is from points in other clusters)[18]. The value range is $[-1, 1]$, where 1 signifies that data points are well-matched to their own cluster and poorly matched to neighboring clusters.

$$S = \frac{b - a}{\max(a, b)}$$

- a : Average distance between a sample and all other points in the same cluster
- b : Average distance between a sample and all points in the nearest cluster

Davies-Bouldin Index

Davies-Bouldin Index measures the average similarity between clusters[17]. A lower value indicates better clustering (means that the clusters are well-separated).

$$DB = \frac{1}{N} \sum_{i=1}^N \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

- σ_i : Average distance of points in cluster i from the cluster centroid
- $d(c_i, c_j)$: Distance between centroids of clusters i and j

In summary, the variety of evaluation metrics indicates their importance in machine learning by helping models achieve optimal performance .

1.9 Real-World Applications of Machine Learning

Machine learning is a modern invention that has improved our daily lives along with many professional and industrial processes, applied not only in tech-heavy businesses but across various fields, such as:

- Speech Recognition.
- Medical Diagnosis.
- Fraud Detection.
- Product Recommendations
- Facial Recognition.
- Healthcare Advancement.
- Email Automation and Spam Filtering.
- Image Recognition[5].

1.10 Advantages of Machine Learning

- Handling large datasets efficiently.
- Automation of repetitive tasks.
- Improved decision-making.

- Reduces operational expenses.
- Identifies trends to enhance predictions.
- Continuous improvement.
- Replaces repetitive tasks.
- Constant technological advancements[1].

1.11 Disadvantages of Machine Learning

- High computational costs.
- Requires high-quality and quantity data.
- Risk of bias and variance issues.
- Challenges with scalability[1, 6].

1.12 Conclusion

The rise of machine learning is an inherently interdisciplinary field that represents a fundamental shift in how we interact with the world. As we conclude this chapter, it is clear that machine learning is more than just a technological way to solve problems, but rather a radical force that is reshaping industries and redefining the boundaries of what computers can achieve.

Chapter 2

TF-IDF

2.1 Introduction

In this project we use TF-IDF (Term Frequency-Inverse Document Frequency) to transform text data into numerical form, capturing the importance of words within each email. It is one of the first and most important steps in any task that requires natural language processing pipeline, such as spam detection, text classification or email author identification.

2.2 What Is TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus). The TF-IDF vectorizer is a tool that transforms text data into a numerical format that can be used for machine learning and natural language processing tasks[26].

2.3 Components Of TF-IDF

2.3.1 Term Frequency (TF)

This measures how frequently a term appears in a document.[26] It is calculated as:

$$TF(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

This gives a higher weight to terms that appear more frequently in a specific document, highlighting that it is likely relevant to the document's content.

2.3.2 Inverse Document Frequency (IDF)

This measures how important a term is across the entire corpus. It is calculated as:

$$IDF(t, D) = \log \left(\frac{\text{Total number of documents } |D|}{\text{Number of documents containing term } t} \right)$$

The idea is that terms that are common across many documents (like “the”, “is”, etc.) will have lower IDF scores, while rare terms will have higher scores.

2.3.3 TF-IDF Score

The final score for a term in a document is calculated by multiplying its TF and IDF values[26].

2.3.4 Example

We have the following documents:

- Document 1: “Data science is fun and powerful.”
- Document 2: “Machine learning is a part of data science.”
- Document 3: “Artificial intelligence and machine learning are evolving.”

Our goal is to calculate the TF-IDF score for the word “learning”.

Step 1: Calculate Term Frequency (TF)

Document 1

- “learning” appears 0 times
- Total words = 6

$$TF(\text{learning}, D1) = 0$$

Document 2

- “learning” appears 1 time
- Total words = 8

$$TF(learning, D2) = \frac{1}{8} = 0.125$$

Document 3

- “learning” appears 1 time
- Total words = 7

$$TF(learning, D3) = \frac{1}{7} \approx 0.143$$

Step 2: Calculate Inverse Document Frequency (IDF)

- Total number of documents ($N = 3$)
- Documents containing “learning” = 2 (D2 and D3)

$$IDF(learning) = \log\left(\frac{3}{2}\right) \approx 0.176$$

Step 3: Calculate TF-IDF

Document 1

$$TF-IDF = 0 \times 0.176 = 0$$

Document 2

$$TF-IDF = 0.125 \times 0.176 \approx 0.022$$

Document 3

$$TF-IDF = 0.143 \times 0.176 \approx 0.025$$

Final Result:

The word “learning” is:

- most important in Document 3.
- slightly less important in Document 2.
- has no importance in Document 1[26].

2.4 Applications

- Feature Extraction: TF-IDF is commonly used in text classification, information retrieval, and natural language processing to convert text documents into numerical vectors that can be fed into machine learning algorithms.
- Text Similarity: It helps in measuring the similarity between documents based on their content[18].
- Search Engines: TF-IDF is often used by search engines to rank documents based on the relevance of the search query.
- Filtering words: it helps removing the unnecessary words out of a text body[24].

2.5 Advantages

- Simple and easy to understand.
- Easy to compute.
- Fixed-Size Input (Good for ML Algorithms).
- It captures the importance of words[25].

2.6 Limitations

- limited linguistic awareness.
- It handles rare or unseen words poorly.
- The performance depends on the length of the document.
- It doesn't capture position in text[24].

2.7 Conclusion

TF-IDF is a fast, transparent, strong and effective tool for many text problems, particularly retrieval and keyword-driven tasks. It performs a

crucial role in improving model performance. Despite other more innovative methods in deep learning, TF-IDF still serves as the baseline for many text-based machine learning applications.

Chapter 3

State of the Art

3.1 Introduction

In recent years, machine learning has expanded to include fundamental tools in text analysis, serving various fields like automatic recognition, classification, and interpretation of large volumes of unstructured textual data. Techniques such as text classification, natural language processing, and stylometric analysis allow systems to extract meaningful patterns from written content, making it possible to identify topics, sentiments, or in our case; email authors. Within this context, email author identification has emerged as an important application, particularly in domains such as cybersecurity, digital forensics, and spam detection.

Recent studies have explored a wide range of machine learning algorithms for text classification tasks, including Support Vector Machines, Logistic Regression, and Naive Bays, as well as ensemble and deep learning methods. These approaches often rely on vectorization techniques such as TF-IDF and n-grams to transform textual data into numerical representations suitable for model training. These few carefully chosen articles that demonstrate projects about email author identification have shown results that prove modern ML algorithms can achieve high levels of accuracy, especially when trained on large and diverse datasets.

3.2 Articles Analysis

3.2.1 The Email Author Identification System (SVM and AHP)

In this article Qinghe Zheng et al [23] proposes a machine learning system designed to identify the author of an email by combining statistical text analysis with a structured way of weighting features. The main idea of the study is that not all features in a text carry the same importance, so the model improves classification by first assigning weights to features before feeding them into a classifier. To do this, the authors use TF-IDF to convert email text into numerical form, then apply the Analytic Hierarchy Process (AHP) to determine the importance of different features, and finally use a Support Vector Machine (SVM) for classification. The system considers a wide range of features that reflect both writing style and structural habits, including word frequency, sentence length, syntax structure, punctuation usage, and email formatting. By combining these elements, the model captures not just what is written but also how it is written. The results reported in the paper are strong, achieving more than 95% accuracy, and showing that adding a feature-weighting step like AHP can significantly improve performance compared to using raw features alone. The importance of this work lies in demonstrating that combining linguistic features with intelligent weighting strategies can enhance authorship identification systems. However, the study also has some limitations. It is mainly based on a Chinese dataset, which may limit its generalization to other languages, and it relies heavily on manually designed features. Additionally, the model only uses SVM for classification without comparing it to other machine learning approaches, which makes it harder to fully evaluate its relative performance.

3.2.2 Identifying the Authors of Suspect Email

Malcolm Corney et al [22] proposes this paper in order to look at whether a person can be identified just from the way they write emails, using a technique called stylometry. The idea behind it is that every writer has a kind of “linguistic fingerprint,” meaning they tend to use language in consistent and recognizable ways without even realizing it. To test this, the researchers extract different writing-style features from emails and then use a Support Vector Machine (SVM) classifier to distinguish between authors. The focus here is more on linguistic behavior than on complex

modern machine learning models. To achieve this, the study extracts stylometric features from email text and applies a Support Vector Machine (SVM) classifier, with a stronger emphasis on linguistic analysis than on highly complex machine learning architectures. The features used include function words such as “and” and “the,” sentence structure, vocabulary richness, as well as character-level and word-level patterns, all of which are considered stable indicators of writing style. The key findings show that even very small email samples, sometimes as little as 100 words, can still be sufficient to identify authors, supporting the idea that writing style remains consistent across different texts. The paper does not report a single explicit accuracy value; instead, it demonstrates successful identification performance across datasets, including cases involving around 20 emails with short text samples, rather than summarizing results into one metric. The importance of this work is that it establishes a foundational idea in authorship analysis: that author identification can be achieved using writing style alone. However, the study also has limitations, including the use of a relatively small dataset, sensitivity to topic changes in the text, and limited scalability when applied to larger or more diverse datasets.

3.2.3 Author Profiling for English Emails

This study by Dominique Estival et al [20], extends authorship analysis to author profiling by moving beyond simply identifying who wrote a text to also predicting personal characteristics of the author. The study uses the Text Attribution Tool (TAT) and applies machine learning classification methods that rely on a combination of stylometric and linguistic features. These features capture different aspects of writing such as style, vocabulary choices, text structure, and general linguistic patterns present in email communication. Using this information, the system is designed to infer several attributes of the writer, including gender, age, native language, education level, and personality traits, which makes the task a multi-label, multi-task learning problem rather than a single prediction task. Because each attribute is evaluated separately, the paper does not provide one overall accuracy score; instead, it reports “promising results across the different profiling tasks. The main contribution of the work is demonstrating that email data can be used not only for identifying authors but also for building detailed user profiles. However, the study also has some limitations, such as not focusing solely on authorship identification, lacking a single unified performance metric, and depending on access to labeled personal data for training the models.

3.2.4 Classification Approach to Email Authorship Identification

This article "Analyzing Dataset (Email) by Using Classification Approach to Authorship Identification" by Khaled Ahmed Adbeib [21], focuses directly on the problem of authorship identification using machine learning classification techniques applied to email datasets. Unlike more complex or hybrid approaches, this work presents a straightforward and practical machine learning solution to determine the author of an email based on its textual content. The methodology relies on applying standard classification algorithms to email text data, making the overall approach relatively simple compared to deep learning or multi-model systems. The features used in the study are mainly text-based and include word frequency patterns along with basic linguistic features extracted from the emails. The key contribution of the paper is showing that even basic classification methods can still be effective in identifying email authors, proving that strong results can be achieved without overly complex models. However, the paper does not clearly report a single accuracy value. Instead, it presents performance comparisons between different classification algorithms without summarizing them into one unified metric. The importance of this study lies in its role as a baseline approach, offering a simpler alternative to more advanced systems while still addressing the core task of authorship identification. At the same time, it has several limitations, including the use of less advanced models, limited diversity in feature selection, and likely lower performance when compared to more sophisticated hybrid methods.

3.3 Comparison Tables

#	Title	Authors	Year	Dataset / Do-main	Methods Used
1	The Email Author Identification System Based on SVM and AHP [23]	Qinghe Zheng, Xinyu Tian, Mingqiang Yang, Huake Su.	2019	Email (Chinese)	TF-IDF + AHP + SVM
2	Identifying the Authors of Suspect Email. [22]	Malcolm Corney, Alison Anderson, George Mohay, Olivier de Vel.	2002	Email (English)	Stylometry + SVM
3	Author Profiling for English Emails [20]	Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford, Ben Hutchinson.	2010s	Email (English)	TAT (Stylometric + ML features)
4	Analyzing Dataset (Email) by Using Classification Approach to Authorship Identification [21]	Khaled Ahmed Adbeib	2023	Email dataset	KNN

Table 3.1: Comparison between articles (basic information)

#	Accuracy	Advantages	Limitations
1	95%	High accuracy, Combines multiple linguistic features, AHP improves feature importance.	Language-dependent, Manual feature engineering, Limited to SVM.
2	80%	Works with small datasets, Strong stylometric foundation, Email-specific features.	Small dataset, Sensitive to topic variation, Limited scalability.
3	Uses multiple metrics (multi-task results)	Combines linguistic + ML features, Email-focused, Goes beyond identification.	Not purely authorship identification, Requires labeled data, Generalization issues
4	80%	Direct focus on authorship identification, Uses classification techniques, Simpler and practical approach.	Less advanced models, Limited feature diversity, Likely lower scalability/accuracy.

Table 3.2: Accuracy, advantages and limitations comparison

3.4 Conclusion

The analyzed articles collectively show that machine learning methods combined with appropriate feature extraction can efficiently tackle email author identification. Article One emphasizes the necessity of assigning weights to linguistic features to enhance classification accuracy, while Article Two demonstrates that stylometric characteristics such as writing style can serve as a reliable author “fingerprint.” Furthermore, Article Three shows that rich linguistic features are not only important for identification but can also be used for author profiling by indicating the depth of information given by email content. Finally, Article Four demonstrates the real-world efficacy of using standard classification algorithms on email datasets.

Together, these works support the use of machine learning models such as Naive Bayes and Support Vector Machines, highlighting that performance depends on both the choice of algorithm and the quality of extracted features.

Chapter 4

Contribution

4.1 Introduction

The graphic design of an application plays an important role in improving user experience, usability, and system accessibility. A well-designed interface helps users navigate the application smoothly and understand its features with minimal effort. In the context of the Email Author Identification system, the graphical user interface (GUI) was designed to be simple, practical, and user-friendly to analyze email texts and predict their potential authors.

Different screenshots of the application are included to illustrate the design and functionalities of each component.

4.2 Dataset

To ensure a more realistic and practical experimentation process, we created our own data set by developing a custom code that can extract emails from specific email addresses. Instead of relying entirely on publicly available datasets, this approach allowed us to work with real email exchanges, making the Email Author Identification process closer to real-world scenarios.

By creating a customized dataset, we aimed to improve the realism and reliability of the experiment while better evaluating the system's ability to identify authors based on their writing characteristics in actual email communications.

The dataset includes the email author, representing the sender of the email, along with the email address, which helps uniquely identify the source of the message. It also contains subject , which is the title of the

email whenever it is available, as some emails may not include a subject. In addition, the dataset stores the email body, which contains the main textual content of the message. This component is particularly important since it serves as the primary source for extracting writing style features and identifying linguistic patterns specific to each author.

To provide more contextual information, the dataset also includes the date and time at which each email was sent. These temporal attributes can help better organize the data and offer additional insights into communication behaviors.

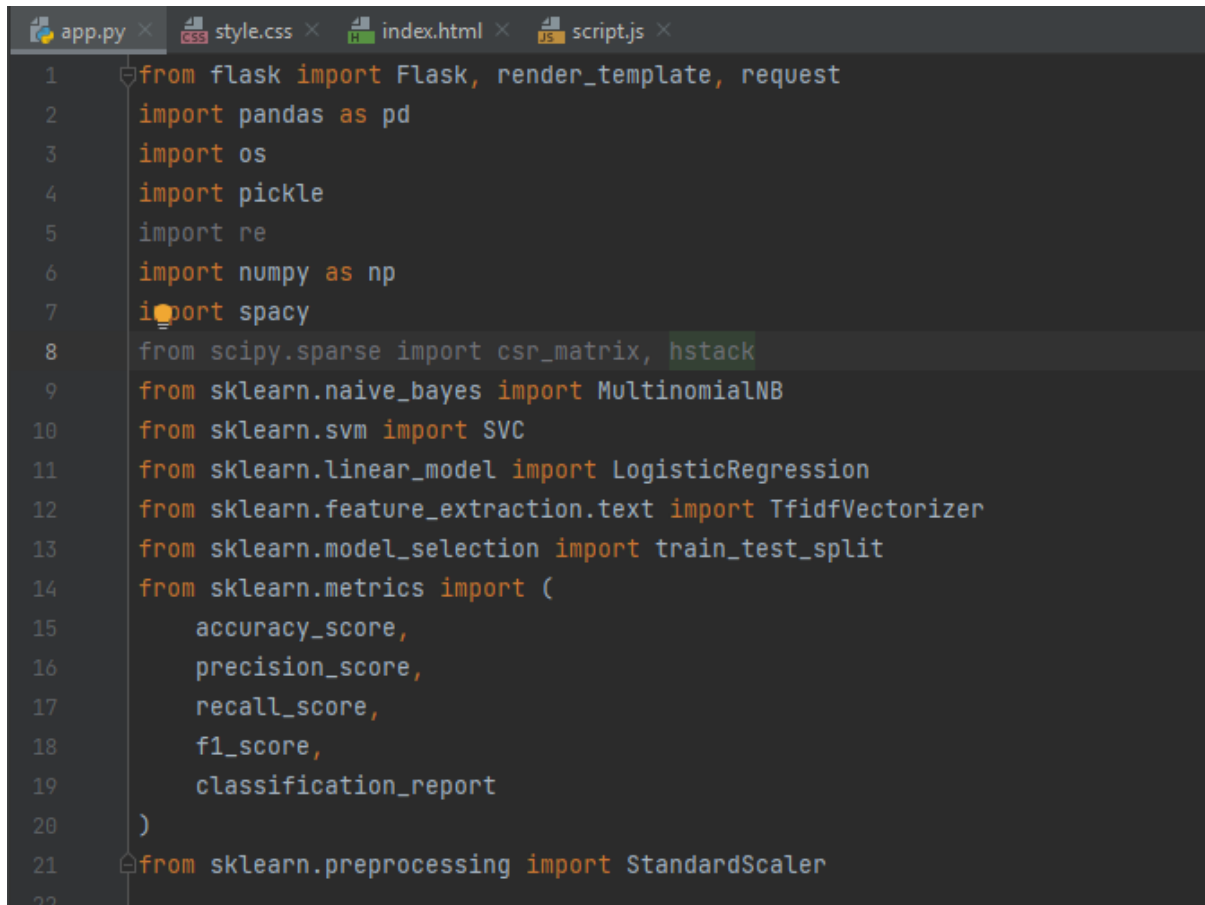
By including these different attributes, the dataset offers a more structured and realistic representation of email exchanges, allowing the system to perform author identification in conditions that closely resemble real-world scenarios.

Dataset	Authors	Emails	Distribution
Not Well Distributed	43	3,952	Imbalanced (different number of emails per author but each author has minimum 20 emails)
Well Distributed	32	960	Balanced (30 emails per author)

Table 4.1: Summary of the Datasets Used

Keeping in mind, the system supports three different dataset formats: (.txt, .xlsx, and .csv). This flexibility allows users to work with datasets stored in different file types based on their preferences or available resources. By supporting multiple formats, the system becomes easier to use and more adaptable

4.3 Libraries Used

A screenshot of a code editor with four tabs: 'app.py', 'style.css', 'index.html', and 'script.js'. The 'app.py' tab is active, showing a list of Python imports. The code is as follows:

```
1 from flask import Flask, render_template, request
2 import pandas as pd
3 import os
4 import pickle
5 import re
6 import numpy as np
7 import spacy
8 from scipy.sparse import csr_matrix, hstack
9 from sklearn.naive_bayes import MultinomialNB
10 from sklearn.svm import SVC
11 from sklearn.linear_model import LogisticRegression
12 from sklearn.feature_extraction.text import TfidfVectorizer
13 from sklearn.model_selection import train_test_split
14 from sklearn.metrics import (
15     accuracy_score,
16     precision_score,
17     recall_score,
18     f1_score,
19     classification_report
20 )
21 from sklearn.preprocessing import StandardScaler
```

Figure 4.1: Libraries

4.3.1 Flask (Web Framework)

Flask is used to build the web interface and manage all user interactions within the application[27]. It handles routing through decorators such as `@app.route("/")`, allowing different functions to respond to HTTP requests. Flask also manages the reception of user inputs including email content, subject lines, and dataset uploads. Additionally, it is responsible for sending processed results back to HTML templates and coordinating the overall workflow of training and prediction actions in the system.

4.3.2 Pandas (Data Handling)

Pandas is used for loading, cleaning, and processing datasets in multiple formats such as CSV, Excel, and TXT files[28]. It provides efficient tools for handling missing values, removing duplicates, and manipulating structured data. In this project, Pandas is also used to transform raw email

data by processing columns such as subject, email body, time, and date, and by generating additional features like hour of the day and day of the week.

4.3.3 NumPy (Numerical Computation)

NumPy is used to perform efficient numerical operations and manage multi-dimensional arrays. In this project, it plays a key role in converting spaCy embeddings into array format and combining different feature representations such as TF-IDF vectors and embeddings using horizontal stacking[29]. This enables the integration of heterogeneous data sources into a unified numerical representation suitable for machine learning models.

4.3.4 Scikit-learn (Machine Learning)

scikit-learn is the core machine learning library used in this project, providing a wide range of tools for feature extraction, model training, and evaluation[30]. It is used to convert text into numerical representations using TF-IDF vectorization and to train classification models such as Multinomial Naive Bayes, Support Vector Machines, and Logistic Regression. It also supports dataset splitting into training and testing sets, computation of evaluation metrics such as accuracy, precision, recall, and F1-score, and feature scaling through StandardScaler.

4.3.5 spaCy (Word Embeddings)

spaCy is used for natural language processing and semantic representation of text. The pretrained model `en_core_web_md` is loaded to generate dense word embeddings of 300 dimensions[31]. These embeddings capture contextual and semantic meaning beyond simple word frequency, allowing the model to better understand relationships between words and improve classification performance.

4.3.6 SciPy (Sparse Matrix Handling)

SciPy is used to support efficient handling of sparse matrices generated by the TF-IDF vectorizer[32]. Since text data often results in high-dimensional sparse representations, SciPy provides optimized data structures and operations that reduce memory usage and improve computational efficiency during model training and prediction.

4.3.7 Pickle (Model Persistence)

pickle is used for model persistence, allowing trained machine learning models, vectorizers, and scalers to be saved to disk and reused later[33]. This eliminates the need to retrain the model every time the application runs and ensures consistency between training and prediction phases by reloading the exact same objects.

4.3.8 Regular Expressions (re module)

Python's re module is used for text preprocessing and pattern matching. It is responsible for cleaning raw text by removing numbers and punctuation and normalizing the input data[34]. It is also used in the visualization component of the system to highlight important words within the email content based on their relevance to the model's prediction.

4.3.9 OS (Operating System Interface)

os is a built-in Python library used to interact with the operating system and manage file and directory operations within the application[35]. In this project, it is used to check the existence of saved model files such as the trained model, vectorizer, and scaler before loading them. It also helps in handling file paths and extracting file extensions from uploaded datasets to correctly determine their format (CSV, Excel, or TXT). This ensures smooth file management and prevents errors during model loading and dataset processing.

4.4 Summary of the System

The system is divided into two main phases: the Training Phase and the Prediction Phase. Each phase behaves differently depending on the selected machine learning model, which includes Naive Bayes (NB), Support Vector Machine (SVM), and Logistic Regression (LR).

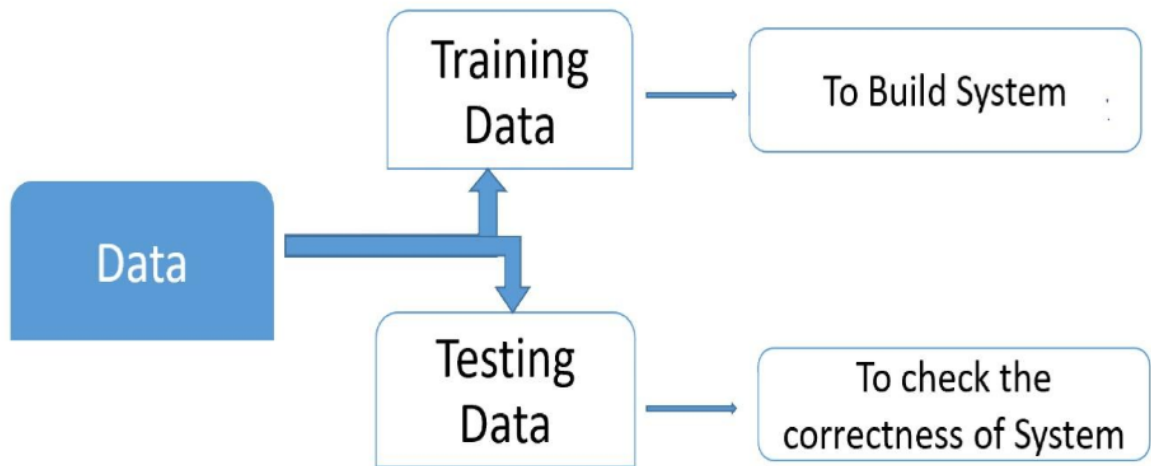


Figure 4.2: Train and Test(Generated by AI)

4.4.1 Training Phase

The first step is the training phase, responsible for learning patterns from labeled email dataset and building a machine learning model capable of identifying the author of a given email. The process is divided into several structured steps to ensure proper preprocessing, feature extraction, model training, and evaluation.

Step 1: Dataset Loading

The system first loads the dataset uploaded by the user. It supports multiple formats such as CSV, Excel, or cached data from previous sessions. If no dataset is available, the system stops and requests the user to upload one.

```

1 file = request.files.get("dataset")
2 df = None
3
4 if file and file.filename:
5     df = load_dataset(file)
6     cached_df = df.copy()
7     current_dataset_name = file.filename
8     dataset_loaded = True
9
10 elif cached_df is not None:
11     df = cached_df.copy()
12
13 else:
14     return render_template("index.html", error="Please upload
    dataset first")
  
```

Step 2: Data Cleaning and Preprocessing

The dataset is cleaned by removing noise, handling missing values, and standardizing the text. This step ensures that the model receives high-quality input data.

```
1 df = clean_data(df)
2 df = process_datetime(df)
```

Step 3: Feature Engineering

The subject, email content, and time-based features are combined into a single text representation. This allows the model to learn both linguistic and behavioral patterns.

```
1 df["text"] = (
2     df["subject"] + " " +
3     df["email"] + " " +
4     "hour_" + df["hour"].astype(str) + " " +
5     "day_" + df["dayofweek"].astype(str)
6 )
```

Step 4: Splitting Dataset

The dataset is divided into training and testing sets to evaluate model performance on unseen data.

```
1 X = df["text"]
2 y = df["author"]
3
4 X_train, X_test, y_train, y_test = train_test_split(
5     X, y, test_size=0.2, random_state=42
6 )
```

the training takes 80% of the dataset and the testing is applied on the remaining 20% of the dataset.

Step 5: TF-IDF Vectorization

The text data is converted into numerical form using TF-IDF vectorization, which represents the importance of words in the dataset.

```
1 vectorizer = TfidfVectorizer(max_features=5000)
2 X_train_vec = vectorizer.fit_transform(X_train)
3 X_test_vec = vectorizer.transform(X_test)
```

Step 6: Model Selection and Training

The model is selected based on user input. If Naive Bayes is chosen, only TF-IDF features are used. For SVM and Logistic Regression, embeddings are added and scaling is applied.

Naive Bayes Model

```
1 if model_type == "nb":
2     model = MultinomialNB()
3     model.fit(X_train_vec, y_train)
4     y_pred = model.predict(X_test_vec)
```

Naive Bayes uses probabilistic learning and works directly with TF-IDF features without requiring scaling or embeddings.

SVM and Logistic Regression Models

```
1 X_train_dense = X_train_vec.toarray()
2 X_test_dense = X_test_vec.toarray()
3
4 emb_train = np.array([get_embedding(t) for t in X_train])
5 emb_test = np.array([get_embedding(t) for t in X_test])
6
7 scaler = StandardScaler()
8
9 X_train_final = scaler.fit_transform(
10     np.hstack([X_train_dense, emb_train])
11 )
12 X_test_final = scaler.transform(
13     np.hstack([X_test_dense, emb_test])
14 )
```

For these models, TF-IDF features are combined with spaCy embeddings to capture both lexical and semantic meaning.

Step 7: Training SVM or Logistic Regression

```
1 if model_type == "svm":
2     model = SVC(kernel="linear", C=2.0, probability=True)
3
4 elif model_type == "lr":
5     model = LogisticRegression(max_iter=2000, C=3.0)
6
7 model.fit(X_train_final, y_train)
8 y_pred = model.predict(X_test_final)
```

SVM finds the optimal separating hyperplane between classes, while Logistic Regression estimates probabilities for each class.

Step 8: Model Evaluation and Saving

The model is evaluated using standard metrics and then saved for future use.

```
1 train_accuracy = round(accuracy_score(y_test, y_pred) * 100,
2                             2)
3 train_precision = round(precision_score(y_test, y_pred,
4                                     average="weighted") * 100, 2)
5 train_recall = round(recall_score(y_test, y_pred, average="
6     weighted") * 100, 2)
7 train_f1 = round(f1_score(y_test, y_pred) * 100, 2)
8 train_report = classification_report(y_test, y_pred)
9
10 pickle.dump(model, open(MODEL_FILE, "wb"))
11 pickle.dump(vectorizer, open(VEC_FILE, "wb"))
12 pickle.dump(scaler, open(SCALER_FILE, "wb"))
```

4.4.2 Naive Bayes (NB) Model

The Naive Bayes model is a probabilistic classifier that is particularly effective for text classification tasks using TF-IDF features. During training, the TF-IDF matrix is directly used as input without requiring feature scaling or additional transformations. The model learns the probability distribution of words for each author by estimating how frequently specific words appear within each class. Based on these probabilities, it determines the likelihood that a given word belongs to a particular author class.

The Naive Bayes model is based on the assumption that all features are conditionally independent given the class label. Despite this simplifying assumption, it performs very well in text classification problems due to the strong relationship between word frequency and class labels. The final output is a lightweight and efficient model that is well suited for fast text classification tasks.

4.4.3 Support Vector Machine (SVM)

The Support Vector Machine model is a supervised learning algorithm designed to find the optimal decision boundary that separates different author classes. During training, TF-IDF features are first converted into dense vectors, and semantic information is added using spaCy word embeddings. These two feature representations are then combined into a single feature vector. Since SVM is sensitive to feature magnitude, feature scaling is applied using StandardScaler.

The model then learns a linear hyperplane that maximizes the margin between different classes. The main idea behind SVM is to achieve the best possible separation between author classes by maximizing the distance between decision boundaries. This makes the model highly effective for high-dimensional data and complex classification problems, often resulting in high accuracy.

4.4.4 Logistic Regression (LR)

Logistic Regression is a statistical model used for multi-class classification problems. In this system, it follows the same feature engineering pipeline as SVM, where TF-IDF features are combined with spaCy embeddings and normalized using StandardScaler. During training, the model learns a set of weights that represent the contribution of each feature to each author class.

The model optimizes a probabilistic function that estimates the likelihood that a given email belongs to a specific author. Unlike SVM, Logistic Regression provides direct probability outputs, making its predictions more interpretable. It is a stable and reliable model that performs well when the relationship between features and classes is approximately linear.

4.4.5 Prediction Phase

The prediction phase is responsible for classifying a new email into its most likely author using the trained machine learning model. The process follows several structured steps to ensure consistency with the training phase.

Step 1: Checking Model Availability

Before making any prediction, the system first verifies whether a trained model is loaded. If no model exists, the process is stopped and the user is prompted to train the model first.

```
1 if model is None:
2     error = "Train model first"
```

Listing 4.1: Prediction Phase

Step 2: Reading and Cleaning User Input

The system retrieves the subject and email content from the user input form. Both fields are cleaned using the same preprocessing function used

during training to ensure consistency.

```
1 subject = advanced_clean(request.form.get("subject", ""))
2 email = advanced_clean(request.form.get("email", ""))
3 text = subject + " " + email
```

Step 3: Text Vectorization

The cleaned text is transformed into a numerical representation using the previously trained TF-IDF vectorizer. This ensures that the input format matches the training data.

```
1 X_vec = vectorizer.transform([text])
```

Step 4: Naive Bayes Explainability (Optional)

If the selected model is Naive Bayes, the system extracts the most influential words contributing to the prediction. These words are later used for visualization and highlighting.

```
1 if current_model_type == "nb":
2     top_words = get_top_words(vectorizer, model, text)
3     words = [w for w, s in top_words]
4     scores = [float(s) for w, s in top_words]
5     highlighted_email = highlight_text(text, top_words)
```

Step 5: Feature Preparation for Prediction

If the model is Naive Bayes, only TF-IDF features are used. For SVM and Logistic Regression, TF-IDF features are combined with spaCy embeddings, and scaling is applied to match the training distribution.

```
1 if current_model_type == "nb":
2     X_final = X_vec
3 else:
4     X_vec_dense = X_vec.toarray()
5     emb = np.array([get_embedding(text)])
6     X_final = scaler.transform(np.hstack([X_vec_dense, emb]))
```

Step 6: Prediction Generation

The trained model is used to predict the most likely author of the email.

```
1 prediction = model.predict(X_final)[0]
```

Step 7: Confidence Score Calculation

If the model supports probability estimation, the system calculates a confidence score indicating how certain the prediction is. Otherwise, the confidence is marked as not available.

```
1 if hasattr(model, "predict_proba"):
2     confidence = round(max(model.predict_proba(X_final)[0]) *
3                          100, 2)
4 else:
5     confidence = "N/A"
```

Naive Bayes (NB) Prediction

In the Naive Bayes prediction phase, the input email is transformed using only TF-IDF features. The model computes the probability of the email belonging to each author class and selects the class with the highest probability as the final prediction. Additionally, the system extracts the most influential words contributing to the decision based on learned feature probabilities. These words can be used to highlight important parts of the email, providing interpretability for the prediction.

Support Vector Machine (SVM) Prediction

For SVM, the input email is transformed using both TF-IDF features and spaCy embeddings in order to capture both lexical and semantic meaning. The same scaling applied during training is also applied during prediction. The model then determines the position of the input relative to the decision boundary and assigns the class based on the closest margin. This geometric approach allows SVM to produce highly accurate predictions, particularly in cases where class separation is complex.

Logistic Regression (LR) Prediction

In Logistic Regression prediction, the same feature pipeline used in training is applied, combining TF-IDF vectors with embeddings and scaling. The model computes probability values for each author class using learned feature weights. The class with the highest probability is selected as the final prediction. Because the output is probabilistic, Logistic Regression provides more interpretable results, allowing the confidence of each prediction to be understood rather than only the final label.

4.5 Models Behavior Summary

Model	Training Style	Prediction Style	Strength
Naive Bayes	Probabilistic word learning	TF-IDF only	Fast & interpretable
SVM	Margin-based classification	TF-IDF + embeddings	High accuracy
Logistic Regression	Probability-weight learning	TF-IDF + embeddings	Stable & balanced

Table 4.2: Comparison of Machine Learning Models

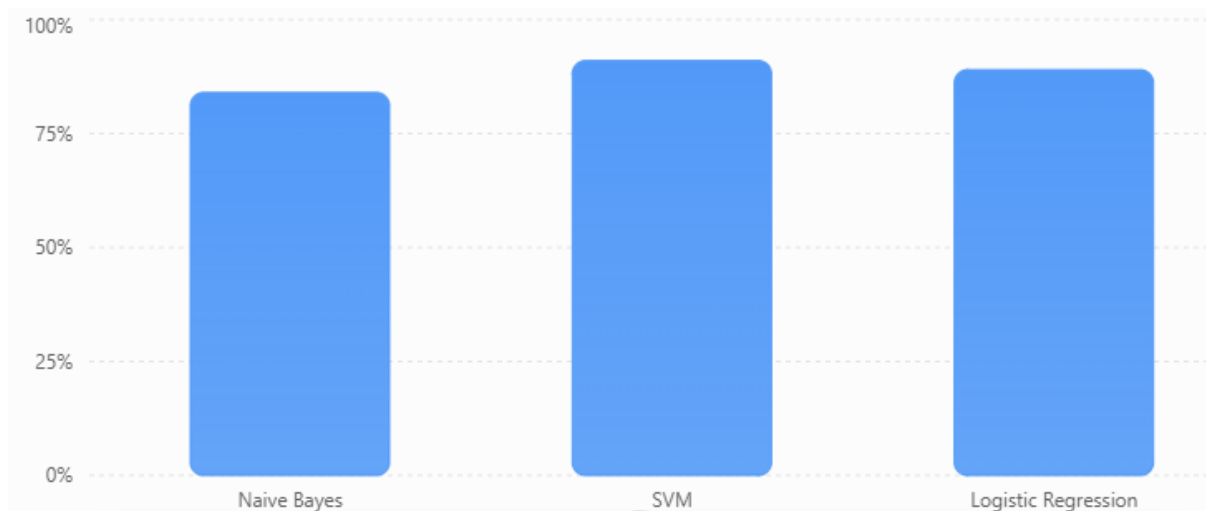


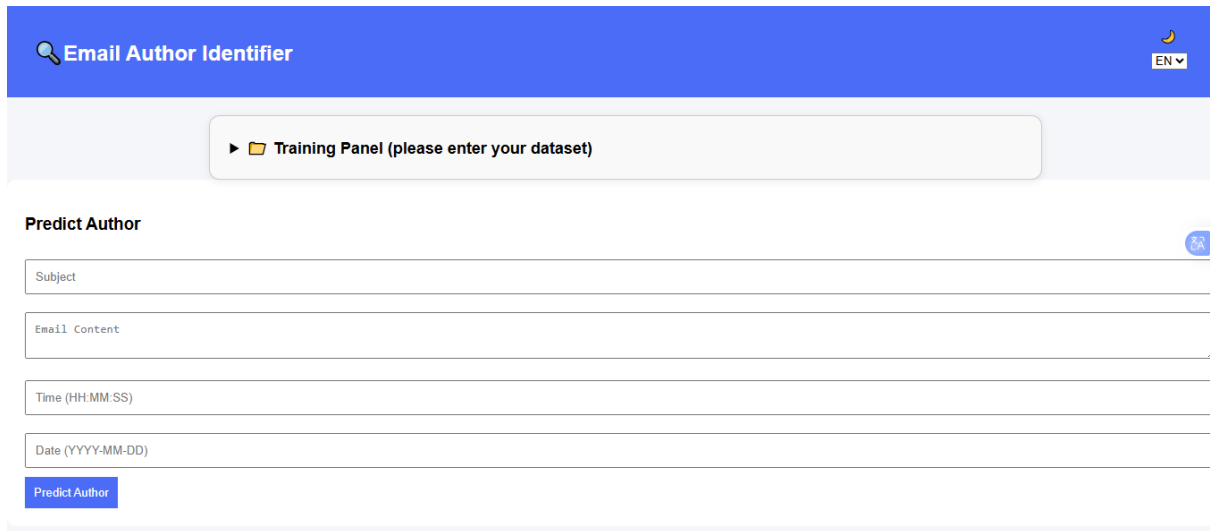
Figure 4.3: Models Accuracy (Generated by AI)

4.6 Overview

The Email Author Identification website was developed to provide users with an easy and interactive way to analyze email content and determine its most likely author using machine learning techniques. The interface was designed with simplicity and clarity in mind to ensure smooth navigation and ease of use.

The following image presents the main and only page that users encounter when accessing the website. Since the application was designed

with a simple and user-friendly interface, all essential functionalities are gathered within a single page to ensure easy navigation and accessibility.



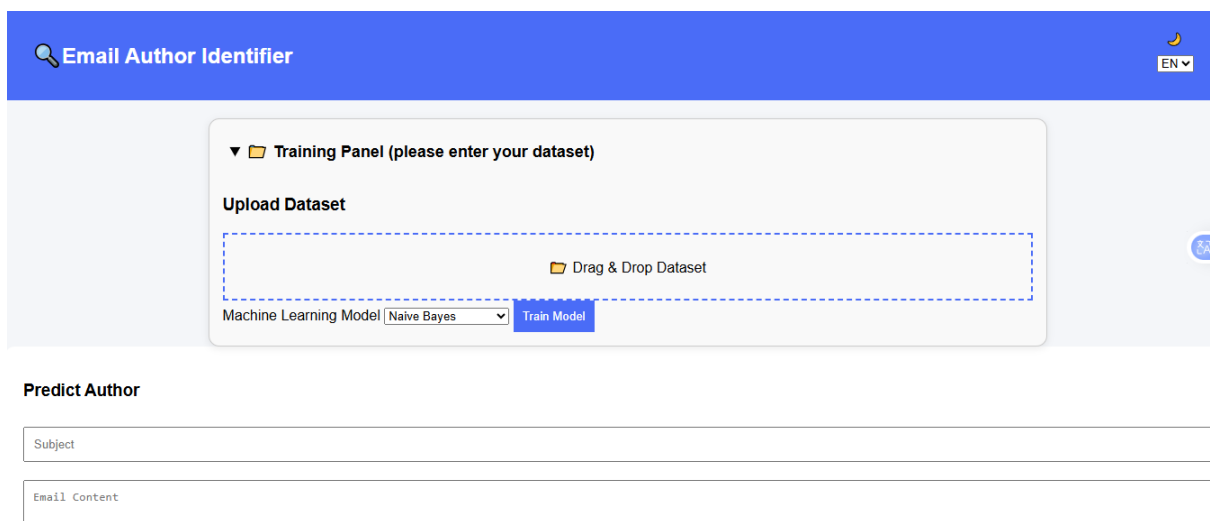
The screenshot shows the main interface of the 'Email Author Identifier' application. At the top is a blue header with the application name and a language dropdown set to 'EN'. Below the header is a light gray box containing a folder icon and the text 'Training Panel (please enter your dataset)'. The main section is titled 'Predict Author' and contains four input fields: 'Subject', 'Email Content', 'Time (HH:MM:SS)', and 'Date (YYYY-MM-DD)'. A blue 'Predict Author' button is located at the bottom of this section.

Figure 4.4: Main Interface

In the following sections, each component of the interface will be explained in detail, highlighting its purpose and the role it plays in the Email Author Identification process. Every feature will be discussed separately to provide a clearer understanding of how users interact with the system.

4.6.1 Training Section

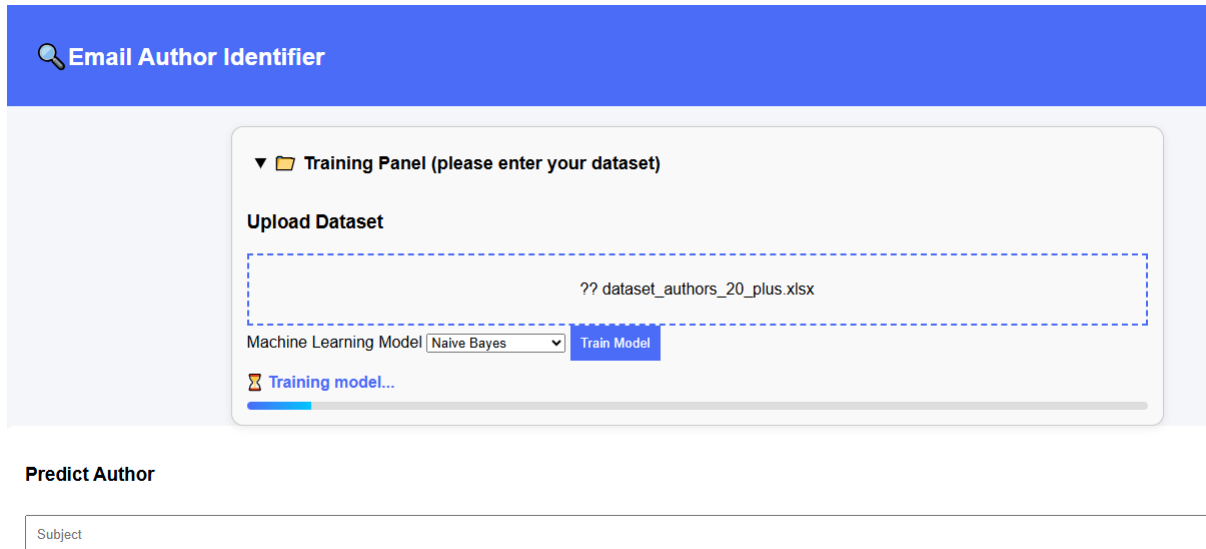
The first section of the website is called the Training Panel, where users begin by preparing the system for the author identification process.



The screenshot shows the 'Training Section' of the application. It features a blue header with the application name and a language dropdown set to 'EN'. Below the header is a light gray box containing a folder icon and the text 'Training Panel (please enter your dataset)'. Inside this box is a section titled 'Upload Dataset' which includes a dashed blue box for 'Drag & Drop Dataset'. Below the drop zone is a dropdown menu for 'Machine Learning Model' currently set to 'Naive Bayes', and a blue 'Train Model' button. Below the training section is the 'Predict Author' section, which contains the same four input fields as in Figure 4.4: 'Subject', 'Email Content', 'Time (HH:MM:SS)', and 'Date (YYYY-MM-DD)'.

Figure 4.5: Training Section

In this section, the user is required to upload or enter the dataset and select the machine learning model they wish to work with. Once these steps are completed, the user can start the training process by pressing the corresponding button.



The screenshot shows a web interface for an "Email Author Identifier". At the top is a blue header with a magnifying glass icon and the text "Email Author Identifier". Below this is a "Training Panel (please enter your dataset)" section. Inside this panel, there is an "Upload Dataset" section with a dashed blue border containing the text "?? dataset_authors_20_plus.xlsx". Below the upload area is a "Machine Learning Model" dropdown menu set to "Naive Bayes" and a blue "Train Model" button. At the bottom of the panel is a "Training model..." progress bar with a small blue segment. Below the training panel is a "Predict Author" section with a text input field containing the word "Subject".

Figure 4.6: Training Load

After the training begins, the system processes the dataset and displays the training results. Users can scroll down to view different evaluation metrics and performance percentages that help estimate how accurately the selected model performs on the provided data. These metrics offer a better understanding of the model's effectiveness and overall performance.

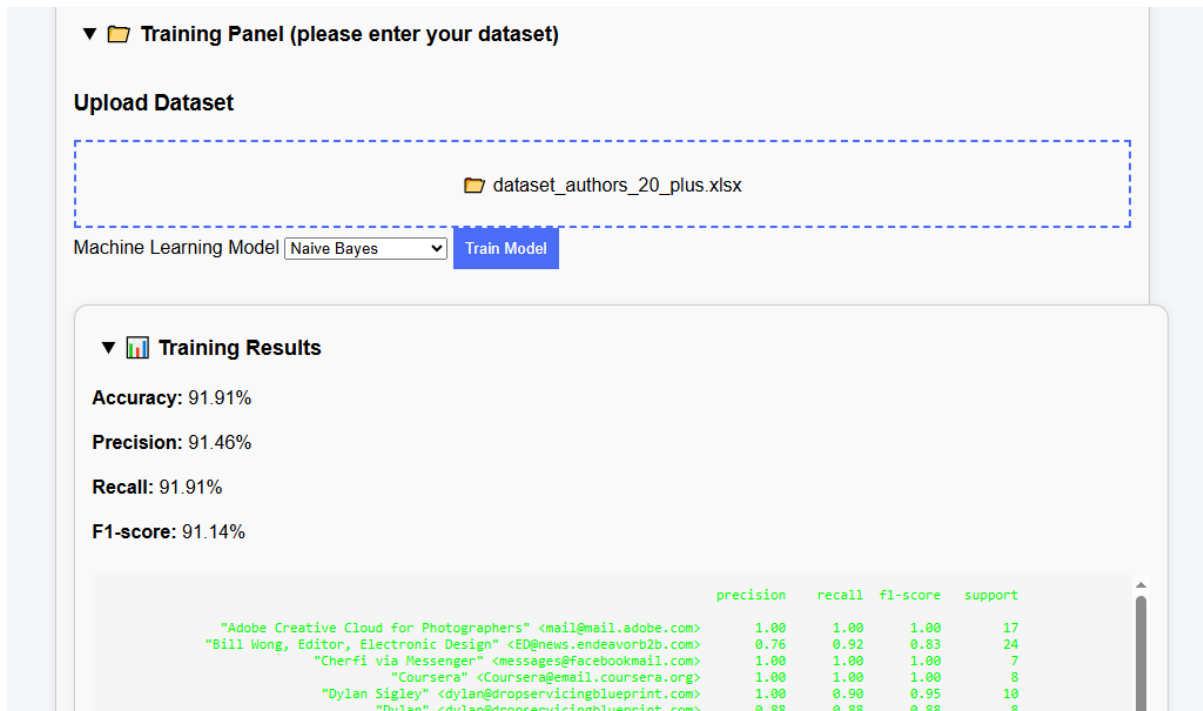


Figure 4.7: Evaluation Matrices

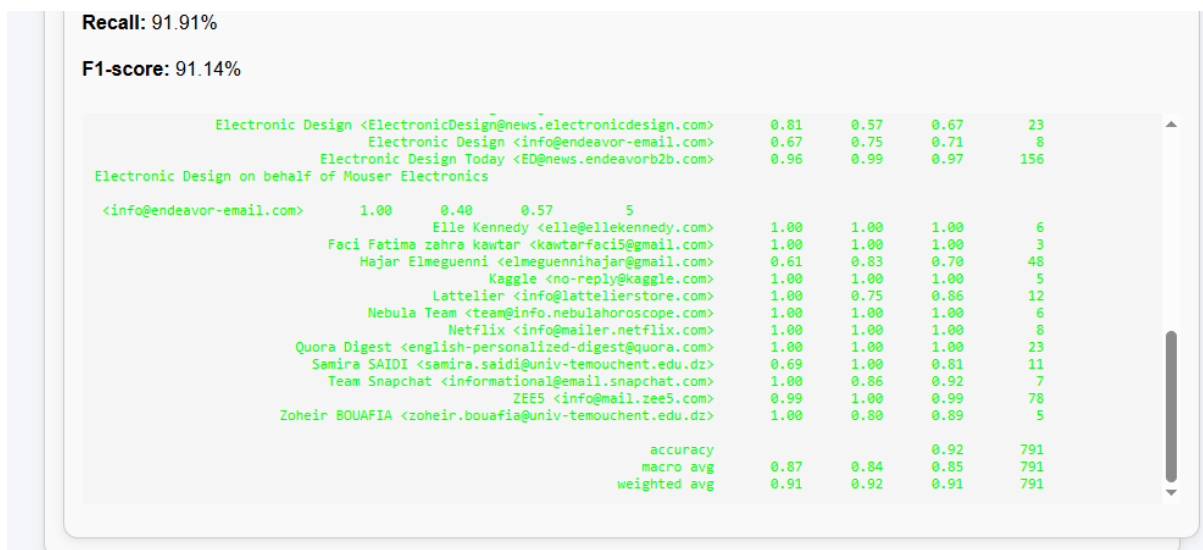
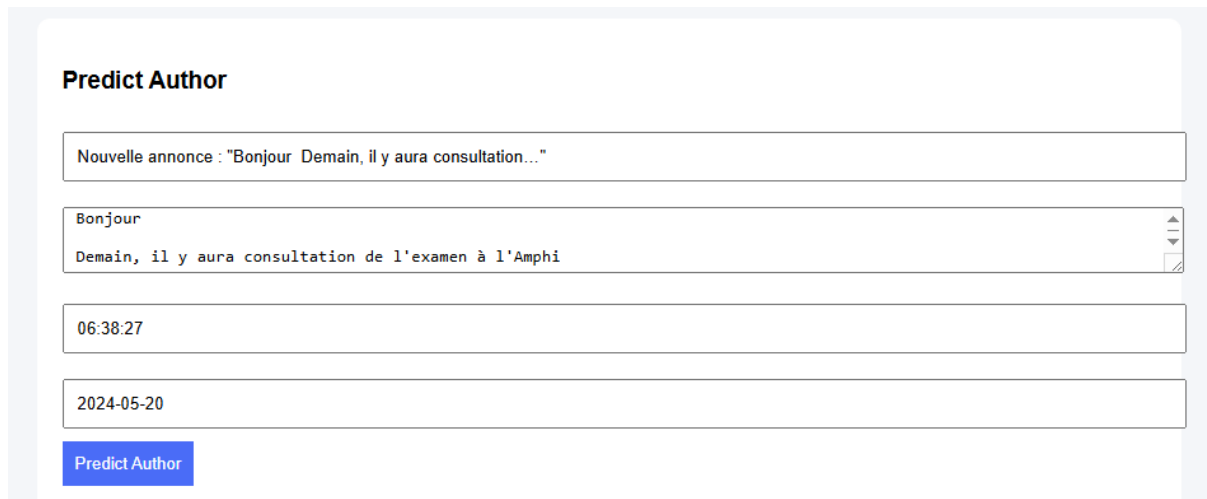


Figure 4.8: Evaluation Metrics For Each Author

It is important to note that the efficiency of the system depends heavily on two main factors: the quality of the dataset and the machine learning model selected. A well-structured dataset combined with an appropriate model generally leads to better prediction accuracy and more reliable results.

4.6.2 Prediction Section

Once the training process is completed, the user can proceed to the prediction phase. In this step, the user fills in the required email information fields, such as the email content and related details, then presses the Predict button. The system then analyzes the input and predicts the most likely author of the email.

The image shows a web form titled "Predict Author" with a light blue header. Below the title, there are four input fields. The first field contains the text "Nouvelle annonce : 'Bonjour Demain, il y aura consultation...'". The second field is a text area containing "Bonjour" and "Demain, il y aura consultation de l'examen à l'Amphi" with a small icon in the bottom right corner. The third field contains the time "06:38:27". The fourth field contains the date "2024-05-20". At the bottom of the form is a blue button labeled "Predict Author".

Predict Author

Nouvelle annonce : "Bonjour Demain, il y aura consultation..."

Bonjour
Demain, il y aura consultation de l'examen à l'Amphi

06:38:27

2024-05-20

Predict Author

Figure 4.9: Prediction Example

After the prediction is completed, the website displays the predicted author's name along with an accuracy percentage, indicating the confidence level of the prediction. Additionally, the system highlights a set of important words extracted from the email. These highlighted terms help explain why the prediction was made, offering users greater transparency and making the system more understandable and trustworthy.

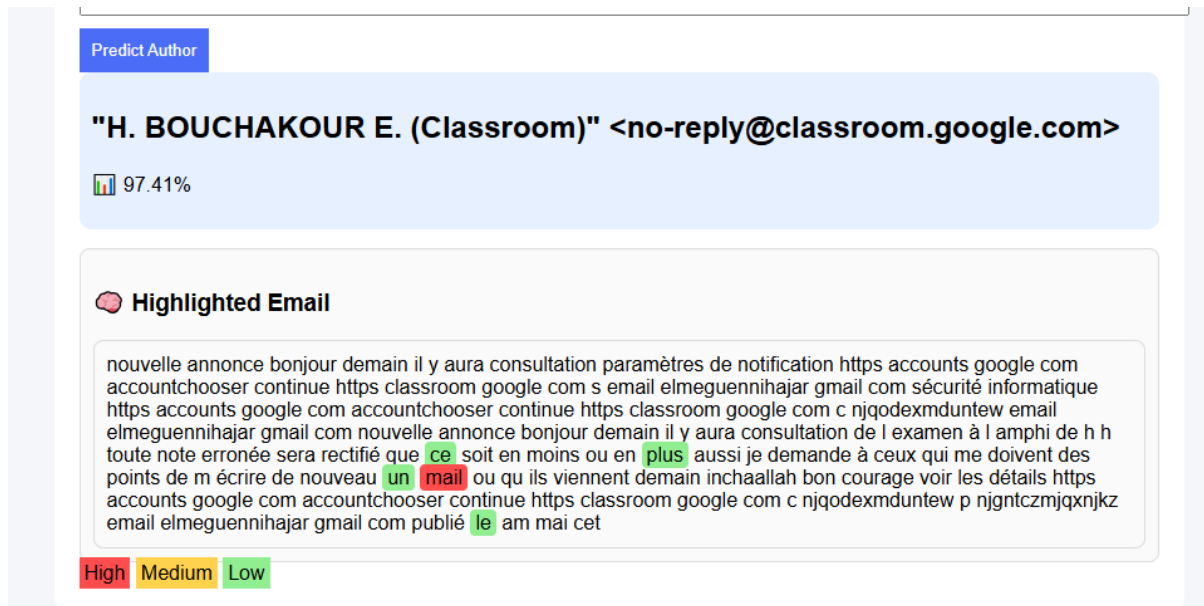


Figure 4.10: Prediction Result

4.7 Conclusion

In this chapter, we explored the graphical design of the Email Author Identification website and explained the purpose of its different components. The website was designed with the goal of creating a simple and practical environment where users can easily interact with the system without facing unnecessary difficulties. By keeping all functionalities within a clear and organized interface, users are able to move smoothly between the training and prediction processes, all while increasing the reliability and trustworthiness of the final results.

General Conclusion

Three machine learning models; Naive Bayes (NB), Logistic Regression (LR), and Support Vector Machine (SVM) were trained and evaluated successfully. The implemented design allowed for an intelligent email author identification system using machine learning technique, with the basis of analyzing email content and predicting the most probable author based on writing patterns and textual characteristics. While the obtained results allowed us to identify the strengths and weaknesses of each algorithm according to several evaluation metrics such as accuracy, precision, recall, and F1-score. By focusing on several fields including machine learning, natural language processing, cybersecurity, and web application development. We gained knowledge in many aspects such as TF-IDF vectorization, semantic embeddings and how style-based features can improve authorship identification performance. We also learned how to preprocess datasets, train classification models, evaluate their performance and integrate the entire system into a functional website. We did encounter some difficulties during this project such as making sure several dataset format were excepted to prioritize users comfort, that included encoding issues like txt files. We also faced a challenging accuracy so we had to balance the models capability along with picking the proper data to have higher dependable accuracy results. That required adding additional algorithms and careful parameter tuning for each model to enhance the performance and achieve stable and accurate results.

For future perspective, we can see this project being improved in several ways. Future work may include additional machine learning and deep learning models such as Random Forests, LSTM networks, or Transformer-based architectures like BERT in order to improve prediction accuracy. Or aiming the work to handle larger multilingual datasets, with advanced techniques and functionalities. Another aspect is to add multiple training and testing on the same dataset since our system goes through one training/test phase. Furthermore, we can extend the system by making it a real-time email analysis applied to your email account. Adding this to any email application(gmail for example) could make the system more scalable and practical for immediate-use serving as a technique to help with cybersecurity.

In conclusion, this project demonstrates that machine learning techniques can effectively contribute to email author identification and cybercrime investigation. The results confirm the importance of combining

textual, stylistic, and semantic features to improve classification performance. This work represents an important step toward the development of intelligent forensic systems by assisting cybersecurity professionals in identifying suspicious digital communications.

Bibliography

- [1] Murphy, K. P. (2012). Machine learning: A probabilistic perspective. MIT Press.
- [2] Alpaydin, E. (2020). Introduction to machine learning. MIT Press.
- [3] Russell, S., & Norvig, P. (2021). Artificial intelligence: A modern approach. Pearson.
- [4] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer.
- [5] Samuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 3 (3): 210–229.
- [6] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge University Press.
- [7] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R. Springer.
- [8] Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books.
- [9] Chapelle, O., Schölkopf, B., & Zien, A. (2006). Semi-supervised learning. MIT Press.
- [10] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.
- [11] Nilsson, Nils J. (1965). Learning Machines. McGraw-Hill.
- [12] Duda, R., Hart P. (1973). Pattern Recognition and Scene Analysis, Wiley Interscience.

- [13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.
- [14] Seeger, M. (2000). Learning with labeled and unlabeled data. University of Edinburgh.
- [15] Provost, F., & Fawcett, T. (2013). Data science for business. O'Reilly Media.
- [16] Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer.
- [17] Tan, P.-N., Steinbach, M., & Kumar, V. (2019). Introduction to data mining (2nd ed.). Pearson.
- [18] Aggarwal, C. C. (2015). Data mining: The textbook. Springer.
- [19] Tom M. Mitchell, machine learning, Publisher: McGraw-Hill Science/Engineering/Math; (March 1, 1997)
- [20] Estival, D., Gaustad, T., Pham, S. B., Radford, W., Hutchinson, B. (2010). Author profiling for English emails. In Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING 2010) (pp. 263–272).
- [21] Adbeib, K. A. (2023). Analyzing dataset (email) by using classification approach to authorship identification. North African Journal of Scientific Publishing (NAJSP), 1(4), 100–108. <https://doi.org/10.65414/najsp.v1i4.98>.
- [22] Corney, M., Anderson, A., Mohay, G., de Vel, O. (2002). Identifying the authors of suspect email. In Proceedings of the Australasian Information Security Workshop (AISW 2002), Gold Coast, Australia.
- [23] Zheng, Q., Tian, X., Yang, M., Su, H. (2019). The email author identification system based on Support Vector Machine (SVM) and Analytic Hierarchy Process (AHP). In Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC 2019).
- [24] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.
- [25] Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python. O'Reilly Media.

- [26] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1–47.
- [27] Grinberg, M. (2018). *Flask web development: Developing web applications with Python* (2nd ed.). O'Reilly Media.
- [28] McKinney, W. (2017). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.
- [29] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362.
- [30] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [31] Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed. draft). Stanford University.
- [32] Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272.
- [33] Van Rossum, G. (1991). *Python programming language reference manual*. Python Software Foundation.
- [34] Friedl, J. E. F. (2006). *Mastering regular expressions* (3rd ed.). O'Reilly Media.
- [35] Van Rossum, G. (1991). *Python programming language reference manual*. Python Software Foundation.
- [36] Renee Jones. March 20 2026. Four Types of Machine Learning You Should Know. NetCom Learning.
- [37] Underfitting-and-overfitting-in-machine-learning. [geeksforgeeks](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/).

All references cited in this study were available and accessible as of 6 July 2026.