Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
2025 | LIU-IDA/LITH-EX-A--2025/009--SE

Autonomous Email Categorization using Machine Learning Models in Thunderbird Client

Venkata Satyanarayana Vamsy Gonnabathula

Supervisor : Martin Funkquist
Examiner : Jendrik Seipp

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`.

© Venkata Satyanarayana Vamsy Gonnabathula

Abstract

Email categorization is a significant challenge in today's digital workplace, where sorting through mass incoming messages may become complicated. This thesis addresses this problem by developing and evaluating an autonomous email categorization system for the Thunderbird email client. Specifically, the study explores several machine learning models: Traditional algorithms (Logistic Regression, Support Vector Machines (SVM), Random Forest, XGBoost); and a Transformer-based model, Distil-BERT; The study also showcases the development of a custom Thunderbird extension - "EClassify", that integrates our machine learning models through a FastAPI backend service, providing empirical verification of the proposed approach. This implementation supports single and ensemble model predictions, offering flexibility in dealing with the trade-off between accuracy and computational resource usage. The findings of this research provide insights and lessons learned regarding the implementation, practical challenges, and future opportunities of automated email categorization, enriching the general understanding of using machine learning solutions for everyday productivity problems.

Acknowledgments

I take this opportunity to sincerely thank my examiner, Jendrik Seipp, and my supervisor, Martin Funkquist. Initially, suggesting this topic from the MRLAB research group sparked interest in exploring an area of autonomous email categorization using machine learning techniques. I also show my appreciation for their support in granting access to the computers in the AI Academy rooms at IDA, Linköping, which was essential for conducting experiments and development efforts that this research required.

Contents

List of Figures

List of Tables

1 Introduction

Professionals in our digital age battle the complicated challenge of managing emails, as overflowing inboxes require attention and organization. Sorting emails into folders manually takes a lot of time, in addition to causing delays in responding to important messages. Mainly devoting much time to organizational tasks and not concentrating on the content. The constant influx of communications can also be overwhelming, causing them to overlook critical messages. This leads to delays in decision-making, posing productivity challenges for organizations. Manually categorizing emails using the customary approach wastes important time and creates many consistent organization schemes that become increasingly more challenging to maintain as communication volumes rise. The cognitive burden of frequently switching between reading, categorizing, and responding to emails greatly diminishes overall workplace effectiveness and contributes to information overload. Organizations have reported that workplace productivity is reduced by up to 20 percent due to poor email management, concentrating on the important effect of this challenge on business operations [6].

Recognizing this issue, there is interest in leveraging machine learning models to automate the process of categorizing emails. Machine learning, a sub-field of artificial intelligence, actively develops algorithms and statistical models that allow computers to learn and make predictions or decisions without explicit programming. Machine learning aims to learn from data and improve performance over time. By analyzing large data sets, machines can be effectively trained to recognize trends, extract meaningful understandings, and make more accurate predictions or decisions. Machine learning's ability is an important practice and a tool for analyzing complicated problems that customary programming approaches find challenging. Multiple machine learning techniques exist, each with its approach and purpose [19]. Supervised learning is one common type where the algorithm learns from labelled data to make predictions or classifications. It requires a training dataset with input-output pairs to develop a model that can generalize to new, unseen data. Techniques such as decision trees, and support vector machines fall under this category, providing a wide range of tools for email classification.

Many machine learning algorithms actively classify emails by analyzing their content and contextual parameters. Autonomous email classification is a valuable technique for performing this task. These algorithms concentrate on multiple attributes such as subject lines,

sender information, and message content. Many emails can then be efficiently classified into pre-defined categories or tags. First, a more organized email environment is achieved, with messages being found and prioritized more effectively by users [7]. Organizations can save time from manual categorization to more productive activities, boosting workflow. Precise email classification also decreases the chances of missing important messages.

Some of the previous attempts to address this challenge have mainly depended on rule-based filtering systems or simple keyword-matching algorithms. While providing basic automation, these solutions fall short in two areas: understanding the context-specific subtleties in the content of an email and adaptability to changing patterns in communication. Rule-based systems need an ongoing maintenance effort to preserve their effectiveness and are not flexible enough to cope with the dynamic features in today's email communications. Moreover, current machine learning-based solutions to email classification typically focus on one particular use case, most notably spam detection, and not so much on generic content-based categorization, which would adapt to the needs of each user. In this research, we explore various machine learning algorithms - from simple baselines to more sophisticated models - we aim to open a pathway to more innovative email management, particularly within the context of email clients like Thunderbird[1], which is a widely used email client, that offers its users a platform for managing their email correspondence.

1.1 Aim

This master thesis aims to create and compare different machine learning models that learn to categorize incoming emails. Providing accurate predictions will help Thunderbird users handle their mail faster.

1.2 Research questions

1. How well do machine learning models perform at categorizing emails and how do they compare in terms of accuracy, model size and their ease of deployment in a Thunderbird plugin.
2. In comparing different machine learning models for email categorization, what trade-offs exist between accuracy, model size, code complexity, and ease of deployment? How do these trade-offs affect the overall effectiveness and practicality of using these models in Thunderbird plugin?
3. What challenges come up when implementing machine learning models for email categorization in Thunderbird, and how do these challenges affect performance and user experience? Based on these findings, which machine learning model and technique is best for autonomous email categorization in Thunderbird?
4. What are the implications of the chosen models for future research directions in the area of email classification?

1.3 Proposed Approach

The approach involves some key components :

Data Collection and Preparation: The Enron Corpus, a large email data set, is used to train and test machine learning models. This data set is rich in various email categories, which helps to develop practical classification algorithms.

[1] `https://www.thunderbird.net/en-US/`

Feature Extraction: A variety of feature extraction methods will be applied to quantitatively represent the textual content of emails. It includes methods like word frequency and n-gram models. This is better to capture the semantic and structural characteristics of emails.

Model Development: A few machine learning algorithms are implemented and compared, including but not limited to Logistic Regression, XGBoost, Support Vector Machines, Distil-BERT, Random Forest. These models are implemented using Python libraries such as SciKit Learn, TensorFlow-Keras, Pandas, and NumPy. The algorithms chosen are selected on the basis of their well-documented performance in text classification tasks and their ability to handle high-dimensional data.

Training and Evaluation: The models are trained on a subset of the Enron Corpus and evaluated using standard metrics such as accuracy, precision, recall and F1 score. Additionally, the analysis considers the models' ability to learn and improve over time, thus emulating real-world email management scenarios.

Thunderbird Plugin Development: Attempt to create a plugin that serves as proof-of-concept for implemented models' integration with the email client Thunderbird. Various limitations of developing such a plugin, including computational constraints, are outlined.

Assumptions and Delimitations : This research assumes that the Enron Corpus is representative of general email communication patterns. It is delimited to text-based email content and does not consider attachments or multimedia elements. The study has focused on English-language emails, with the understanding that future work may need multilingual models.

This approach will address the research questions systematically, comprehensively evaluating machine learning techniques for autonomous email categorization. The methodology tries to balance theory with practical applicability to develop better and more efficient email management tools.

The experimental results in this study eventually show that the approaches implemented are practical and that the transformer-based DistilBERT model can obtain the best accuracy of 89% using confidence thresholds, compared to traditional models such as logistic regression in 75%, random forest in 78% and XGBoost in 82%. Integrating these models into the Thunderbird email client through the E-Classify extension proved successful, the system handled real-time email classification, taking into account computational limitations by using a modular design. The results emphasize broad trade-offs among model complexity, accuracy, and considerations for practical implementation, where Distil-BERT shows higher accuracy but at the cost of greater computational demands, while much simpler models like Logistic Regression provide faster classification at lower accuracy. These findings guide the deployment of machine learning-based email classification systems in real-world settings.

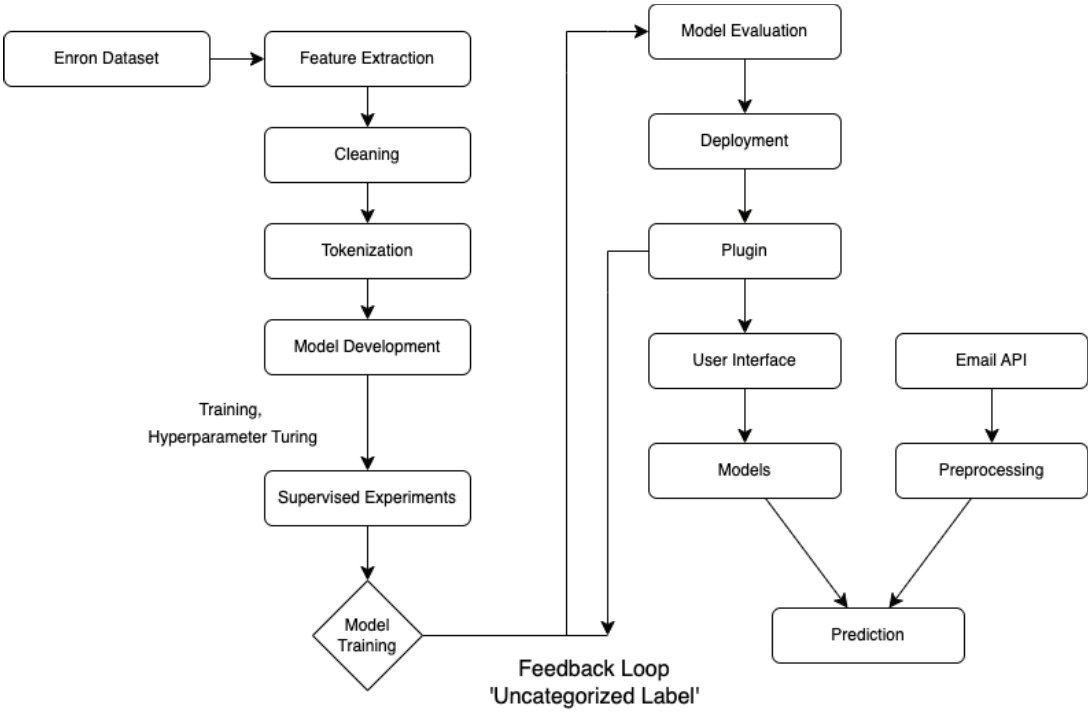

Figure 1.1: Proposed Workflow

2 Background and Related Work

In this chapter, we shall explore the theoretical background and practical applications of some machine learning models applied in our email categorization system. Considering the following models: Logistic Regression, Support Vector Machines (SVM), Random Forest, XGBoost and Distil-BERT. Each model represents a different approach to email classification, from traditional statistical methods to deep learning techniques. We analyze each model's mathematical foundations, benefits, and drawbacks, offering a thorough overview of their functions in classification tasks. This context prepares the ground for understanding our methodology in creating an autonomous email categorization system, underlining how these diverse approaches enhance our solution's overall efficacy and resilience. We will then continue to discuss related studies that compare our method.

2.1 Background

Logistic Regression: Logistic regression is a classification model that estimates class membership probabilities using the logistic function, transforming linear predictions into values between 0 and 1. Its strength lies in simplicity and interpretability, making it an effective baseline classifier.

For binary classification, the logistic function is:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n)}}$$

where $X_1, \ldots, X_n$ are input characteristics and $\beta_0, \ldots, \beta_n$ are model parameters.

In the context of email classification, logistic regression can be used to predict the probability that an email belongs to a certain category (e.g., spam / ham or different topic categories). Genkin et al (2007) [8] demonstrated the effectiveness of logistic regression for large-scale text categorization tasks, highlighting its ability to handle high-dimensional data efficiently.

The advantage of Logistic regression is that it is easy to understand and implement. The coefficients (β values) provide insight into the importance of each feature in the classification

decision. It performs well with large datasets and high-dimensional feature spaces, making it suitable for text classification tasks. This was demonstrated in a research for large-scale text categorization [8]. Logistic regression also provides prediction probability scores, which can help rank or threshold adjustments in email classification tasks.

The drawback of Logistic regression is that it assumes a linear relationship between features and the logarithmic odds of the outcome. This may not always be true in complex email classification scenarios. As a linear model, it may underperform in capturing highly nonlinear decision boundaries in some email classification tasks. Logistic regression can be sensitive to outliers, potentially skewing the decision boundary. The capture of nonlinear relationships may require manual feature engineering or transformation; this can be very time-consuming and generally requires domain experience.

Support Vector Machine (SVM): The SVM algorithm constructs a hyperplane or a set of hyperplanes in a high-dimensional space to separate the data points into different classes. The objective is to find the optimal hyperplane that maximizes the margin between the classes.

The decision function can be represented as:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

Here, x is the feature vector representing the email (for example: bag-of-words, TF-IDF, word embeddings). x_i are the support vectors (training examples closest to the decision boundary). y_i is the class label ($+1$ or -1) of the corresponding support vector. $K(x_i, x)$ is the kernel function that maps the email data to a higher-dimensional space. α_i are the Lagrange multipliers obtained during the SVM training. b is the bias term.

In the context of this thesis, the focus is to find the optimal hyperplane that separates the emails into different classes (for example: spam/ham, or different topic categories) with the maximum margin.

The benefit of SVMs is that they are well-suited for high-dimensional data as they maximize the margin between classes, making them effective in text classification tasks. SVMs can handle linear and nonlinear data using different kernel functions, such as linear, polynomial, or radial basis function (RBF) kernel [10]. This is beneficial in email categorization, where relations between words, features, or categories may not be linear. SVMs have good generalization capabilities and are less prone to overfitting, especially when using appropriate kernel functions and regularization parameters [5].

The disadvantage when training SVMs on large datasets is that it can be computationally intensive, especially for nonlinear kernels, limiting their scalability [11]. SVMs generally provide little insight into the relationship between individual features and the predicted class labels. While they offer accurate predictions, understanding the underlying reasons for classification decisions may be difficult [13].

Random Forest: This is an ensemble learning method that combines multiple decision trees to improve the overall accuracy of the prediction and reduce overfitting. In the context of email classification, the Random Forest algorithm constructs multiple decision trees, where each tree takes the email features (e.g., subject line, sender, body text) as input and outputs a class prediction. The final prediction is made by aggregating the predictions of all the individual trees. For classification tasks, the majority vote among trees is used as the final

prediction, it can be represented as:

$$\text{Predicted Class} = \text{mode}\left(\text{Tree1}(x), \text{Tree2}(x), \ldots, \text{TreeN}(x)\right)$$

Here, N is the number of decision trees in the Random Forest and Tree$i(x)$ is the class prediction of the i-th decision tree for the features of email x.

The advantage of random forests is that they are less sensitive to noise and outliers in the data, as the ensemble approach helps mitigate the impact of individual decision trees [2]. Random forests can effectively handle high-dimensional data, such as text data, without requiring feature selection or dimensionality reduction [9]. The ensemble approach and the random subspace method used in Random Forests help reduce overfitting and improve generalization capabilities [2].

The disadvantage when training random forests is that they can be memory intensive, especially for large data sets or when using multiple trees [14]. Random forests can be biased towards features with more categories or levels, potentially leading to suboptimal performance in specific scenarios [21].

XGBoost: It is a scalable and efficient implementation of the gradient boosting algorithm, an ensemble learning technique combining multiple weak learners (decision trees) to create a strong predictive model.

The function can be expressed as,

$$L(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where: $l(y_i, \hat{y}_i)$ is the loss function that measures the difference between the predicted class $\hat{y}_i$ and the true class y_i for the email. $\Omega(f_k)$ is the regularization term that penalizes the complexity of the individual trees f_k.ϕ represents the parameters of the XGBoost model. Based on this, our goal is to find the optimal set of decision trees that minimize the overall loss and classify emails into the correct categories.

The benefits of XGBoost are that they are highly optimized and can handle large datasets and high-dimensional data efficiently, making them suitable for text classification tasks [4]. XGBoost supports parallel processing, allowing faster model training and prediction (Chen & Guestrin, 2016). XGBoost incorporates various regularization techniques, such as L1 and L2 regularization, to avoid overfitting and improve generalization [4].

One limitation with XGBoost is its several hyperparameters that must be carefully tuned for optimal performance, which can be time-consuming and challenging [4]. While XGBoost includes regularization techniques, it can still overfit the data if the hyperparameters are not tuned correctly or if the model is trained for too many iterations [3]. Like other ensemble methods, XGBoost models can be challenging to interpret and understand, as they combine multiple decision trees [15].

DistilBERT: A distilled version of BERT (Bidirectional Encoder Representations from Transformers), which is a state-of-the-art natural language processing model. It captures deep semantic relationships in text data. DistilBERT uses the transformer architecture with a self-attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q, K, and V are the query, key and value matrices, and d_k is the dimension of the key vectors.

Sanh et al. (2019) [18] introduced DistilBERT, showing that it retains 97 percent of BERT's language understanding capabilities while being 40 percent smaller and 60 percent faster. Sun et al. (2019) [22] further demonstrated the effectiveness of fine-tuning BERT models for text classification tasks, achieving state-of-the-art results on multiple benchmarks.

Distil-BERT offers high performance in capturing complex semantic relationships in text for nuanced email categorization tasks. Long-range dependencies can be handled in text, which helps understand the context of lengthy emails. It requires less computational resources than the full BERT model while retaining 97 percent of BERT's language understanding capabilities, making it more suitable for deployment in resource-constrained environments. Distil-BERT can be fine-tuned on specific tasks with relatively small amounts of labeled data, leveraging its pre-trained language knowledge.

Despite being a smaller model than BERT, Distil-BERT still requires significant computational resources for training and inference compared to traditional models. As a deep learning model, Distil-BERT is prone to overfitting on small datasets, requiring regularization and validation strategies.

2.2 Related work

The Enron Data Set [12] is commonly used in machine learning. It consists of many emails exchanged among Enron employees before the company's collapse. It contains communications covering various topics and domains. Given its extensive collection of emails, it serves as a good resource for training and evaluating models designed to categorize emails into different classes or labels.

According to an experiment on finding the best machine-learning algorithm email for classification by M.Javed (2020), Support Vector Machines (SVM) had the highest accuracy score of 0.98 percentage. However, it had the longest training time. This shows one area where SVM has a better competitive advantage than the others, but it comes at a cost because it takes more time. Considering the nature of the email classification problem, the time factor is important, especially regarding current user needs. Since this implementation will be applied to real-time incoming emails, it is preferable to prioritize efficient computing methods. Random Forest had a accuracy score of 0.77. The author used other methods like Naïve Bayes and k-Nearest Neighbours (KNN), but SVM outperformed them.

The researchers utilized a binary classification model in this approach, assigning a value of 1 to spam emails and 0 to non-spam emails. However, the effectiveness of this method could have been much higher. To improve upon the initially implemented model, more advanced machine learning techniques, including K-Nearest Neighbors (K-NN), Support Vector Machines (SVM), and Artificial Neural Networks (ANN), were implemented, and their respective accuracies were evaluated for spam detection. The classification technique for spam filtering is adopted based on the effectiveness of machine learning and knowledge engineering. Initially, data is collected from the user training set, where spam emails are compared and identified. Subsequently, a global training set is utilized to optimize the classification technique. By employing this approach, the precision rate is enhanced by a minimum of 2 percent.

Under supervised learning techniques, a Support Vector Machine is the most often used.

SVM also performed better than followed by decision trees and Naive Bayes [16]. A review of the commonly used machine learning methods was done to determine which is more suitable for classifying spam emails. A performance comparison was conducted on Spam Assassin spam. The results were unusual as the Naive Bayes classifier performed better than the other methods with 99.46 per cent and SVM, K-NNs, Neural Networks, Artificial Immune Systems and Rough Sets with an accuracy score of 96.9, 96.20, 96.83, 96.23, 97.42 percentages in their respective order [1]. The accuracy scores are very high, and the difference in performance is relatively small. The high accuracy could be attributed to a number of reasons, such as a better understanding of the dataset, adequately cleaned and pre-processed. The other negative reason could be over-fitted data on the training dataset. However, this usually results in a lower score on the test data, which is not seen in this experiment.

Singh et al (2018) [20] discussed solutions and classification processes for spam filtering and a combined classification technique is proposed to obtain better spam filtering. Through data mining, they gathered all the information about previous failures, successes, and current spam filtering issues. Sah et al (2017) [17] introduced a technique to identify malicious spam by utilizing feature selection. They aimed to enhance the training time and accuracy of the malicious spam detection system. The researchers compared two classifiers, Naïve Bayes (NB) and Support Vector Machine (SVM), based on accuracy and computation time. The proposed method consisted of the following steps: text data preparation, word dictionary creation, feature extraction process, and classifier training. The dataset was divided into a training set (702 mails) and a test set (260 mails), with spam and ham mails categorized accordingly. The feature selection process involved generating a feature vector matrix. According to their approach, Naïve Bayes was identified as one of the superior classifiers.

In contrast to the existing studies, this master thesis follows a novel approach for email categorization in Thunderbird by prioritizing autonomous learning without relying on prior user-specific knowledge. The aforementioned experiments, such as the one conducted by M. Javed (2020), highlighted increased accuracy in Support Vector Machines (SVM) but had concerns about their lengthy training times; the approach in this thesis aims to optimize accuracy and efficiency in all models. Unlike Singh et al.'s (2020) emphasis on combining classification techniques for spam filtering, we focus on fine tuning machine learning models to autonomously improve their categorization in a scenario where more data is introduced over time just like in real time emails we receive in our inbox. Furthermore, this thesis differs from Sah et al's (2017) approach to malicious spam detection by customizing the models for the Thunderbird email categorization system needs, covering data preparation, feature extraction, and training/hyperparameter tuning of classifiers. In focusing on the peculiarities of Thunderbird live incoming emails, this thesis also aims to develop a Thunderbird plugin for users with a fast and reliable solution for email management.

3 Dataset Analysis

The dataset used in this research is the Enron email dataset, which is one of the biggest and most popular real-world email corpora. The Enron dataset contains emails on diversified topics. Hence, it is a perfect choice for studying email categorization and other natural language processing tasks.

3.1 Preprocessing

Preprocessing the dataset is a fundamental step in preparing a dataset for machine learning experiments. Noisy, inconsistent, or irrelevant information is present in the raw email data; hence, the preprocessing ensures that the dataset used will be cleaned and standardized and, therefore, can be used in the training of the machine learning model. The following preprocessing steps are carried out:

Removing Years: Since the Enron email corpus contains data from the late 1990s to the early 2000s, entries with invalid timestamps outside this expected range were removed. These anomalous timestamps were likely due to system errors or incorrect date formatting when the data was collected.

Removing Missing Rows: Emails with missing values, especially in the body column, were excluded to maintain data quality. Missing data can introduce biases and reduce the effectiveness of the models.

Text Data Cleaning: The clean column function is designed to clean text data for easy analysis or further processing by making it consistent and usable. It takes string data as input and performs a series of cleaning operations. First, it converts the text to lowercase in order to standardize it. It then removes common prefixes such as "*re:*" and also characters such as hyphens and underscores. It also removes data within square brackets, punctuation marks, newline characters, and numerical values. HTML tags were also pulled out from the text. In addition, it expands common contractions (e.g., "can't" is expanded to "cannot") and removes specific phrases commonly found in email data, such as forwarded email headers. Finally,

it deals with cases where the text indicates a forwarded email by removing the forwarding information. It returns the cleaned text, or '*No Subject*' if the input is *None*.

3.2 Dataset analysis

The dataset contained 517,401 emails from 20,328 unique senders to 58,563 different receivers. This shows an extensive network of communication within the Enron Corporation. A preliminary dataset analysis was conducted to understand its structure and gather information on email communication patterns. Moreover, an analysis of the number of emails sent by each employee was done to study the distribution of email traffic within the organization. The frequency of emails per employee was computed and stored in a DataFrame. The result counts were labelled in a column named "*Count*" and a new column "*Employees*" with the corresponding names of employees were added to it. In order to visualize the distribution properly, a horizontal bar plot is presented with the top 40 employees by email count. The graph clearly shows that "*kaminski-v*" and "*dasovich-j*" are the top email senders, with much higher numbers than their peers. This pinpoints key communicators in the organization.

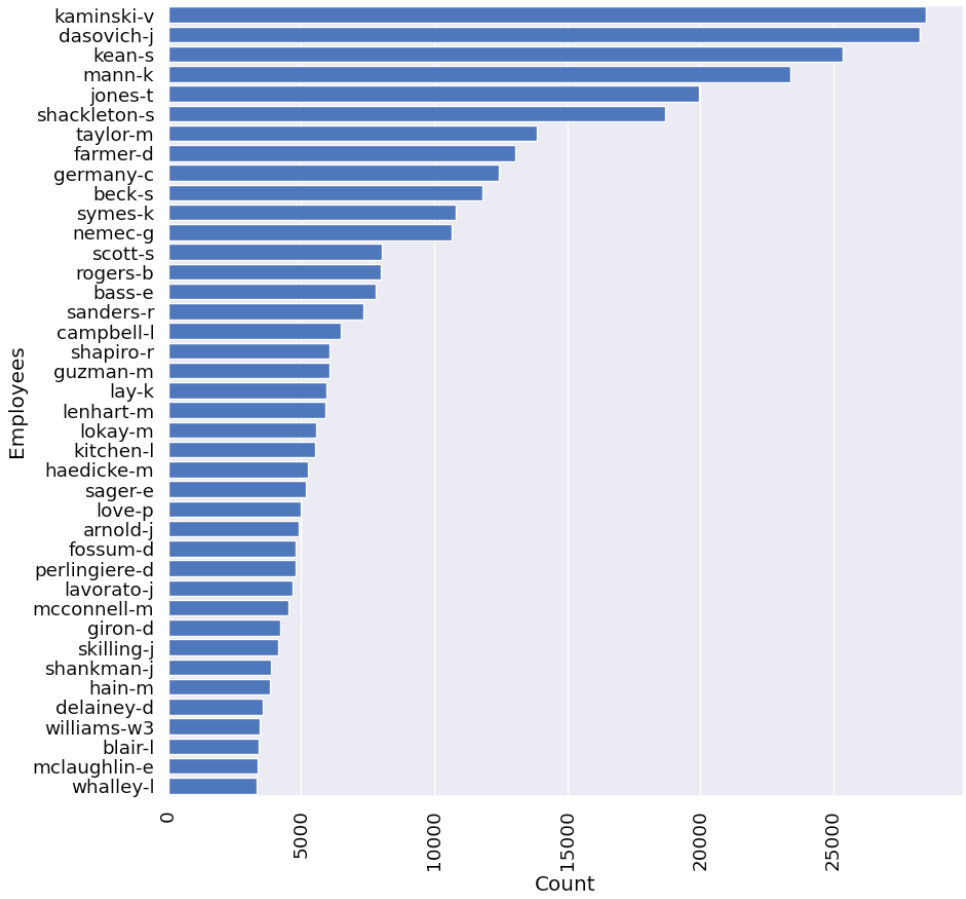

Figure 3.1: The top 40 email senders and the volume of emails sent were identified

Monthly and Yearly Distribution: The following bar chart displays the number of e-mails sent per month. October and November are located at the top in the number of e-mail activities, having more than 60,000 e-mails each. At the other end, e-mail traffic is lowest during

July, with about 25,000 e-mails. A strong increase can be observed in e-mail correspondence from August until November, followed by a little decrease in December. January also shows high email volume, possibly indicating increased communication at the start of the year. The months from February to June exhibit moderate email activity, with some fluctuations.

From 1997 to 1998, there was practically no recorded email activity; in contrast, the period from 1999 to 2000 shows a significant increase in email correspondence, up to about 200,000 emails in 2000. The peak of email communication was in 2001, with more than 250,000 emails. In 2002, there was a significant decrease in email volume. From 2003 through 2004, email activity was recorded as next to nil.

Temporal analysis of email activity in the Enron dataset reveals strong patterns and variations yearly and monthly. Hence, insight into the communication dynamics of this organization is quite important. These may show organizational business cycles because increased use during the autumn could be further connected with the yearly reporting and planning procedures. The distribution over a year demonstrates significant differences in how emails are used, which may reflect overall company development. The months show regular fluctuations that coincide with traditional business cycle patterns and seasonal effects. These visual representations can provide a significant understanding of the organization's interaction patterns. Temporal patterns (like conference emails clustering around specific months) informed my understanding of category distribution, but had less direct impact on model selection. These contextual patterns helped ensure that the categories reflected natural email groupings rather than artificially imposed structures.

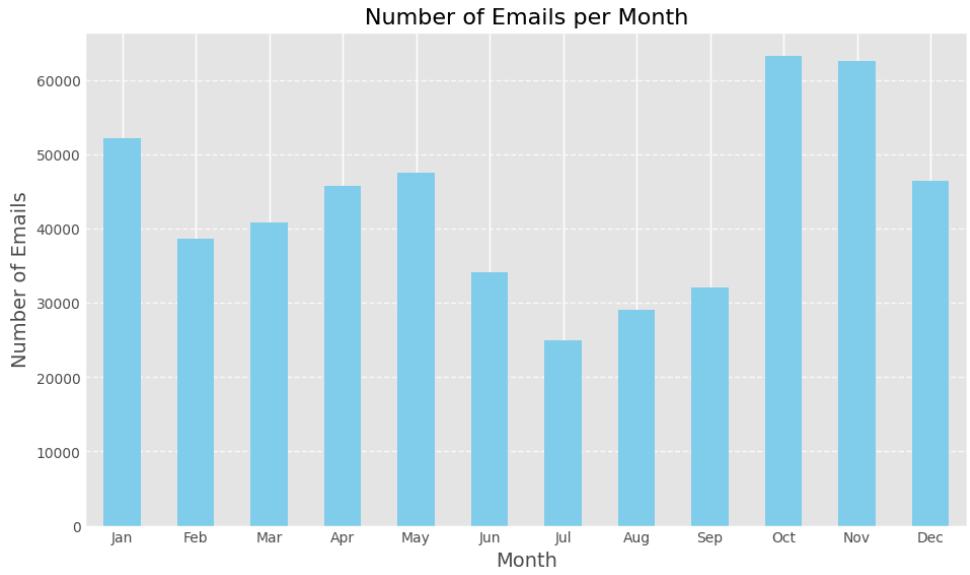

Figure 3.2: Monthly trends were analyzed to identify seasonal patterns in email usage.

4 Method

This chapter comprehensively explores email categorization techniques using supervised machine-learning approaches. The experiments are conducted on the Enron Dataset. In the supervised learning experiments, Logistic Regression, Support Vector Machines (SVM), Random Forest, XGBoost and Distil-BERT models are implemented. These models are trained on labelled data derived from the email folder structure of users in dataset , allowing us to evaluate their performance in classifying emails into predefined categories.

4.1 Supervised Learning Experiments

4.1.1 Data Preprocessing

The preprocessing steps involved several tasks. Key headers including "*Date*", "*Subject*", "*X-Folder*", "*X-From*", and "*X-To*" were extracted from each email to capture essential metadata. The main content was extracted and stored in a dedicated "*Message-Body*" column. Additional columns were derived from existing data, including employee names extracted from file paths. The "*X-Folder*" column underwent normalization to standardize folder names by removing unwanted characters and converting text to lowercase. Non-topical folders such as "*all documents*" and "*deleted items*" were filtered out to maintain focus on relevant categories. Finally, rows containing missing values were removed to ensure data integrity and consistency throughout the dataset.

4.1.2 Class Formation for Supervised Experiments

The process of forming classes for our email categorization task was primarily based on the inherent structure and metadata of the Enron email dataset. The existing email attributes were used to create meaningful and representative experiment categories. The key attributes used in this process were:

Employee Information: Derived from the "*file*" column, identifying the email owner or primary sender.

X-Folder: Representing the folder structure used by employees to organize their emails.

X-From and X-To: Indicating the sender and recipient(s) of each email, respectively. The sender information has also influenced the category selection process. For example, the prevalence of management-related emails from specific senders helped establish the "Management" category.

Subject: Providing additional context and potential thematic information.

By analyzing the distribution of emails across different folders, the following categories were identified. The categories were derived from the folder structure in the Enron dataset, reflecting how users naturally organize emails. Different category types (e.g., intent-based vs. content-based) would likely impact algorithm performance differently.

Intent-based categories (action items vs. information sharing) might require more contextual understanding, potentially favoring transformer models even more strongly, while content-based categories might be more amenable to keyword-based approaches of traditional models. The scope of the research is limited to content-based categories for more reliable ground truth, but this represents an interesting direction for future research.

Email Categories	**Counts**
Management	802
Resumes	547
Projects	379
Universities	367
Personal	281
Energy Capital and Trading (ene ect)	270
Conferences	223

Table 4.1: Email Categories and their Corresponding Counts

4.1.3 Supervised Experiments with a Confidence Threshold

Email Categories	**Counts**
Management	802
California	682
Resumes	547
Projects	379
Universities	367
Personal	281
Heat Wave	244
Conferences	223

Table 4.2: Email Categories and their Corresponding Counts (for experiments with confidence threshold)

In these experiments, two more categories were identified from the dataset: *'California'*, *'Heat Wave'*, in addition to existing ones *'Management'*, *'Personal'*, *'Projects'*, *'Resumes'*,*'Universities'* and *'Conferences'* as seen in table 4.2. These were taken into consideration based on their relevance and their content which can benefit in training our models. In many scenarios, not every email may fit neatly into predefined categories. Rather than creating an explicit *'uncategorized'* category, a probability threshold-based approach was implemented to handle uncertainty in classification. For instance, if the confidence threshold is set (eg.

70 percent) and if the highest probability for any category is above that threshold then that category is assigned to the email. If no category's probability exceeds the threshold the email can be given uncategorized category.

This approach prevents forced classification of uncertain cases, allowing emails to remain unclassified when confidence thresholds are not met, thus preserving the flexibility of adjustable thresholds while allowing for category-specific precision control. The threshold-based classification has some clear advantages. The model learns only actual category patterns, "uncategorized" becomes a post-processing decision, The ability to tune threshold without retraining, More natural handling of uncertainty, where there is a better alignment with real-world email classification needs which is essential for this thesis. These thresholds can be adjusted based on observed performance making our models more adaptable and maintainable.

4.1.4 Training and Feature Generation for Supervised Learning Models

In order to fulfil autonomous email categorization qualities, various algorithms were created and compared. We will discuss the feature generation and training aspects for each model developed.

A similar experimental procedure was followed for each model, dividing the dataset into 80% training and 20% testing sets. Each model was then trained with its appropriate hyperparameter tuning to get better at categorizing with incremental addition of data. The models were used to make predictions on the test set, and their performance was evaluated using accuracy, precision, recall, and F1 score.

For traditional models (Logistic Regression, Support Vector Machine, Random Forest, and XGBoost), a **bag-of-words (BoW)** approach was employed for feature generation. This involved text data preprocessing, extracting the text from columns like *X-From*, *X-To*, *Message-Body*, and *Subject*. These fields were selected as they contain the most relevant information for determining email categories while avoiding redundant or non-informative metadata. The text is tokenized into individual words. Further, regular expressions are used to remove non-alphanumeric characters. The cleaned and tokenized text was further converted into a Bag-of-Words (BoW), where each word is considered a feature. The resulting feature matrix was a sparse representation of word occurrences across different emails.

Logistic Regression: Separate Logistic Regression models were developed for each email folder in the data set. This allowed the creation of tailored models to learn the specific characteristics and trends of emails in each folder more effectively. These models were trained using the logistic regression implementation provided natively in Scikit-learn within Python. It is optimized through GridSearchCV to tune parameters like regularization strength (C), penalty type, and solver algorithms. During training, the models learned to adapt to new email patterns by periodically updating their parameters and associating the attributes of emails, such as the number of words and the presence of specific keywords, with their corresponding folder labels to classify emails better.

Support Vector Machine (SVM): The SVM models were trained similarly to the Logistic Regression models, with specific adjustments to cater to the unique characteristics of SVMs. These adjustments included selecting an appropriate kernel function (linear, polynomial, radial basis function (RBF)) and fine-tuning hyperparameters like the regularization parameter (C) and the kernel coefficient (gamma). The training process involved adjusting these parameters iteratively to identify the optimal decision boundary that maximizes the margin between different email categories, thereby enhancing the model's classification accuracy.

Random Forest: As an ensemble method, Random Forest builds several decision trees, each trained on a random subset of the data and features. Thus, training involves the optimization of primary hyperparameters: the number of trees (n estimators), maximum tree depth, and the minimum number of samples required to exist in the leaf. Herein, this step is Bootstrap aggregation—or just bagging. It also calculated the importance of features to identify the most important features in classification tasks. The final prediction is made by aggregating the predictions of all the trees, typically by majority voting or averaging, which improves the accuracy and robustness of the model.

XGBoost: XGBoost, a very well-known library for efficient and scalable gradient boosting, was further applied in classification performance enhancement. The training process included carefully tuning key hyperparameters, such as the learning rate, number of boosting rounds, maximum tree depth, and minimum child weight. XGBoost's gradient boosting framework iteratively refines the model by focusing on the errors from previous iterations, enabling it to capture complex patterns and interactions within the data.

Distil-BERT: For the DistilBERT model, feature generation differs significantly from the previous models, offering a more contextually aware representation of the email content. The text data (specifically the email column *Message-Body*) is tokenized using the BERT tokenizer, which converts the text into a sequence of CLS(Classification) tokens and SEP(Seperator) tokens BERT can understand. These tokens are fed into the DistilBERT model, which outputs a fixed-length vector (representation) for each input. These vectors serve as the features for the classification task. Using these features, the DistilBERT model explores different learning rates, batch sizes, and training epochs which is then fine-tuned on the email classification task, allowing it to learn the intricate patterns within the text data for accurate classification.

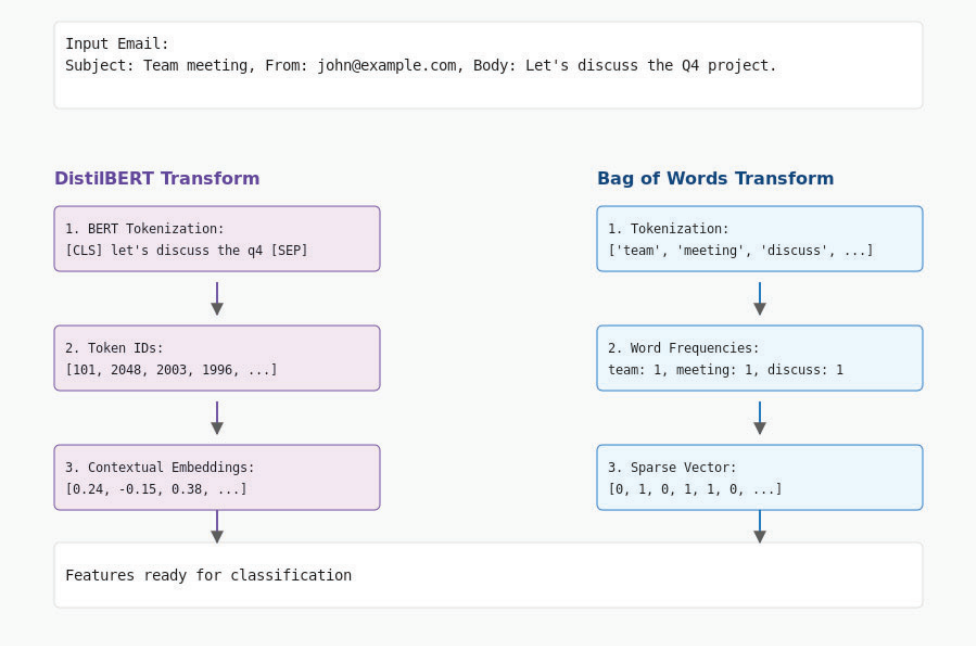

Figure 4.1: Example Transformation for Distil-BERT and Bag-of-words (BoW)

5 Evaluation

The final results acquired from experiments conducted shall be discussed in this chapter.

5.1 Logistic Regression Model Results

5.1.1 Experiments without confidence threshold:

The model performed best in the "*management*" and "*resumes*" categories. There is a notable tendency to misclassify emails as "*management*," particularly from the "*ene ect*" and "*projects*" categories. The "*conferences*" and "*universities*" categories show some mutual misclassification.

The consistency across all four metrics (65) suggests that the model has a balanced performance in terms of precision and recall. An accuracy of 65 percent indicates that while the model performs better than random guessing (which would be about 14 percent for 7 categories), there is significant room for improvement.

The logistic regression model in this experiment showed moderate performance in classifying emails into seven categories. While it performs well for some categories like "*management*" and "*resumes*," there is a tendency to over predict the "*management*" category. The consistent metrics suggest that the model provides a reasonable baseline but could benefit from further refinement or applying more sophisticated algorithms to improve classification accuracy.

5.1.2 Experiments with confidence threshold (0.60 or 60%):

The threshold of 0.60 was chosen as it provides a good balance between being too strict, which can leave too many emails unclassified, and being too lenient. The confusion matrix 5.2 shows reduced misclassifications, particularly for the "*management*" category, in which the previous experiment had many false positives. The increased recall suggests better identification of true positives while maintaining reasonable precision.

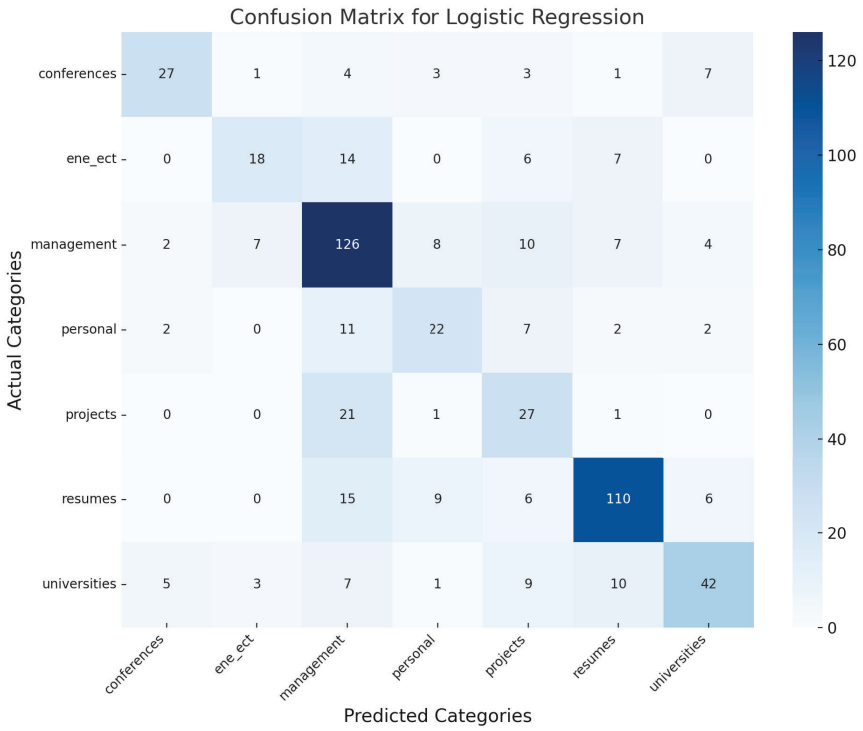

Figure 5.1: Confusion Matrix for Logistic Regression

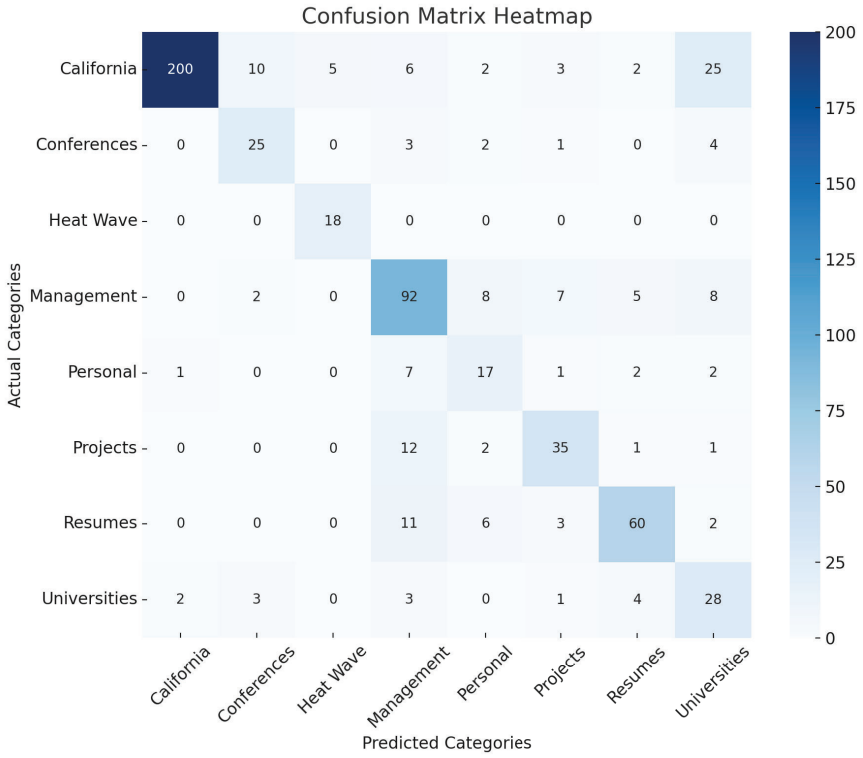

Figure 5.2: Confusion Matrix for Logistic Regression with 60% threshold

Metric	Without Threshold (%)	With Threshold (%)
Accuracy	65	75
Precision	66	71
Recall	65	73
F1 Score	65	71

Table 5.1: Overall Metrics for Logistic Regression

5.2 Support Vector Machines (SVM) Model Results

5.2.1 Experiments without confidence threshold:

The model performs best on the "*management*" category, but tends to overclassify emails into this category. There is a significant issue with the "resumes" category, as the model fails to correctly classify any instances. The "*management*" category shows high recall but lower precision. *"Universities"* and *"conferences"* categories show relatively good performance.

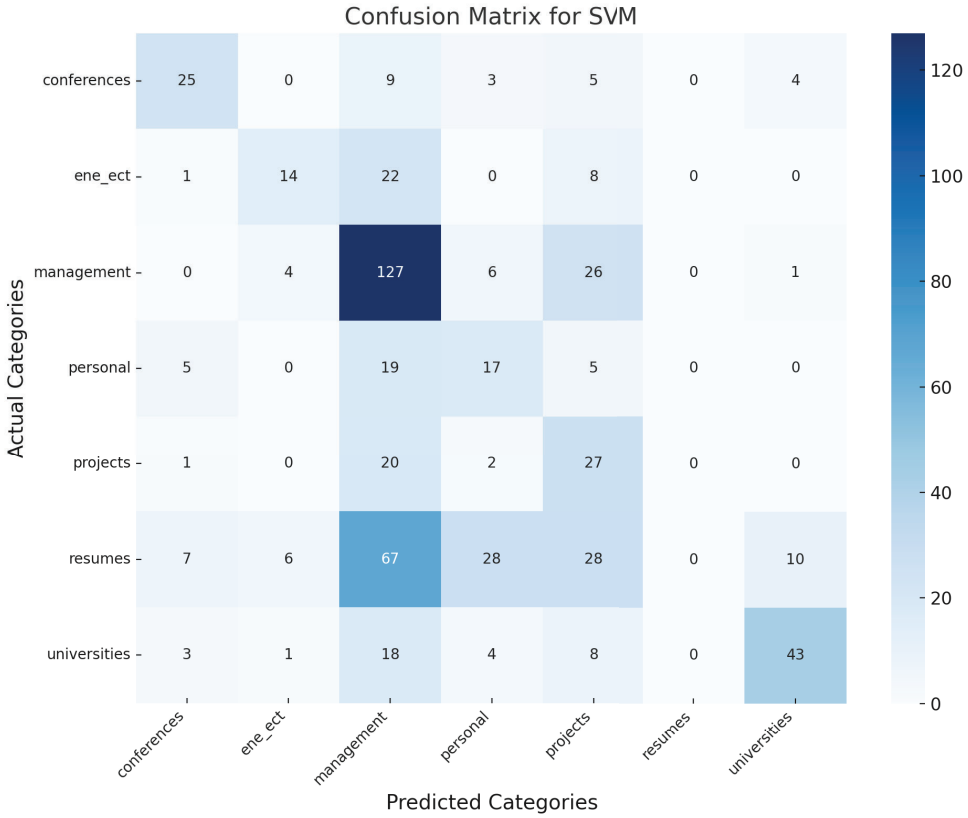

Figure 5.3: Confusion Matrix for SVM

Metric	Percentage
Accuracy	61%
Precision	65%
Recall	61%
F1 Score	61%

Table 5.2: Metrics for SVM Model

The accuracy of 61 percent is slightly lower than the logistic regression model. Precision (65 %) is slightly higher than the other metrics, suggesting that when the model makes a positive prediction, it's slightly more likely to be correct. Recall and F1 Score are consistent with the accuracy, indicating a balanced performance. The lower overall metrics compared to logistic regression suggest that the SVM model might be struggling more with this particular classification task. The SVM model performs below average in this experiment with some significant issues. It showed a strong bias toward the "management" category and failed to classify "resumes." The consistent metrics around 61% would suggest that the model, while doing better than random guessing, still has substantial room for improvement in further stages.

5.2.2 Experiments with confidence threshold:

SVM model had taken a lot of time to train for this experiment to return any predictions for a long time. The training of SVMs becomes significantly slower when adding classes and their data samples due to the model's computational complexity and the increased dataset size. SVMs operate by finding an optimal hyperplane to separate classes, which involves solving a quadratic optimization problem. When data samples are combined, the feature space grows, and the computation of kernel functions, especially in non-linear SVMs, becomes more resource-intensive, scaling quadratically with the number of samples. Additionally, identifying the decision boundary requires more time and computational effort if the classes are imbalanced or have overlapping features. These factors might contribute to the longer evaluation times observed.

Due to this computational burden, it was not successful to extract the final results for this particular experiment. Training SVM for experiments without thresholds also took an intensive time to process, and the final prediction was not satisfactory (61 %). This can be connected to what M.Javed (2020) mentioned about training times. This computational constraint has prioritized focused development efforts on more efficient models to deliver better performance within reasonable resource bounds. Although SVM is a theoretically interesting approach for email classification, practical limitations suggest that it may not be the best choice when designing a real-time email classification system.

5.3 Random Forest Model Results

5.3.1 Experiments without confidence threshold:

The model performs best on the "*management*" and "*resumes*" categories. There is still a tendency to misclassify emails as "*management*," particularly from the "*ene ect*" and "*projects*" categories, but less pronounced than in previous models. The "*conferences*" and "*universities*" categories show improved classification in this model. Overall, the Random Forest model seems to have reduced misclassifications across most categories in these expermients.

The consistency across all four metrics suggests that the model has a very balanced performance in terms of precision and recall. An accuracy of 70% indicates a significant improvement over the previous models (Logistic Regression and SVM).

In this experiment, the Random Forest model considerably improved the classification of emails into the seven categories with reduced misclassifications. The increased metrics to approximately 70% suggest the model can provide a robust and accurate classification. Because of the ensemble approach, the Random Forest algorithm combines multiple decision trees and is good at capturing complex patterns in email data. The improvement can be attributed to the model's ability to handle non-linear relationships and its robustness to overfitting. While there is still room for improvement, especially in classes like "*ene*

Figure 5.4: Random Forest Confusion Matrix

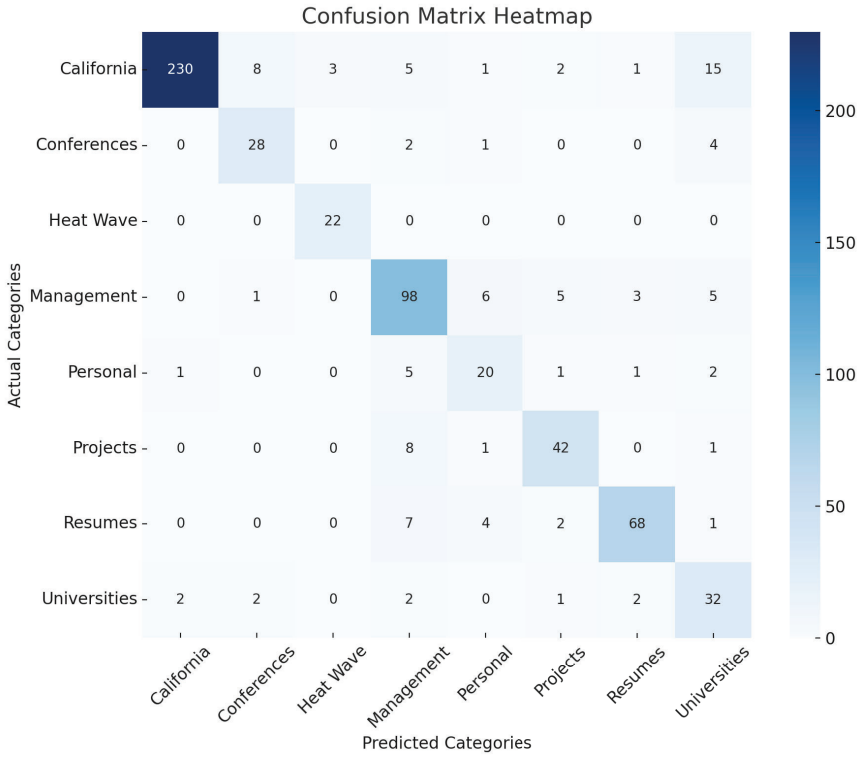

Figure 5.5: Random Forest Confusion Matrix with 55% threshold

ect" and "*personal*," the Random Forest model showed a significant gain in classification effectiveness for the email classification task.

5.3.2 Experiments with confidence threshold (0.55 or 55%):

This threshold setting was effective because random forest naturally produce well-calibrated probability estimates. The model's ensemble nature suggests reliable, high-confidence predictions. The confusion matrix shows better separation between categories, especially for problematic classes like "*Projects*" and "*Personal*" in previous experiment.

Metric	Without Threshold (%)	With Threshold (%)
Accuracy	70	78
Precision	70	77
Recall	70	75
F1 Score	69	77

Table 5.3: Metrics Comparison for Random Forest Model

5.4 XGBoost Model Results

5.4.1 Experiments without confidence threshold:

In these experiments, The model performs best on the "*management*" and "*resumes*" categories, similar to previous models. There is still a tendency to misclassify emails as "*management*," particularly from the "*ene ect*" and "*projects*" categories. The "universities" category shows improved classification compared to some previous models. The "*projects*" category seems to be challenging for this model, with more misclassifications than correct classifications.

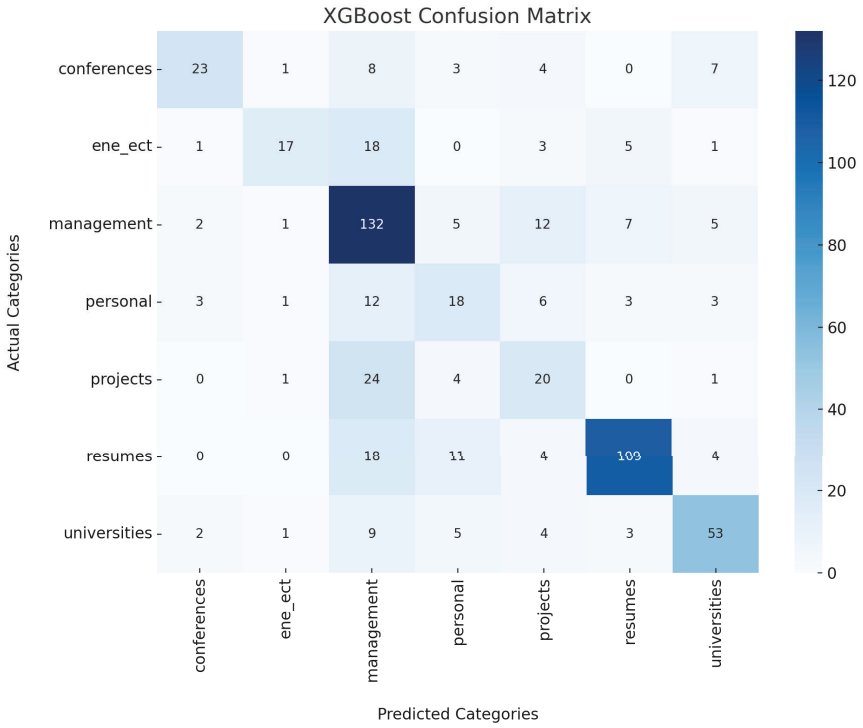

Figure 5.6: XGBoost Confusion Matrix

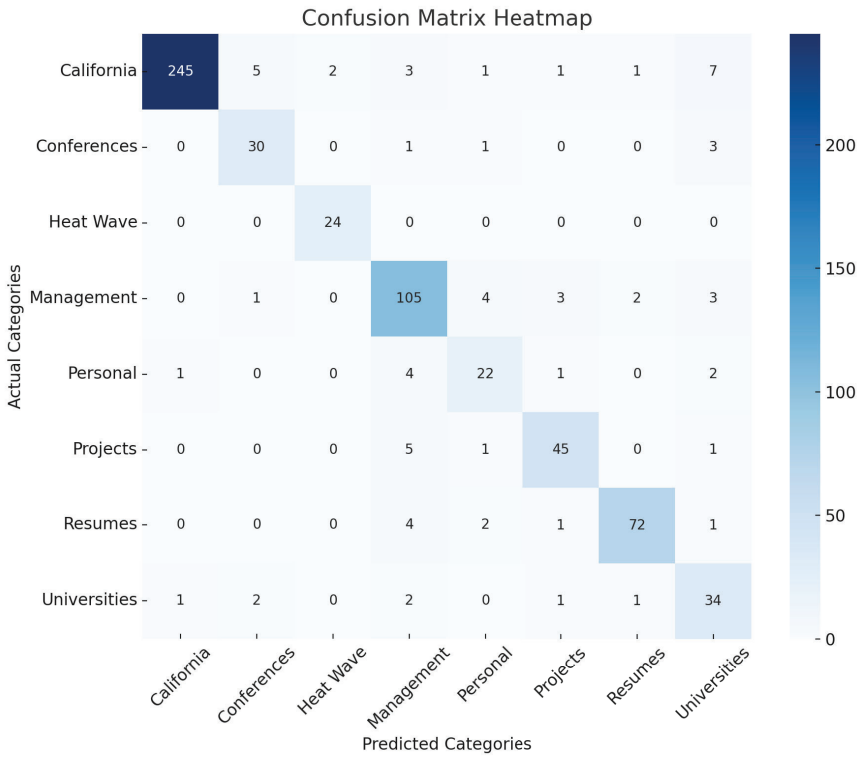

Figure 5.7: XGBoost Confusion Matrix with 50% threshold

An accuracy of 67% shows an improvement, but it's slightly lower than the Random Forest model (which had 70% accuracy).

The XGBoost model used in these experiments shows admirable performance in classifying emails into seven different categories, demonstrating improvements compared to previous model experiments; however, it slightly underperforms compared to the Random Forest model. Although it does well with most classes, it struggles with specific misclassifications, particularly for the "*management*" class. As implemented in XGBoost, Gradient boosting effectively uncovers complicated patterns in the email data set to outperform the simpler models. However, the lingering misclassification of some types, such as classifying "*projects*" into "*management*, gives evidence that there could be overlapping features or content among the classes that the model fails to distinguish.

5.4.2 Experiments with confidence threshold: (0.50 or 50%)

For XGBoost, setting threshold of (0.50 or 50%) was observed to be beneficial as this model provides better-calibrated probabilities. Also, the gradient boosting offered in this model improves prediction confidence.

Metric	Without Threshold %	With Threshold %
Accuracy	67	82
Precision	65	85
Recall	67	83
F1 Score	67	82

Table 5.4: Overall Metrics for XGBoost Model

5.5 Distil-BERT Model Results

5.5.1 Experiments without confidence threshold:

The model shows improved performance across all categories compared to previous models in these experiments. Misclassifications are generally reduced, with fewer instances being incorrectly categorized as "*management*". The "*projects*" category, which was problematic in previous models, shows significant improvement. The model maintains strong performance in categories like "*resumes*" and "*management*".

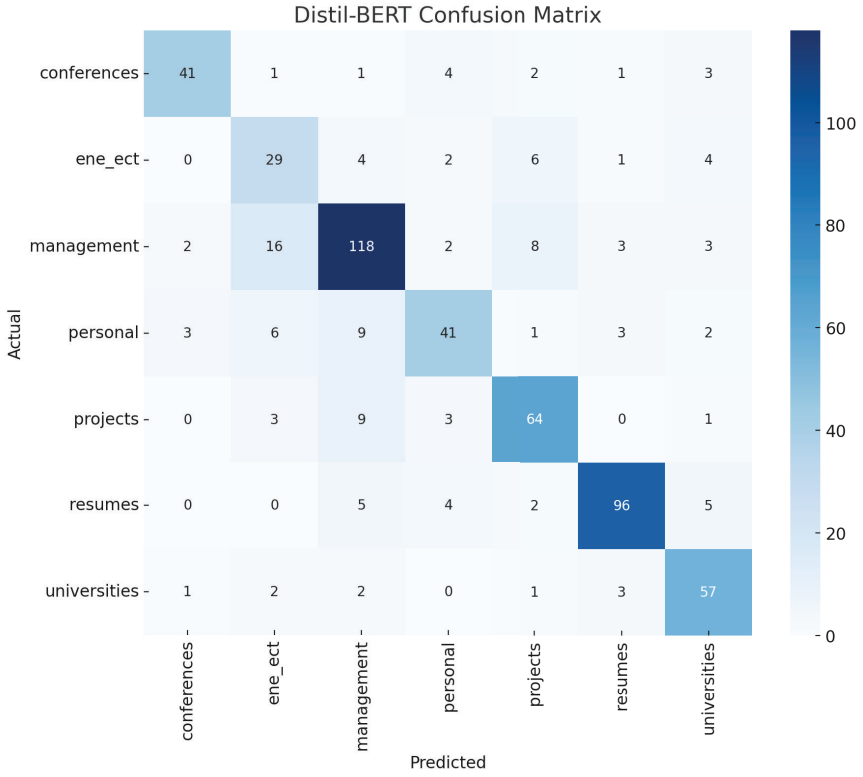

Figure 5.8: Distil-BERT confusion matrix

An accuracy of 78% shows a significant improvement over all previous models (Logistic Regression, SVM, Random Forest, and XGBoost).This performance represents a substantial step up in classification accuracy compared to the traditional machine learning models.

The Distil-BERT model in this experiment demonstrates superior performance in classifying emails into the seven categories, showing marked improvement over all previously discussed models. The confusion matrix reveals better handling of all categories, significantly reducing misclassifications.

Distil-BERT uses a transformer structure to know how to deal with complicated relationships in text, and thus, it is perfect for sorting emails. That way, it learns to understand subtle language and context, likely boosting performance, mainly in classes where regular machine learning approaches face difficulties.

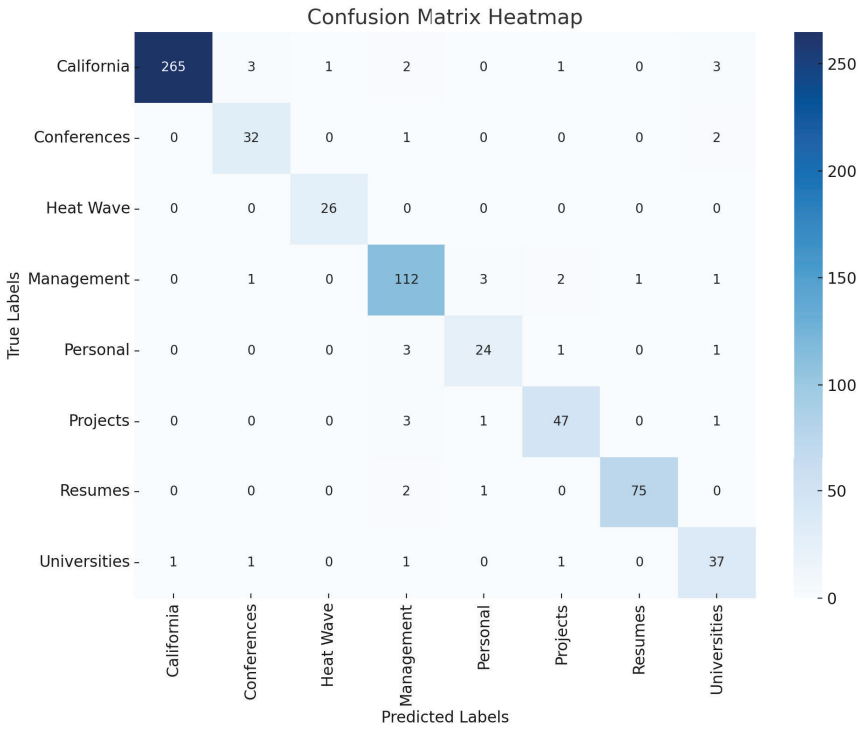

Figure 5.9: Distil-BERT confusion matrix with 45% threshold

5.5.2 Experiments with confidence threshold (0.45 or 45%):

Transformer models like Distil-BERT produce more nuanced probability distributions. The contextual understanding of the model allows for more reliable predictions. The confusion matrix 5.9 shows excellent separation between categories with minimal misclassifications.

Metric	Without Threshold	With Threshold
Accuracy	78	89
Precision	78	90
Recall	78	87
F1 Score	78	88

Table 5.5: Overall Metrics for Distil-BERT Model

5.5.3 Reflection from Experiments

The initial experiments without threshold showed clear evidence of class imbalance affecting predictions, particularly with the "*management*" category being overrepresented and leading to over prediction. Despite its high representation, the "*California*" category does not show the same over prediction issues that "*Management*" had in earlier experiments. The models were likely still biased towards classes with more training examples, though the confidence thresholds helped improve overall accuracy and reduce misclassifications.

This research has also experimented with varied thresholds for different categories, it was found that this approach introduced additional complexity without providing a clear advantage. Setting different thresholds per category introduced new challenges, such as creating artificial boundaries between classes, difficulty determining optimal threshold values for

each category, and creating new types of bias in the prediction process. This could be addressed with future suggestions like automated methods of identifying and dealing with class imbalance using synthetic data generation or advanced sampling techniques, dynamic threshold adjustment based on user feedback, multi-level confidence handling, better model calibration techniques, and hybrid approaches combining confidence scores with other metrics.

Distil-BERT performed better with imbalanced data than traditional models, suggesting that model architecture and training approach may be more influential than post-processing techniques like category-specific thresholds.

In another case, if we balance the classes to contain a similar number of samples from the dataset, then no single class is overrepresented. This adjustment can enable the model to treat all classes fairly, which would resulting in a uniform and balanced performance across all metrics. However, models handling imbalanced data are still beneficial for this research because the imbalance in email categories reflects real-world email distributions. In practical email usage, some categories naturally occur more frequently than others, and balanced categories are rare. For example, "*management*" emails being more numerous than "*conferences*" is likely representative of typical business email patterns. Training models on artificially balanced data might hurt real-world performance by creating unrealistic expectations. So, this is a trade-off between quality and quantity. Having accurate predictions can be better than forcing classifications for all emails.

Model	**Metric**	**Without Threshold (%)**	**With Threshold (%)**
Logistic Regression	Accuracy	65	75
	Precision	66	71
	Recall	65	73
	F1 Score	65	71
SVM	Accuracy	61	N/A
	Precision	65	N/A
	Recall	61	N/A
	F1 Score	61	N/A
Random Forest	Accuracy	70	78
	Precision	70	77
	Recall	70	75
	F1 Score	69	77
XGBoost	Accuracy	67	82
	Precision	65	85
	Recall	67	83
	F1 Score	67	82
Distil-BERT	Accuracy	78	89
	Precision	78	90
	Recall	78	87
	F1 Score	78	88

Table 5.6: Performance Metrics for all models with and without Confidence Threshold Setting

6 Integration with Thunderbird Email Client

Integrating machine learning features into an email client involves unique challenges that require technical implementation and consideration of user experience and system resources. This chapter details a system's architectural design and implementation, incorporating email classification for the Thunderbird email client through a modular, extensible approach. The system contains two main components: a custom Thunderbird plugin (EClassify 6.1)and a FastAPI server[1]. This integration seeks to improve email management and user productivity. This implementation also aims to adopt an architecture that emphasizes modularity and the separation of concerns, where every module will operate independently and have efficient communication channels.

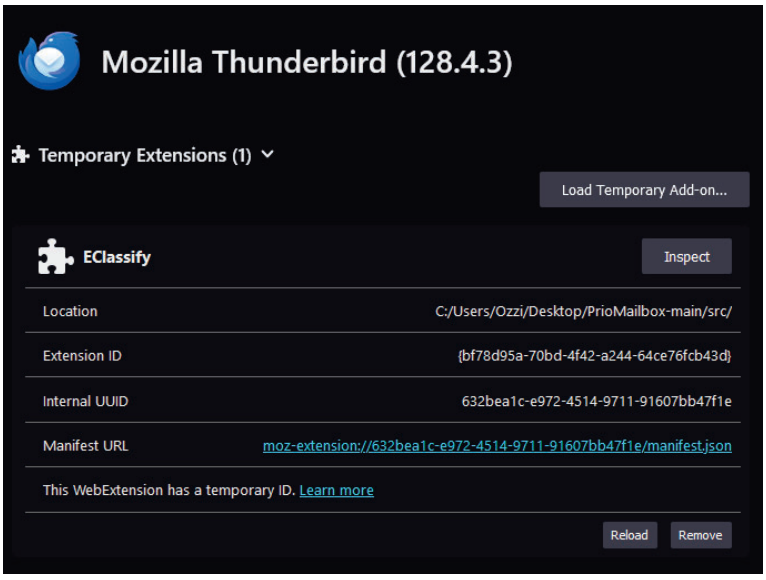

Figure 6.1: EClassify - Implemented Thunderbird Extension

[1] https://fastapi.tiangolo.com/

6.1 System Architecture

The implementation architecture integrates machine learning-based email classification models into the Thunderbird email client through a dual-component system. Before Classification, for the email filtering process, the plugin uses criteria-based selection mechanisms which allow users to narrow down the scope of Classification based on various parameters such as message status (read/unread), sender information, subject line content, in order to optimize the classification process for relevant subsets of emails. Once the appropriate emails are identified, the data preparation phase begins, where the plugin systematically processes each email to extract pertinent information while removing superfluous elements - this includes parsing email headers to retain only classification-relevant metadata, cleaning the message body to remove formatting artifacts, standardize text content, and structuring the extracted data into a consistent JSON format aligns with the FastAPI service's expected input specifications. The backend system uses FastAPI, managing multiple machine-learning models for email classification. FastAPI provides performance, ease of use, and asynchronous capabilities, which help handle multiple requests from the Thunderbird plugin. The frontend system handles user interactions within Thunderbird with a user interface integrated within Thunderbird, allowing users to interact with the classification system. This interface provides options for triggering email classification and viewing the categorization results.

Our fine-tuned ML models are loaded into the FastAPI application. The plugin establishes a connection with a FastAPI server, facilitating the transmission of email data for classification and the reception of classification results. This integration forms the core of the system's functionality, linking the user-facing component with the backend classification models. The model will be exposed as an API endpoint and will be ready to receive email data for classification. The Thunderbird plugin uses web technologies like JavaScript and HTML/CSS. The plugin interfaces with Thunderbird's core functionality through Mozilla's WebExtension API[2], which provides methods to access emails, folders, and other pertinent data within the Thunderbird client. When a user initiates classification by selecting individual emails or processing multiple messages in batch, the plugin's API request handler packages the email content. It transmits it to the designated FastAPI endpoint. The back-end service then performsms the classification process, using the deployed machine learning model to analyze the email content and generate categorical predictions. After receiving these classification results, the plugin updates Thunderbird's interface, applying appropriate tags to the emails that include the predicted category it understands. The entire process is depicted in the Figure 6.2.

6.2 Model Loading and Feature Processing

Preprocessing functions are implemented in the FastAPI code to handle incoming email text as seen in Listing 1. These functions prepare the email content by tokenizing the text, removing unwanted patterns, and converting the tokens into a format suitable for machine learning models. For classical models (Logistic Regression, Random Forest, XGBoost), a bag-of-words representation is used to convert the email text into numerical feature vectors. This transformation is crucial for making predictions, as machine learning models require numerical input. The Distil-BERT model uses tokenization and embedding techniques specific to transformers. The *AutoTokenizer* is used to tokenize the email text, and the *AutoModelForSequenceClassification* model generates predictions based on the encoded tokens.

[2] `https://developer.thunderbird.net/add-ons/about-add-ons`

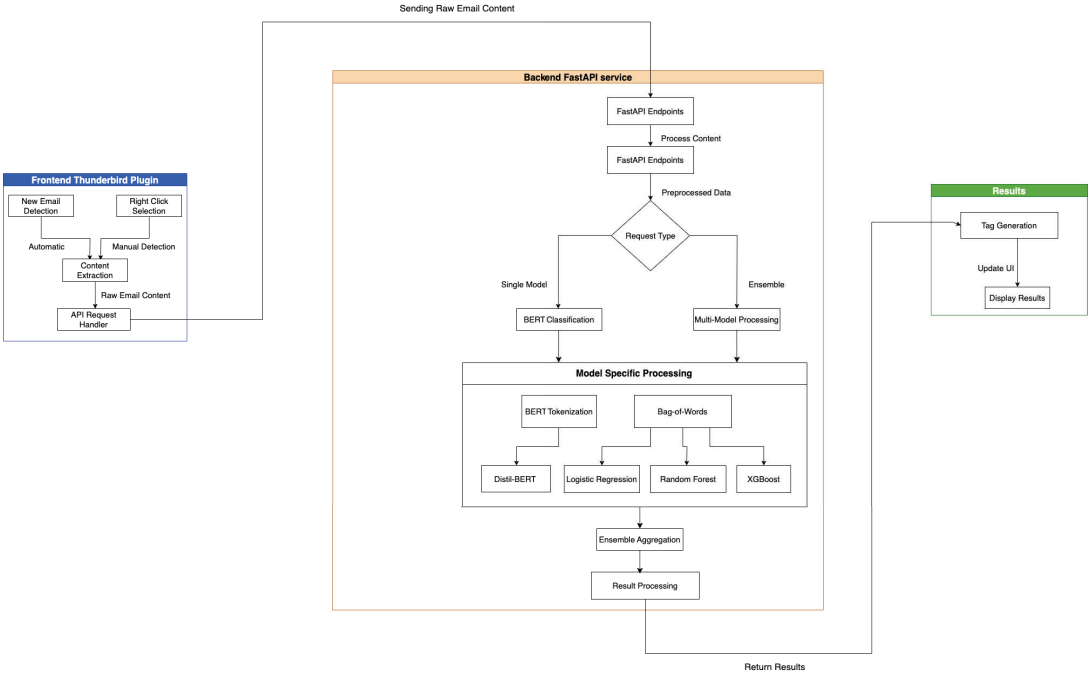

Figure 6.2: Email Processing System Architecture

6.3 Front-end Implementation

The front-end part of the Thunderbird plugin acts as a link between the email client and the FastAPI backend service, implemented in a background.js file. It handles exceptions, such as server unavailability or data processing errors. This file contains logic for crucial tasks such as:

6.3.1 Continuous Monitoring and Classification

The system implements a continuous observer pattern that constantly monitors the Thunderbird inbox as seen in Listing 2. The background.js component automatically extracts and processes the content of the email when you receive new emails, including the subject, body, and metadata. This content is pre-processed and formatted into a structured payload compatible with the back-end FastAPI service. The implementation utilizes AJAX requests to enable efficient data transmission to the server.

6.3.2 Dynamic Tagging and Organization

When classification results are received from the FastAPI service, background.js file processes the predictions and integrates them into Thunderbird's interface as seen in Listing 3. Based on the machine learning model's outputs, the system tags emails with predicted categories such as "Management," "Personal," that it recognizes. The creation of new tags also adds specific colours to the tags. The script also dynamically manages tags to ensure all predicted categories are represented in Thunderbird's tagging system.

6.3.3 User Interaction and Bulk Operations

This system provides users interactive capabilities for manual classification control and automatic functionality. Users can initiate classification for individual emails or apply bulk

classification across many selected emails using a right-click context menu as shown in Listing 4. This can be useful in adjusting email organization to personal preference or overriding model misclassifications.

6.4 Backend Implementation (FastAPI)

The FastAPI backend implements an API framework that handles individual and ensemble model predictions. This component is responsible for receiving requests from the front end (via background.js), processing the data, and returning predictions generated by the various machine learning models. Each model is loaded from pre-trained files saved on the server and used for prediction when a request is received. The FastAPI server manages CORS (Cross-Origin Resource Sharing) policies [3] to allow requests from the front end, ensuring communication across different environments.

The backend server exposes several API endpoints for the frontend to interact with, most notably:

1. **/classifyemail:** This endpoint, in Listing 5 receives raw email content and processes it using a BERT model for classification. The email text is tokenized and fed into the pre-trained BERT model, which returns a predicted category.
2. **/classifyemailensemble:** As shown in Listing 6, This endpoint performs an ensemble classification approach using multiple models (Logistic Regression, Random Forest, XGBoost, and Distil-BERT). The prediction of each model is returned along with its associated probability, and the class with the highest probability is selected as the final prediction,
3. **/test:** This simple endpoint is included to verify the API's functionality, returning a success message to confirm that the backend is up and running.

6.5 Model Loading and Prediction Speeds

The table 6.2 shows the differences in model loading times on two different hardware setups: Kaggle's environment with an Intel Xeon CPU, a P100 GPU, and 32GB of RAM versus a laptop with an Intel i7 7th generation CPU, 8GB of RAM, and no GPU. In Kaggle's environment, all models load significantly faster because of the extra power of the high-performance CPU and GPU. For example, DistilBERT is loaded in a mere 0.3 seconds, while it takes almost 34 seconds on the laptop. This is a considerable time gap because the laptop does not have a GPU, which is necessary to speed up deep learning models like DistilBERT. In contrast, Kaggle's system GPU significantly speeds up computation times, especially for more complex models like XGBoost and DistilBERT.

The table 6.1 and the graph 6.3 show the categorization of the prediction speeds of different models based on ranges of input text lengths so that their performance can easily be compared on the Kaggle server. The Distil-BERT model shows consistently fast prediction times across all input lengths, averaging 0.03 seconds for texts up to 200 tokens and slightly rising to 0.07 seconds for longer texts falling within the 200-500 tokens range. In contrast, the classic machine learning models, Logistic Regression and Random Forest, are much slower in their prediction times: on average, about 6.7 and 6.0 seconds, respectively, for the different length ranges. Remarkably, the XGBoost model is slow in making predictions, averaging over 34 seconds across all text ranges a computationally intensive model. Categorized by model type, this again highlights the efficiency of transformer models in particular

[3] `https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS`

Model	Email Length Range	Avg Time (s)
Distil-BERT	0–50	0.03
	50–100	0.03
	100–200	0.03
	200–500	0.07
Logistic Regression	0–50	6.71
	50–100	6.72
	100–200	6.73
	200–500	6.72
Random Forest	0–50	6.00
	50–100	5.97
	100–200	5.96
	200–500	5.99
XGBoost	0–50	34.13
	50–100	34.06
	100–200	34.15
	200–500	34.29

Table 6.1: Prediction speeds across different email length ranges

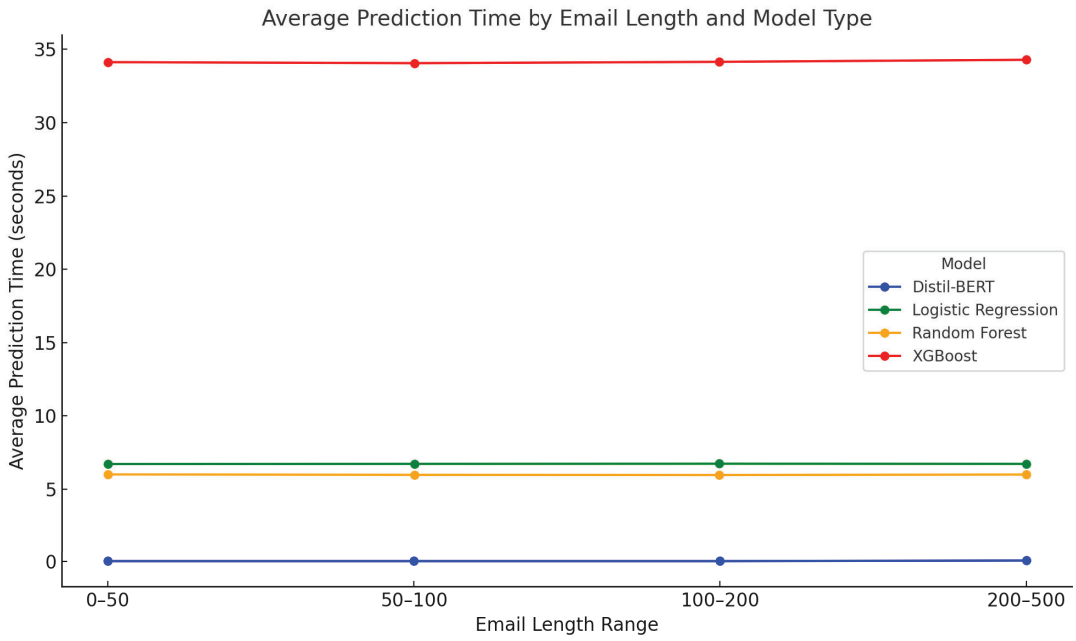

Figure 6.3: Graphical representation of recorded prediction times

to quickly make predictions. Such findings are striking when benchmarked against traditional and ensemble models.

6.6 Classification with Live Emails

After running the tests using live email samples, the models demonstrated accurate predictions. The implementation has been updated, so it also handles uncertainty by assigning an

System	Specifications	Model Name	Loading Speed (s)
Kaggle	CPU: Intel Xeon, 32GB RAM GPU: P100, 16GB RAM	Logistic Regression Random Forest XGBoost Distil BERT	0.056185 0.317743 0.946873 0.299794
Laptop	CPU: Intel i7 7th Gen, 8GB RAM GPU: None	Logistic Regression Random Forest XGBoost Distil BERT	0.25 0.67 2.08 33.88

Table 6.2: Model Loading Speeds across systems

"*uncategorized*" tag to emails when the model's prediction confidence is low or when the content significantly diverges from the derived training categories or when the content includes videos or images. Although this might seem like a shortcoming in the current system's ability to track broad classification, it serves a vital function in the model's development. Tracking "*uncategorized*" labeled emails creates a valuable feedback loop. Focusing on *uncategorized* emails, we can form a new data set to identify emerging categories and precisely understand where current models fall short. It can serve as additional training data for future model updates and validate if certain thresholds need adjustment. In this way, the system's reliability is maintained by avoiding forced classifications while simultaneously creating opportunities for continuous model improvement through future retraining or fine-tuning. Thus, this method can become instrumental in continuing the development and adaptation of the model to new email patterns and categories. Figure 6.4, 6.5,6.6 show the working and results from plugin.

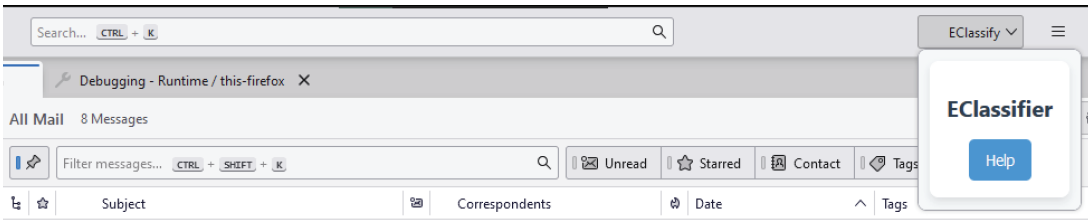

Figure 6.4: Extension activated in Thunderbird UI

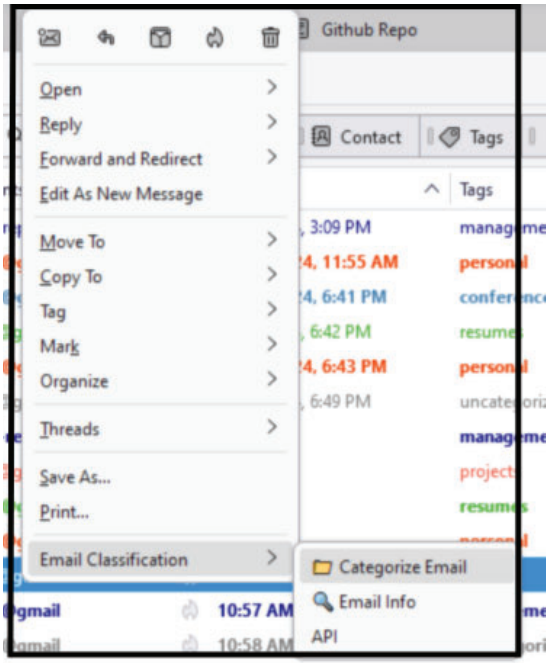

Figure 6.5: Option for user to right click and classify a selected email

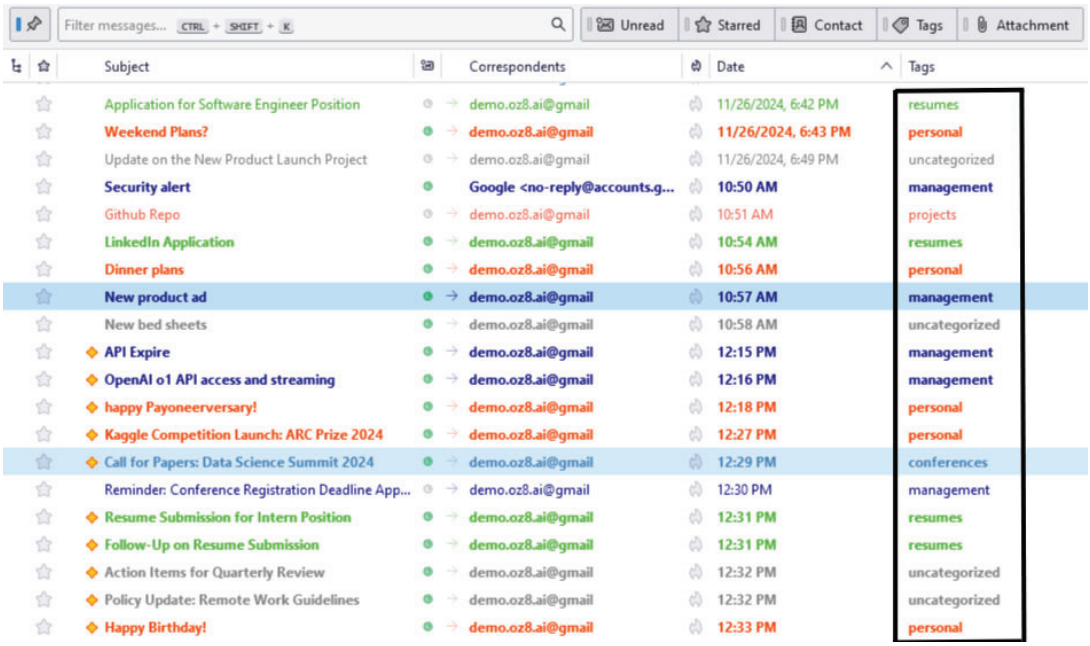

Figure 6.6: Displaying predictions on Thunderbird UI

7 Discussion

Handling new labels: Traditional models, such as logistic regression, must be retrained whenever new labels are added, as they depend on predefined categories. On the other hand, Distil-BERT can add new labels to the model by fine-tuning its parameters on the labeled data for the new categories, so that it will be able to recognize and classify these labels. For traditional models, incremental learning approaches could help mitigate the full retraining problem. Techniques like stochastic gradient descent updates for logistic regression or online random forests could allow partial model updates with new label data. Another approach would be to use transfer learning techniques to initialize new category classifiers with existing knowledge. These methods would not match Distil-BERT's fine-tuning capabilities but would reduce the computational burden of adaptation significantly. This is worth exploring in future work.

Models performance on new labels: Initially, models struggle with new labels if they haven't been trained on similar data. Without fine-tuning, performance on these labels can be poor. After fine-tuning with sufficient labelled data, models, particularly transformer-based ones, can achieve good performance, though this is highly dependent on the quality and quantity of the new data. This can make us question how many examples of new labels are needed to get good performance. The number of examples needed varies by model type and task complexity. We need hundreds or even thousands of examples for each new label for traditional models. Due to their pre-trained language understanding, transformer models can require fewer examples, sometimes only a few dozen to a few hundred. However, more data generally leads to better performance, especially for complex or highly nuanced labels. This also varies between different new labels as the number of examples required can vary based on how similar the new labels are to existing ones. Fewer examples may be needed if the new labels are conceptually close to those the model has already seen. Conversely, if the new labels are very different or represent a new domain, more examples will be required to perform well.

Measuring the dissimilarity to labels: Dissimilarity can be measured using cosine similarity which compares label feature vectors to assess their similarity or difference. Embedding-based Distance using pre-trained embeddings (eg. from BERT) can also be used to calculate the distance between the new and existing labels in a high-dimensional space. An unsuper-

vised learning method called cluster analysis also groups similar labels and measure how far new labels are from these clusters, indicating their dissimilarity.

7.1 Challenges

The initial plan was to configure the environment to fetch live Thunderbird emails using a custom API, list the user, and print the message body and the email. However, some "cookie policies" prevented me from doing so. This particular issue was time-consuming, and a Thunderbird support executive suggested the solution of using a web extension API, Mozilla, to solve it. It was successful after testing and set a path for successful integration.

The main challenges in implementing machine learning models for email categorization in Thunderbird could be related to both performance and user experience. The most significant hardware constraint is probably that the majority of users probably do not have access to GPUs or high-performance CPUs. While highly accurate, transformer models, such as Distil-BERT, are very computationally expensive and may have slow inference times on resource-constrained devices, which could lead to delays in email classification. This total latency affects the usability in real time, frustrating users who want immediate results. More extensive models also require more memory and storage, which can strain systems with limited resources and further impact performance. When deploying the models, The traditional models (Logistic Regression, Random Forest) required simpler deployment pipelines with fewer dependencies, while transformer model like Distil-BERT required managing more complex dependencies and had larger memory requirements. Beyond raw performance metrics, user experience differences between models include prediction confidence visualization (how uncertainty is communicated to users), prediction latency (noticeable waiting time for classifications), false positive tolerance (some categories may be more problematic for misclassification from a user perspective) and adaptability to user feedback. In this research, I mainly focused on performance metrics and latency, but future work could incorporate deeper qualitative user studies that evaluated these additional dimensions of user experience.

Another challenge is integration complexity: orchestrating the APIs of Thunderbird with a back-end system running on FastAPI does not come trivially. Firm error handling and detailed implementation are required to make sure that email data extracted by the Thunderbird plugin is correctly preprocessed, formatted, and sent to the backend for classification. This also brings more complex models, such as PyTorch or TensorFlow, increasing the complexity in deployment and potentially making the plugin harder to maintain or update. Besides, the classification model itself may be updated frequently, which can also lead to a re-deployment of the backend and thus affect the plugin's functionality and user experience. Given the preceding challenges, the decision for a machine-learning model and technique for Thunderbird depends on the trade-offs between a few key characteristics: accuracy, model size, query speed, code complexity, and ease of deployment. In this respect, Distil-BERT would be a very strong contender where high accuracy is one of the top priorities, and there are abundant hardware resources. Its ability to understand nuanced email content makes it incredibly effective when fine-tuned on datasets.

The deployment difficulty also varied significantly among models: Logistic Regression and Random Forest were straightforward to deploy, requiring minimal dependencies and computational resources. XGBoost added complexity with additional library requirements and larger model files. Distil-BERT presented the more deployment challenges, GPU optimization considerations, and more memory management. These differences challenged the design of my modular plugin architecture to accommodate varying hardware capabilities.

Optimizing Distil-BERT through ONNX quantization[1] or knowledge distillation significantly reduces its size and speeds up inference, making it much more feasible in real-world applications. Logistic regression with feature engineering gives the lowest complexity alternative in a resource-constrained environment, though at the cost of some loss in accuracy due to its weighty nature and low computation requirement, thus being much easier to integrate and deploy. The best solution considers the target environment: Distil-BERT balances accuracy with functionality on high-performance systems, while logistic regression gives usability without overwhelming the system with too many resources on constrained setups. These made-to-measure solutions keep the Thunderbird plugin efficient, reliable, and accessible to a large user base.

[1] https://onnxruntime.ai/docs/performance/model-optimizations/quantization.html

8 Conclusion

8.1 Conclusions

This study has addressed the research questions by showcasing the effectiveness of autonomous email categorization through a comparative analysis of traditional machine learning and transformer-based deep learning models. The empirical results show that the transformer-based DistilBERT model has better classification performance with and without a confidence threshold than traditional algorithms at the cost of adding computational overhead considerations. The application of model-specific confidence thresholds improved the reliability of the classification of all models.

The design, development and deployment of the EClassify Thunderbird extension using the FastAPI backend architecture validate the practical feasibility of machine learning-based email categorization in real-world environments. Its modular architecture supports dynamic model selection about the computational resources at hand, showing how it can adapt to varied deployment constraints. Moreover, adding an "uncategorized" category opened the door to continuous learning potential, which includes the capability to handle uncertainty and its collection pattern of unclassified information. These have the potential to get even better through incremental learning and pattern adaptation.

Real-world implementation establishes the system's ability to maintain optimal performance while considering practical constraints. These findings provide a practical framework for autonomous email categorization that effectively connects theoretical machine learning methodologies with the actual requirements of email management.

8.2 Future Work

The choice of models for the classification of emails in Thunderbird has substantial directional implications for future research in this area. More recently, advanced models, like DistilBERT, have brought contextual understanding to the forefront of text classification. Success has shown that transformer-based architectures could handle much more difficult categorization tasks. Future research can dwell on refining such models to become much more efficient, realizing their deployment in resource-constrained environments without a penalty

in accuracy. Future possibilities may involve investigating techniques such as quantization, pruning, or knowledge distillation to find lighter versions of transformer models and close the gap with practicality.

The second important finding is data quality and class balance: model performances with balanced datasets can underline the critical role of pre-processing and data preparation; this may indicate future studies on automated methods of identifying and dealing with class imbalance using synthetic data generation or advanced sampling techniques. It may further explore the integration of feedback loops by users to provide means for dynamic updating of the classification models so that they remain relevant and efficient while email patterns change. The current implementation provides opportunities for substantial extension, particularly by incorporating machine reasoning and adaptive learning mechanisms.

Three main future research directions can aid in system improvement:

Unsupervised Pattern Recognition Framework: The clustering algorithms developed for high-dimensional email data would help to discover latent semantic structures in unclassified instances. This framework will apply the dimensionality reduction methods in combination with density-based clustering to reveal emerging communication patterns. However, it may expose natural category boundaries within the feature space using hierarchical agglomerative clustering methods, which could eventually reveal previously unrecognized classes of emails. That is, current threshold-based classification will be overcome by integrating unsupervised learning mechanisms for continuing category discovery.

Semantic Ontology Evolution System: A semi-supervised ontology expansion mechanism may be implemented; taxonomy evolution should be performed systematically. The system applies transformer-based semantic analysis to relate emergent patterns with existing categories such as management, personal, and projects. Causal inference methods and temporal pattern analysis techniques will be applied to quantitatively evaluate possible category modification. Therefore, the provided evidence helps decide that the ontological structure can be adapted by ensuring coherent category evolution with system stability.

Statistical Category Validation Architecture: Developing a statistical framework for category validation would ensure the systematic assessment of proposed taxonomic changes. This architecture would encompass information-theoretic measures for category distinctiveness, statistical significance testing of category coherence, Bayesian methods of inference for assessing the stability of categories, and cross-validation protocols for performance verification.

The validation framework would utilize probabilistic modeling to evaluate the stability and separation of categories, using metrics such as silhouette coefficients and mutual information scores. This enhancement would improve the confidence threshold mechanism by incorporating dynamic protocols for category validation.

Integrating these components would require a meta-learning system capable of orchestrating the interaction between pattern discovery, ontology evolution, and validation processes. The resulting system would aim to find the optimal trade-off between exploration and discovering new categories and exploitation, maintaining the performance in already known categories by employing reinforcement learning mechanisms.

Future work should also investigate the implications of computational complex-

ity due to such additions, specifically in real-time email classification within the Thunderbird client. It is worth looking into efficient approximation algorithms and incremental learning methods to maintain the system's responsiveness under continuous adaptation.

This proposed enhancement framework would be an adaptive learning architecture capable of autonomous evolution in response to changing communication patterns. The integration of these components allows for principled advancement beyond the current threshold-based approach while retaining stringent performance guarantees.

The challenges in deploying machine learning models in email clients open up other possible research avenues: edge computing and federated learning. The latter would train or make inferences locally on users' devices to maintain privacy and reduce the burden on centralized servers. There is an increasing interest in privacy-preserving AI solutions involving applications such as email, which is widespread in handling personal information.

The trade-offs between traditional and deep learning models point to the potential for hybrid approaches. Future research could investigate how to combine lightweight models with embeddings from transformers, offering a middle ground between speed and contextual accuracy. These innovations could pave the way for more adaptable and efficient email classification systems that contribute to natural language processing and its practical applications.

9 Appendix

A.1 Model Loading and Feature Processing

```
import torch
from transformers import BertTokenizer, AutoTokenizer
from transformers import AutoModelForSequenceClassification

# print(np.__version__)
import xgboost as xgb

app = FastAPI()

def tokenize(row): ...

def remove_reg_expressions(row): ...

def assemble_bag_single(data): ...

def prepare_features_single(email_text, model_columns): ...

# Load models from saved files
def load_xgboost_models(base_path, filename_prefix="xgboost_model"): ...

# Load the model in FastAPI
def load_model_bert(path=r".\Models\bert_model.pth"): ...

def load_models(): ...

log_regression_models, rf_models, xg_models = load_models()

# Load tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("distilbert/distilbert-base-uncased")
model = load_model_bert()
```

Listing 1: Model Loading and Feature Processing

A.2 Continuous Monitoring

```
function classifyEmail(messageId) {
}

function onNewMailReceived(folder, messages) {
  messages.messages.forEach((message) => {
    classifyEmail(message.id);
  });
}
```

Listing 2: Continous Monitoring

A.3 Email Tagging

```
let tagsToAdd = [];
let tagsToRemove = [];  // Remove tags based on the classification

getEmailContent(messageId)
  .then((content) => {
    console.log('Email Content:', content);

    classifyEmailContentAPI(content)
      .then((newTagName) => {
        // Get the color for the new tag name (default to black if not found)
        let tagColor = classColorMapping[newTagName] || "#000000"; // Default to
            black
        console.log("NEW TAG NAME:", newTagName);
        let newTagKey = '$' + newTagName;

        // Check if a tag with the name exists in tagNameToKeyMap
        let tagKey = tagNameToKeyMap[newTagName];
        if (!tagKey) {
          // Create the tag if it doesn't exist
          messenger.messages.tags.create(newTagKey, newTagName, tagColor)
            .then(() => { /* success logic */ })
            .catch((error) => { /* error handling */ });
        } else {
          tagsToAdd = [tagKey];
        }
      });
  });
```

Listing 3: Email Tagging

A.4 Handling Menu Click Events

```
messenger.storage.onChanged.addListener((changes, area) => {
  // Handle storage changes
});

messenger.menus.onClicked.addListener((info, tab) => {
  // Handle menu item clicks
});

function handleMenuClick(info, messageId) {
  // Load the latest bayesData with every click
  messenger.storage.local
    .get("key")
    .then((result) => {
      // Process the result
    })
    .catch((error) => {
```

```
      // Handle errors
    });
}
```

Listing 4: Handling Menu Click Events

A.5 /classifyemail Endpoint

```
@app.post("/classifyemail")
async def classify_email(emailContent: dict):
    # Your email content
    print(emailContent['emailContent'])

    y_hat = predict_bert(emailContent['emailContent'])
    print(y_hat, type(y_hat))
    category = class_le.inverse_transform([y_hat])[0]
    return {"category": category}  # tag/class
```

Listing 5: /classifyemail endpoint

A.6 /classifyemailensemble Endpoint

```
@app.post("/classifyemailensemble")
async def classify_email_ensemble(emailContent: dict):
    email_text = emailContent['emailContent']

    y_hat_bert = predict_bert(email_text)
    y_hat_reg = predict(email_text, log_regression_models, filtered_folders,
        rf_model_columns)
    y_hat_rf = predict(email_text, rf_models, filtered_folders, rf_model_columns)
    y_hat_xg = xg_predict(email_text, xg_models, filtered_folders, rf_model_columns)

    y_hat_bert = class_le.inverse_transform([y_hat_bert])[0]
    y_hat_reg = class_le.inverse_transform([y_hat_reg])[0]
    y_hat_rf = class_le.inverse_transform([y_hat_rf])[0]
    y_hat_xg = class_le.inverse_transform([y_hat_xg])[0]

    analysis = {"Classification": "Supervised Models", ...}
    return analysis
```

Listing 6: /classifyemailensemble endpoint

Bibliography

[1] Wael Abou Awad and SM ELseuofi. "Machine learning methods for spam e-mail classification". In: *International Journal of Computer Science & Information Technology (IJCSIT)* 3.1 (2011), pp. 173–184.

[2] L Breiman. "Random Forests". In: *Machine Learning* 45 (Oct. 2001), pp. 5–32. DOI: 10.1023/A:1010950718922.

[3] Jason Brownlee. *XGBoost With python: Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery, 2016.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[5] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20 (1995), pp. 273–297. DOI: 10.1007/BF00994018.

[6] Mary Czerwinski, Paul Johns, and Shamsi T. Iqbal. "The Cost of Email Use in the Workplace: Lower Productivity and Higher Stress". In: 2016. URL: https://api.semanticscholar.org/CorpusID:36974961.

[7] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, et al. "Machine learning for email spam filtering: review, approaches and open research problems". In: *Heliyon* 5.6 (2019).

[8] Alexander Genkin, David Lewis, and David Madigan. "Large-Scale Bayesian Logistic Regression for Text Categorization". In: *Technometrics* 49 (Aug. 2007). DOI: 10.1198/004017007000000245.

[9] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. "Variable selection using random forests". In: *Pattern Recogn. Lett.* 31.14 (Oct. 2010), pp. 2225–2236. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2010.03.014. URL: https://doi.org/10.1016/j.patrec.2010.03.014.

[10] Chih-wei Hsu, Chih-chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin". In: (Nov. 2003).

[11] Thorsten Joachims. "Text Categorization with Support Vector Machines". In: *Proc. European Conf. Machine Learning (ECML'98)* (Jan. 1998). DOI: 10.17877/DE290R-5097.

[12] Bryan Klimt and Yiming Yang. "The Enron Corpus: A new dataset for email classification research". In: *European conference on machine learning*. Springer. 2004, pp. 217–226.

[13] Deepak Kumar and Rahul Kumar. "Spam Filtering using SVM with different Kernel Functions". In: *International Journal of Computer Applications* 136 (Feb. 2016), pp. 16–23. DOI: 10.5120/ijca2016908395.

[14] Gilles Louppe. *Understanding Random Forests: From Theory to Practice*. 2015. arXiv: 1407.7502 [stat.ML].

[15] Scott Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. "From Local Explanations to Global Understanding with Explainable AI for Trees". In: *Nature Machine Intelligence* 2 (Jan. 2020). DOI: 10.1038/s42256-019-0138-9.

[16] Ghulam Mujtaba, Liyana Shuib, Ram Gopal Raj, Nahdia Majeed, and Mohammed Ali Al-Garadi. "Email classification research trends: review and open issues". In: *IEEE Access* 5 (2017), pp. 9044–9064.

[17] Umesh Kumar Sah and Narendra Parmar. "An approach for malicious spam detection in email with comparison of different classifiers". In: *International Research Journal of Engineering and Technology (IRJET)* 4.8 (2017), pp. 2238–2242.

[18] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

[19] Iqbal H Sarker. "Machine learning: Algorithms, real-world applications and research directions". In: *SN computer science* 2.3 (2021), p. 160.

[20] Vishal Kumar Singh and Shweta Bhardwaj. "Spam mail detection using classification techniques and global training set". In: *Intelligent Computing and Information and Communication: Proceedings of 2nd International Conference, ICICC 2017*. Springer. 2018, pp. 623–632.

[21] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. "Bias in random forest variable importance measures: illustrations, sources and a solution". In: *BMC Bioinformatics* 8 (Jan. 25, 2007), p. 25. DOI: 10.1186/1471-2105-8-25.

[22] Chi Sun, Luyao Huang, and Xipeng Qiu. "Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 380–385. DOI: 10.18653/v1/N19-1035. URL: https://aclanthology.org/N19-1035.