Journal of Technology and Informatics (JoTI)
Vol. 6, No. 1, October 2024
P-ISSN 2721-4842
E-ISSN 2686-6102
DOI: 10.37802/joti.v6i1.715

Exploring the Application of Machine Learning for Automatic Inbound Email Classification in CRM System at XYZ Company

Lukman Arif Sanjani[1], **Raden Bimo Mandala Putra**[2], **Umi Laili Yuhana**[3]
[1,2,3]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
e-mail: 6025232027@student.its.ac.id[1], 6025232022@student.its.ac.id[2], yuhana@if.its.ac.id[3]
* Corresponding author: E-mail: 6025232027@student.its.ac.id

Abstract: *Customer service has become increasingly crucial in today's business landscape, necessitating companies to provide fast, responsive, and personalized assistance to their clientele. However, amidst the challenges posed by surges in email volume, manual categorization and response strategies often lead to performance declines. To address this, we propose a system leveraging Machine Learning techniques for automated email classification. Our evaluation reveals promising results, with SVM achieving the highest accuracy of 96.59%, followed by XGB (96.02%) and RF (95.27%). These models exhibit commendable precision, recall, F1 scores, and Matthews Correlation Coefficient (MCC), showcasing their effectiveness in improving customer service efficiency and responsiveness. This integration of technology not only enhances operational efficiency but also fosters harmonious customer relationships, ultimately leading to increased loyalty and profitability for companies.*

Keywords*: Classification, Customer Service, Machine Learning, Random Forest, SVM, XGBoost*

INTRODUCTION

Customer service has become an increasingly important aspect in the current era, especially in the context of intensifying business competition. It is no longer sufficient to compete solely on the basis of product quality or service; companies must also be able to provide fast, responsive, and comfortable service to their customers. Customer satisfaction fosters harmonious relationships between product / service providers and consumers, ultimately leading to consumer loyalty and profitability for the company. Amidst rapid technological advancements and rising customer expectations, the integration of technology to enhance customer service has become imperative. Automation is a crucial aspect of the current Digital Era. With advancements in Artificial Intelligence (AI) and Machine Learning (ML), many companies are adopting these technologies across various operational aspects. Automation not only allows companies to improve efficiency but also enables them to deliver more personalized and responsive service to customers.

PT. XYZ is a company operating in the field of Customer Relationship Management (CRM). Being part of an industry heavily reliant on customer service, there are often significant spikes in email volume at certain times, causing difficulties for available customer service personnel to respond promptly to the influx of emails. This leads to a decline in the performance rating of the Customer Service team. To address this issue, the Customer Service team manually reads each incoming message, categorizes them, and then responds using pre-existing templates tailored to each category. While this method is somewhat helpful, it is still deemed ineffective in improving Customer Service performance during email spikes, as the team must individually read and categorize each email. To address this challenge, a system utilizing Machine Learning approaches is needed, where the system will automatically classify incoming email categories based on patterns in the text content. In this study, three machine learning algorithms were employed, as follows:

1. Support Vector Machines (SVM): Machine learning is widely used in classification, such as using SVM in email classification systems. For instance, Drucker created a spam classification system using SVM that incorporated a thousand features and spanned seven thousand dimensions, resulting in shorter training times and enhanced accuracy and speed [1]. Paper [2]proposed a spam filtering algorithm with SVM, achieving higher accuracy by reducing false positive rates. The result produced by this classifier is a hyperplane that aims to maximize the distinction between feature vectors belonging to different classes. When presented with a new instance, SVM assigns a label based on the subspace to which its feature vector is aligned.
2. Random Forest: A well-known approach in Supervised Learning is the Random Forest technique. It's proficient in addressing both classification and regression challenges within machine learning. Random Forest has demonstrated its efficacy in implementing ensemble learning concepts, where numerous classifiers are merged to tackle intricate problems and enhance model accuracy[3]. This method operates on two fundamental principles: firstly, it generates numerous decision trees and amalgamates them into a unified model, and secondly, it makes its final decision based on the majority consensus among the trees under consideration.
3. XGBoost: XGBoost Classification stands out as a robust machine learning technique tailored for classification purposes. It represents a specialized adaptation of the XGBoost algorithm crafted explicitly to address classification challenges. XGBoost, which stands for "Extreme Gradient

Boosting," has become popular due to its ability to provide highly accurate predictions in various classification tasks [4]. XGBoost has become popular for its high performance in machine learning tasks. It builds a series of decision trees sequentially, with each model correcting the errors of the previous ones. The final prediction is made by weighted averaging of all models' forecasts.

Previous research about this topic has examined email classification, primarily focusing on spam detection and grouping emails into various folders based on different categories [5][6]. Additionally, previous studies have also explored hierarchical folder classification methods [7]. While previous research focused on classifying emails into spam and non-spam categories, our research has developed a new API system that can automatically classify email categories based on the text content. The category names can be dynamically set by users of a CRM application. Furthermore, the system developed in this research has the potential to be implemented in CRM systems of different companies.

METHOD

In this research, a separate system will be developed, which will later be connected to the CRM system through an API. This is done to facilitate easier application maintenance and further development in the future. For the development of this classifier application, the Waterfall methodology was employed. The waterfall model was chosen due to its structured, sequential approach. This method covers gathering requirements, design, implementation, verification and maintenance stages, ensuring clear direction and control, and minimizing errors [8]. The stages of the Waterfall method are illustrated in Figure 1.

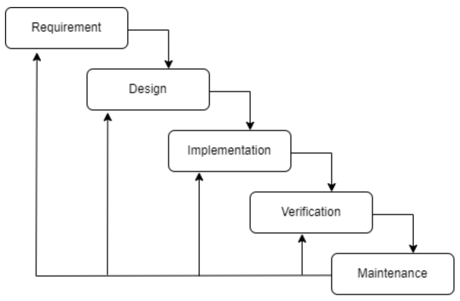

Figure 1. Waterfall SDLC Method

Requirements Analysis

In this stage, the developer initiates dialogue to grasp users' requirements and software limitations. This insight is commonly acquired through interviews, discussions, or in-person surveys. Following this, the collected data undergoes analysis to extract essential user-centric information.

System Design

During this phase, the requirements specification acquired from the preceding stage is reviewed, and the system design is formulated. The purpose of system design is to identify hardware and system prerequisites, as well as to outline the overall architecture of the system. In this study, a Use Case Diagram was also created. It was used to illustrate the interaction between actors and activities within the use case. Additionally, this diagram provides a comprehensive model of what is being done, who is involved internally, and which individuals or actors play a role externally in order to explain the various functions that users / actors can perform within the Email Classification Application [9].

Implementation

At this point, the system undergoes its initial development phase, where it is broken down into small programs termed as units, which will later be integrated in the following stage. Every unit will individually developed and then will be tested for functionality, a procedure commonly named by unit testing.

Integration Testing And Verification

Once each unit crafted during the implementation stage undergoes thorough individual testing, they are seamlessly amalgamated into the system. Following integration, rigorous testing of the entire system ensues, aiming to unearth any latent flaws or discrepancies.

Operation And Maintenance

In the concluding stage of the waterfall model, the completed software is launched and sustained through maintenance operations. This maintenance phase includes addressing any overlooked errors and optimizing system components, as well as enhancing system functionality to accommodate emerging needs.

RESULT AND DISCUSSION

Requirement

The required elements in the development of this application are the functional requirements list, as well as data from previous emails which will be needed to create the machine learning model. The main features that must be possessed by the application are Dataset Storage, Model Training, & Data Prediction.

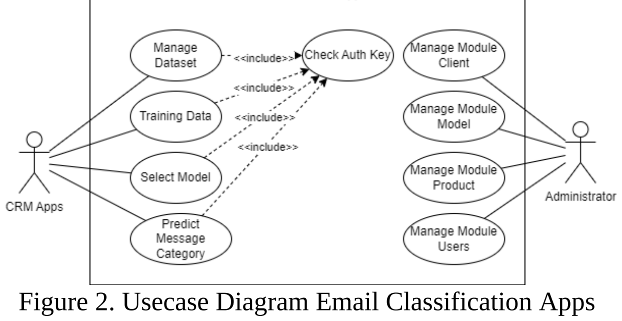

Figure 2. Usecase Diagram Email Classification Apps

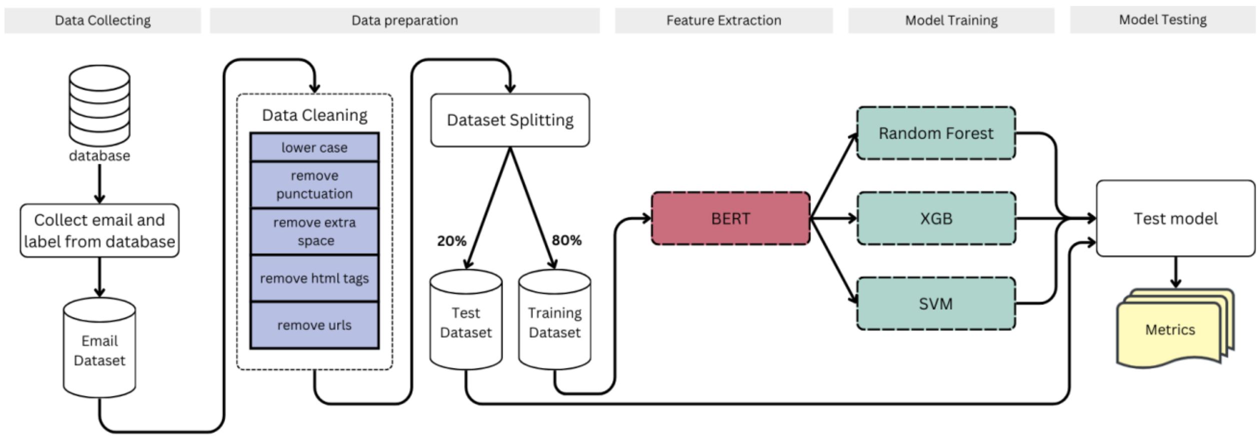

Figure 3. Machine Learning Workflow and Architecture

System Design

1. API Architecture

Initial design is required to ensure that the resulting application meets the needs and business processes of XYZ Company. This system is built with the aim that users / customer service leaders can send datasets for training, perform data training, and choose the machine learning model to be used independently. This can make the company's business processes more efficient. Of course, for the first few weeks of implementation, Customer Service must manually classify procedures first. When the data has been collected, it can be sent to the email classifier application through the existing CRM application for data training. Below is the Use Case Diagram explaining the processes that can be performed by the Customer Service Leader on the Email Classifier Application in Figure 2.

The data in this application will be stored in a Relational Database Management System (RDBMS) MySQL. All datasets received from the CRM System through the API will be stored in the training_datas table. Previously, the Administrator of this Application will configure data related to Client, available Machine Learning Models, Name of Product / Campaign / Division of the client, and user data from this application. For the model data, it will be stored in a File, then the path along with the results will be stored in the model_product table. Figure 4 shown the complete overview of the Entity Relational Diagram in this application. Here is an explanation of the functions of each table in Figure 4:

- users: This table is used to store data on the application's users in this research.
- clients: This table indicates the company from which the users engaging with the application originate. This is provided to facilitate potential integration with other companies' CRM applications.
- training_datas: This table is used to store various training data input by the users.
- models: This table stores the names of the machine learning algorithms used in this research, namely SVM, Random Forest, and XGBoost.

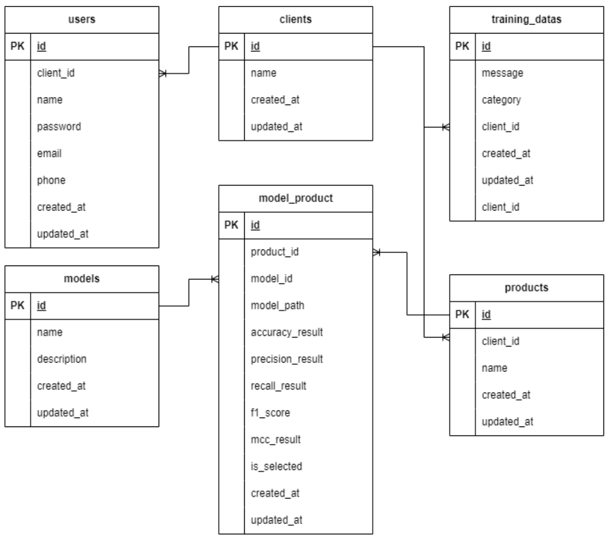

Figure 4. Entity Relational Diagram Database Email Classifier Apps

- products: Each client may have several service products, each with different training data. This table is needed to distinguish between these products.
- model_products: This table is used to store the results of the machine learning modeling. The model files are stored, and their paths are recorded in the model_path column. Additionally, the evaluation results of each mode are stored. This table also keeps records of which models are used by users for predictions when new email data is received into their CRM application.

In this application, communication between Customer Service and the application cannot be done directly, but must go through the CRM application commonly used by Customer Service. Eventually, the two applications will be connected through API communication. The main API used in this application is listed in Table 1. When accessing each endpoint, validation of the Authorization Key will be performed first, where this Key will be created by the Administrator of the email classifier application when adding a new Client. This is done to ensure that every HTTP Request

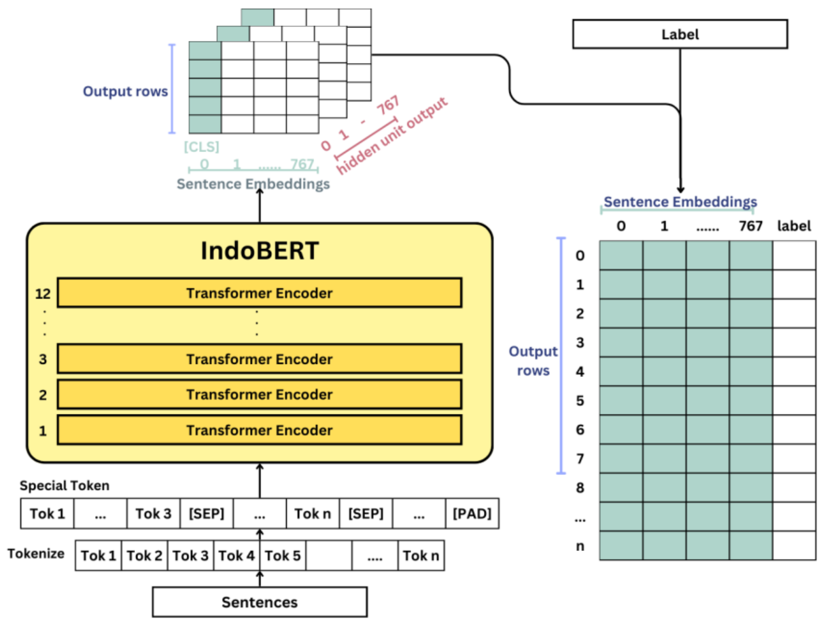

Figure 5. Feature Extraction Workflow

transaction is secure and not infiltrated by requests from other entities. The first API is used to send datasets to the system, which includes functions to delete old datasets and input new datasets for the client. Then, the second API is used to perform the Train Model of Machine Learning action. The third API is used to select the model that will be used for auto prediction/auto classification. Finally, the last API is used to perform prediction/classification if there are any mixed emails entering the CRM system.

Table 1. List of Main API Endpoint

Method	URL API	Parameter
POST	https://{base_url}/api/training-data/create	client_id, product_id, data (contains an array of Message & Category)
POST	https://{base_url}/api/model/train	client_id
POST	https://{base_url}/api/model/choose	client_id, model_id
POST	https://{base_url}/api/model/predict	client_id, message

2. Machine Learning Workflow and Architecture

Figure 3 describes the approach employed in our experiments. The study's workflow is segmented into four main sections: data collection, data preparation, feature extraction, and training of machine learning.

2.1. Data Collecting

The data is sourced from the API database, where clients have previously uploaded data to the training data table via the API. To evaluate the system's performance, we employ the XYZ Company dataset as a reference point for evaluating the architecture's capabilities. This dataset is loaded into the training_datas table before retrieving the data from the database for analysis. The evaluation dataset utilized consists of emails sent by clients to the XYZ Company helpdesk via email. These emails have been annotated by workers affiliated with XYZ Company. The dataset comprises a total of 1112 emails, with labels classified into three distinct categories: 'Question', 'Incident', and 'Request'. The distribution of emails across these categories is as follows: 'Question': 739, 'Incident': 115, 'Request': 258

2.2. Data Preparation

In order to evaluate the model's performance, we divided the evaluation dataset into 80% for training data and 20% for testing data. Subsequently, in pursuit of enhanced model accuracy, we implemented cleaning procedures on both the training and testing datasets to eliminate noise. We employed various text cleaning methods, including:

a) Lower casing
b) Remove Html tags.
c) Remove URL.
d) Remove certain punctuation
e) Remove extra space

2.3. Feature Extraction

In our research, we employed pre-trained IndoBERT, a transformer-based model similar to BERT, trained on an Indonesian corpus. It functioned as an extractor of feature, harnessing contextual information to generate word embeddings from textual data. [10]. The BERT model was selected because previous studies have already conducted a comparison of the performance of various Indonesian BERT models, with Indo BERT emerging as the top performer [11]. In this study, IndoBERT serves primarily as a feature extraction tool with two main functions tokenization and embedding. Main process flow of IndoBERT can be shown on Figure 5.

BERT, a sophisticated embedding model, possesses the remarkable ability to grasp contextual nuances, generating fluid embeddings for words within

Table 2. Training Result

Model	Precision	Accuracy	Recall	MCC	F1 Score
Random Forest	0.952657	0.952652	0.952652	0.905307	0.952651
XGBoost	0.960390	0.960227	0.960227	0.920613	0.960222
SVM	0.966334	0.965909	0.965909	0.932238	0.965899

their contextual framework. It perceives sentences as sequences of tokens and can adeptly handle both single sentences and sentence pairs. Token embeddings are meticulously crafted from a vocabulary pool assembled from Word Piece embeddings, encompassing an extensive repertoire of 30,000 tokens.[12]. In our research, we leverage the AutoTokenizer library to facilitate the tokenization process, harnessing the power of pre-trained IndoBERT sourced from Hugging Face. BERT's functionality extends beyond mere token embeddings, incorporating positional embeddings and segment embeddings for each token, enriching the model's contextual understanding. [13]. Positional embeddings hold information about the sequence of tokens, while segment embeddings indicate sentence boundaries: tokens from the first sentence receive an embedding of 0, and tokens from the second sentence receive an embedding of 1 [14]. Prior to encoding text with BERT, the tokenizer inserts the [CLS] token at the beginning of the initial sentence for classification tasks, and the [SEP] token at the conclusion of each sentence . This process ensures that BERT can effectively capture the semantic meaning of the input text by considering the entire context, while also providing clear markers for sentence boundaries. These tokens play a crucial role in the architecture of BERT, facilitating tasks such as sentence classification, semantic similarity assessment, and named entity recognition [13]. After tokenizing the input text using the pre-trained IndoBERT tokenizer, the next step is to extract the vector embeddings associated with each token. These embeddings capture the contextual information of each token within the given text. Specifically, the vector embedding for each token is obtained from the hidden state that the IndoBERT model outputs for that token.

In our architecture, we focus on utilizing the hidden state corresponding to the special token [CLS] for classification tasks. The [CLS] token represents the aggregated representation of the entire input sequence and is often used as a summarization of the semantic content of the text. By extracting the hidden state of the [CLS] token, we effectively capture the representative semantic meaning of the entire sequence [14]. The extracted vector embeddings are then fed into downstream classification tasks These tasks leverage the contextual embeddings provided by IndoBERT to make predictions or perform further analysis on the input text [14].

2.4. Model Training

We selected the 3 Machine Learning classifiers algorithm, SVM, Random Forest and XGB to choose. They were implemented using Scikit-learn, a Library of Python Machine Learning to build the predictive models. The models will be trained on client private data, then save the model to the model library.

2.5. Test Model

Users of our API can compare the performance of models using five metrics: Precision, Accuracy, F1 Score, Recall, and Matthews Correlation Coefficient (MCC), customizing their evaluation according to their specific needs and preferences.

- Overall Accuracy

This metric computes the ratio of properly predicted occurrences to total instances in the dataset. It offers a comprehensive evaluation of the model's effectiveness However, accuracy alone can be misleading in imbalanced datasets, hence the need for additional metrics.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- Precision

Refers to the accuracy of confident forecasts. It is the proportion of accurately predicted positive observations to total projected positive observations. Precision is crucial when the cost of false positives is high.
.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- Recall (Sensitivity)

Recall also known as sensitivity or true positive rate, represents the proportion of actual positive cases that were accurately recognized by the model. Recall is important in scenarios where missing a positive case is costly.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- F1 Score

The F1 score represents the balance of precision and recall. It maintains a balance between precision and recall and is effective in classes that are uneven. The F1 score is particularly useful when both false positives and false negatives are important

Table 3. Testing Result

Method	URL API	Parameter	Result
POST	https://{base_main_url}/api/training-data/create	client_id, product_id, data (contains an array of Message & Category)	Success
POST	https://{base_main_url}/api/model/train	client_id	Success
POST	https://{base_main_url}/api/model/choose	client_id, model_id	Success
POST	https://{base_main_url}/api/model/predict	client_id, message	Success

$$F1\ Score\ = 2x\ \frac{Precision\ x\ Recall}{Precision +\ Recall} \qquad (4)$$

- Matthews Correlation Coefficient (MCC)

MCC represents the correlation coefficient between observed and predicted binary classifications. It considers genuine and false positives and negatives and is widely recognized as a balanced metric. MCC is especially useful for imbalanced datasets because it provides a more informative and truthful score in such situations compared to accuracy.

$$MCC = \frac{(TP\ x\ TN\ -\ FP\ x\ FN)}{\sqrt{(TP\ +\ FP)(TP\ +\ FN)(TN\ +\ FP)(TN\ +\ FN)}} \qquad (5)$$

In these equations:
- TP means True Positives
- TN means True Negatives
- FP means False Positives
- FN means False Negatives

By using these metrics, users can get a well-rounded understanding of their model's performance, ensuring that they can make informed decisions based on the specific requirements and potential costs associated with false positives or false negatives in their applications.

Implementation

After the analysis and design stages have been completed, the next stage is the implementation stage. The application is built using the Python language with the Flask Framework. The structure used is MVC (Model, View, Controller). This is done considering that the application can be easily further developed in terms of functionality in the future [15]. Flask Framework itself is chosen because this application is in the form of an API service, so it is more suitable to use a Microframework for faster performance. Additionally, Flask is chosen because of its maturity, having been released for 14 years, making this framework stable in functionality and performance. Third-party libraries used for modeling include Scikit-learn and Xgboost. Using the API for training models shows quite satisfactory results, with the highest model accuracy achieved by the SVM model with an accuracy of 0.97. The complete experimental results can be found in Table 2.

Integration Testing And Verification

We have conducted comprehensive testing on all primary APIs, confirming their performance aligns with our expectations. Each API has successfully produced the intended results, meeting all specified requirements and functionality criteria. The complete testing results can be found in Table 3.

Operation And Maintenance

Based on the successful testing outcomes and the application's operational functionality, it can be concluded that the application is ready for use. Any deviations or errors encountered will be addressed promptly through further refinement processes

CONCLUSION AND SUGGESTION

This application has successfully performed classification effectively with high accuracy, thus potentially facilitating future customer service operations.

Further research is needed to investigate how the system performs when dealing with a larger number of categories beyond three. If there is a significant decrease in performance, further development may be necessary, possibly incorporating deep learning techniques

REFERENCES

[1] H. Drucker, D. Wu, and Y. N. Yapnik, “Support Vector Machines for spam categorization,” *IEEE Trans Neural Netw*, vol. 10, no. xx, 1999.

[2] Md. R. Islam, M. U. Chowdhury, and Wanlei Zhou, “An Innovative Spam Filtering Model Based on Support Vector Machine,” in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06)*, IEEE, pp. 348–353. doi: 10.1109/CIMCA.2005.1631493.

[3] C. Anilkumar, A. Karrothu, N. S. Mouli, and C. B. Tej, “Recognition and Processing of phishing Emails Using NLP: A Survey,” in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, Jan. 2023.

[4] R. Suprayoga, S. Zega, Muhathir, and S. Mardiana, “Classification of Mango Leaf Diseases Using XGBoost Method and HoG Feature Extraction,” in *2023 International Conference on Modeling & E-Information Research, Artificial Learning and Digital Applications (ICMERALDA)*, 2023.

[5] K. Schneider, “A comparison of event models for Naive Bayes antispam e-mail filtering,” in *Proceedings of the II th Conference of the European Chapter of the Association for Computational Linguistics (EACL’03)*, 2003.

[6] K. Iqbal and M. S. Khan, “Email classification analysis using machine learning techniques,” *Applied Computing and Informatics*, May 2022, doi: 10.1108/ACI-01-2022-0012.

[7] J. D. Brutiag and C. Meek, “Challenges of the email domain for text classification,” in *ICML*, 2000, pp. 103–110.

[8] A. Putu Candra *et al.*, “Sistem Informasi Penjualan Online Thrift Shop Berbasis Web,” *Journal of Technology and Informatics (JoTI)*, vol. 5, no. 2, pp. 116–124, Apr. 2024, doi: 10.37802/joti.v5i2.586.

[9] A. Hidayatur, A. Yuana, A. Wafi, R. Harjo, T. Maulana, and A. Terza, “Pengembangan Proses Bisnis Pelayanan Statistik Terpadu Badan Pusat Statistik Kota Surabaya Menggunakan Metode Prototyping,” *Journal of Technology and Informatics (JoTI)*, vol. 5, no. 2, pp. 70–79, Apr. 2024, doi: 10.37802/joti.v5i2.548.

[10] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, “IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP,” in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds., Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 757–770. doi: 10.18653/v1/2020.coling-main.66.

[11] H. Lucky, Roslynlia, and D. Suhartono, “Towards Classification of Personality Prediction Model: A Combination of BERT Word Embedding and MLSMOTE,” in *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*, 2021, pp. 346–350. doi: 10.1109/ICCSAI53272.2021.9609750.

[12] Pavithra and Savitha, “Topic Modeling for Evolving Textual Data Using LDA, HDP, NMF, BERTOPIC, and DTM With a Focus on Research Papers,” *Journal of Technology and Informatics (JoTI)*, vol. 5, no. 2, pp. 53–63, Apr. 2024, doi: 10.37802/joti.v5i2.618.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”.

[14] S. Ravichandiran, *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing Ltd, 2021.

[15] L. A. Sanjani, S. J. Hartati, and P. Sudarmaningtyas, “Rancang Bangun Sistem Informasi Penggajian Pegawai dan Remunerasi Jasa Medis Pada Rumah Sakit Bedah Surabaya,” *Jurnal Sistem Informasi dan Komputer Akuntansi (JSIKA)*, vol. 3, no. 1, pp. 87–93, 2014.