**Email Classification for Helpdesk**

1. **Abstract**Email has taken a significant role in all businesses around the world. It is a fast and effective way to communicate between customers and company or organization, but it is also an inherent problem when the volume of email increases. It is posing a challenge for email management and decision making based on the purpose of the emails. With the hope to find a way to help to automate the making decision process, this proposal outlines a project aimed at developing a machine learning model for the classification of email content. The primary objective is to automate the decision-making process, shedding light on the specific nature of each email, whether it pertains to IT support, general inquiries, or addresses particular issues within the IT domain base on this classification information another system can take appropriate action such as automating ticket creation process and assign the ticket to appropriate department or specific person for the next action in the process.
2. **Introduction**  
   Email plays a significant role in modern businesses, supporting quick and effective communication between customers and organizations. However, the increase in email volume poses challenges in effective communication and efficient management. The problem happens in all organizations manual email management and decision-making.  
     
   Our proposal introduces an Email Classification project focused on developing a machine learning model for email content classification. The main goal is to automate classifying email process, providing detailed insights into each email's nature to pose a decision-making whether it related to IT support, general inquiries, or specific IT issues.  
     
   Manual Email classification is not just a time-consuming task, it also incurs significant costs. Automating this process could save substantial time for helpdesk agents, allowing them to prioritize critical tasks. The long run efficiency promises bring benefits for organizational performance.  
     
   The Email Classification Project is an experimental initiative leveraging machine learning to autonomously categorize emails. Using email content as input, the model outputs the email type. This output, in turn, becomes a valuable input for decision systems, contributing to broader automation efforts within organizations.
3. **Motivation**  
   Creating a framework to automate email classification is tough because the nature of email content is varied and unstructured. Therefore, it cannot be processed as structured data.   
      
   Existing solutions are based on predefined rules to search through emails’ content to check some phrases of words and based on these words for clarifying the emails. These kinds of solutions are half AI solutions, and they are fragile solutions. Because emails’ content can be arbitrary and if there are any things changed in the email these programs need to be modified. This proposal aims to use machine learning to tackle practical email classification issues, contributing to business process automation.   
     
   A major challenge is the lack of suitable data, especially with sensitive business emails. The solution for it is to use Generative AI to create synthetic data for training and testing the model. This ensures the model learns from diverse scenarios, making it adaptable and robust in handling real-world email content.
4. **Methods**  
   A diagram of a software process

   Description automatically generated
   1. **Preprocessing steps**: Lowercasing is the first step in NLP, it helps reduce the vocabulary size and consistency in the presentation of words. Filtering stop words, rare words and corpus-specific common words, these kinds of words don’t bring a significant meaning and some of them also contribute noise to the dataset. Filtering them helps improve the performance of the model.
   2. **Feature Engineering:** Text data is not ingestible into machine models, so the feature engineering technique is applied to convert text into numeric that can fix into the models. Tf-Idf (Term Frequency - Inverse Document Frequency) is a used to convert to vector of numeric. The target labels of email classification are also text so they must be encoded into numeric and encoding class technique is applied. After the readiness of the dataset, it is split into training and testing set. Training data is used to train the model and testing data to check its performance.
   3. **Choosing Algorithm:** The targets are multi kinds of email type. It is a multi-classifications task, MultinomialNB, LinearSVC are two of many algorithms supporting classification tasks that are selected for this experiment and the better score output will be selected.
   4. **Train and Evaluate Model**: The performance of the model depends on many factors. Input data, Algorithms and Hyperparameters. At this phase, input data is ready so to select the right model, Algorithms and Hyperparameters are two factors need to be pairs selected. A technique that can be applied is Grid Search CV algorithm. It evaluates the best params with the right algorithm. To improve the performance of the output model Cross validation 5-folded is also applied while training the model. This technique ensures that all the data will be used for training and validation. Finally, metrics such as accuracy, precision, recall, F1-score, false positive rates and receiver operating characteristic (ROC) curves are used to measure the performance of the model.
5. **Data Collection**  
   Email collections for specific areas of business are not available and they are also impossible to collect because of sensitive information. To make this experiment work, Synthethic data solution is going to be applied. Email collection can be collected from Generative AI models are available on market such as ChatGTP or Elastic Email tool.
6. **Classification Type of Email Scope**

Email Classification is a very general term. It can be any email type that needs to be classified. To make this proposal possible, the scope of email classification needs to be defined clearly. In addition, Sources of email for this proposal is also rare. The sources in this proposal are collected from <https://www.cs.cmu.edu/~./enron/> and from chat gpt and the email types are limited to this list:

1. HR

2. Shipping

3. Business

4. IT

5. Sale

1. **Implementation**The above diagram is the class diagram of email classification including three classes:

* SyntheticEmailExtraction class handles synthetic emails which were collected from ChatGPT.
* EnronEmailExtraction class handles enron’s extracted email dataset.
* EmailClassificationExtraction is a wrapper class which handles both kinds of emails data, synthetic email and enron’s extracted email dataset.

Beside the above classes, there are also some utility functions that help handle repeated common tasks:

* hyperparameters\_tuning: find best parameters for a model and visualize a heat chart
* unique\_words\_count: count words
* summarize\_emails: report email summary.
* remove\_stopwords: remove stop words from email dataset.
* remove\_punctuation: remove punctuation from email dataset.
* remove\_stopwords\_and\_punctuation: a wrapper function to remove stop words and punctuation.
* perform\_cross\_validation\_and\_report: perform a cross validation for a model and report the metrics.
* perform\_learning\_curve\_and\_visualization: perform a learning curve for a model and visualize the chart.

1. **Metrics Summary for MultinomialNB Model**

**8.1 Crovalidation Metrics Report**

A screenshot of a computer

Description automatically generated

**8.2 Learning Curve Metrics Report**

A screen shot of a graph

Description automatically generated

**8.3 Hyperparameters Metrics Report**

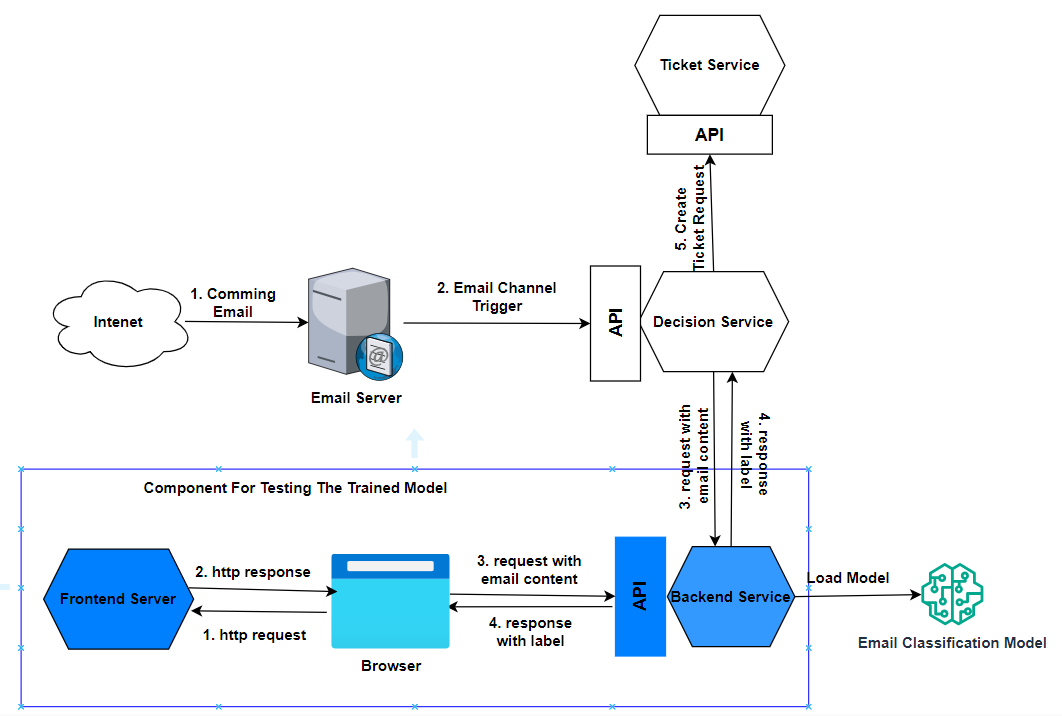
**A screenshot of a computer

Description automatically generated**

**A graph with numbers and squares

Description automatically generated**

1. **Deployment**



The above diagram includes one use-case of Email Classification Model which are Email Server, Decision Service and Ticket Service. These components are out of scope in this proposal. The remaining components are in blue ones which are used for demonstration and testing in this proposal.

1. **Source Code**

**10.1. Jupyter notebook file:** this is the experiment notebook file.

**10.2. Backend Server:** check README.md to run the server **A screenshot of a computer

Description automatically generated**

To test backend rest api, postman desktop application can be used to test backend api. Paste the flowing data into the body -> raw text of the poat man request.

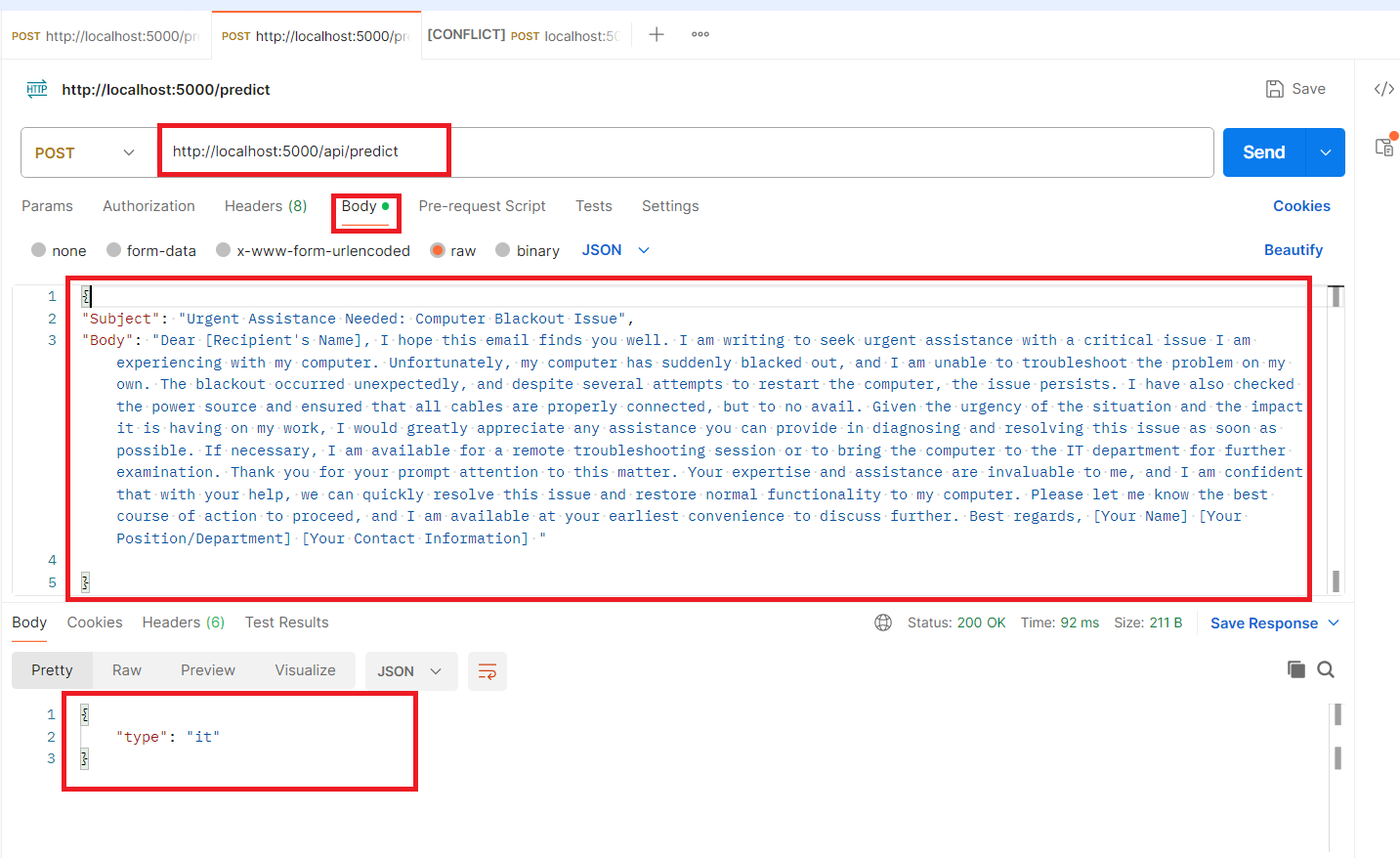
{

"Subject": "Urgent Assistance Needed: Computer Blackout Issue",

"Body": "Dear [Recipient's Name], I hope this email finds you well. I am writing to seek urgent assistance with a critical issue I am experiencing with my computer. Unfortunately, my computer has suddenly blacked out, and I am unable to troubleshoot the problem on my own. The blackout occurred unexpectedly, and despite several attempts to restart the computer, the issue persists. I have also checked the power source and ensured that all cables are properly connected, but to no avail. Given the urgency of the situation and the impact it is having on my work, I would greatly appreciate any assistance you can provide in diagnosing and resolving this issue as soon as possible. If necessary, I am available for a remote troubleshooting session or to bring the computer to the IT department for further examination. Thank you for your prompt attention to this matter. Your expertise and assistance are invaluable to me, and I am confident that with your help, we can quickly resolve this issue and restore normal functionality to my computer. Please let me know the best course of action to proceed, and I am available at your earliest convenience to discuss further. Best regards, [Your Name] [Your Position/Department] [Your Contact Information] "

}

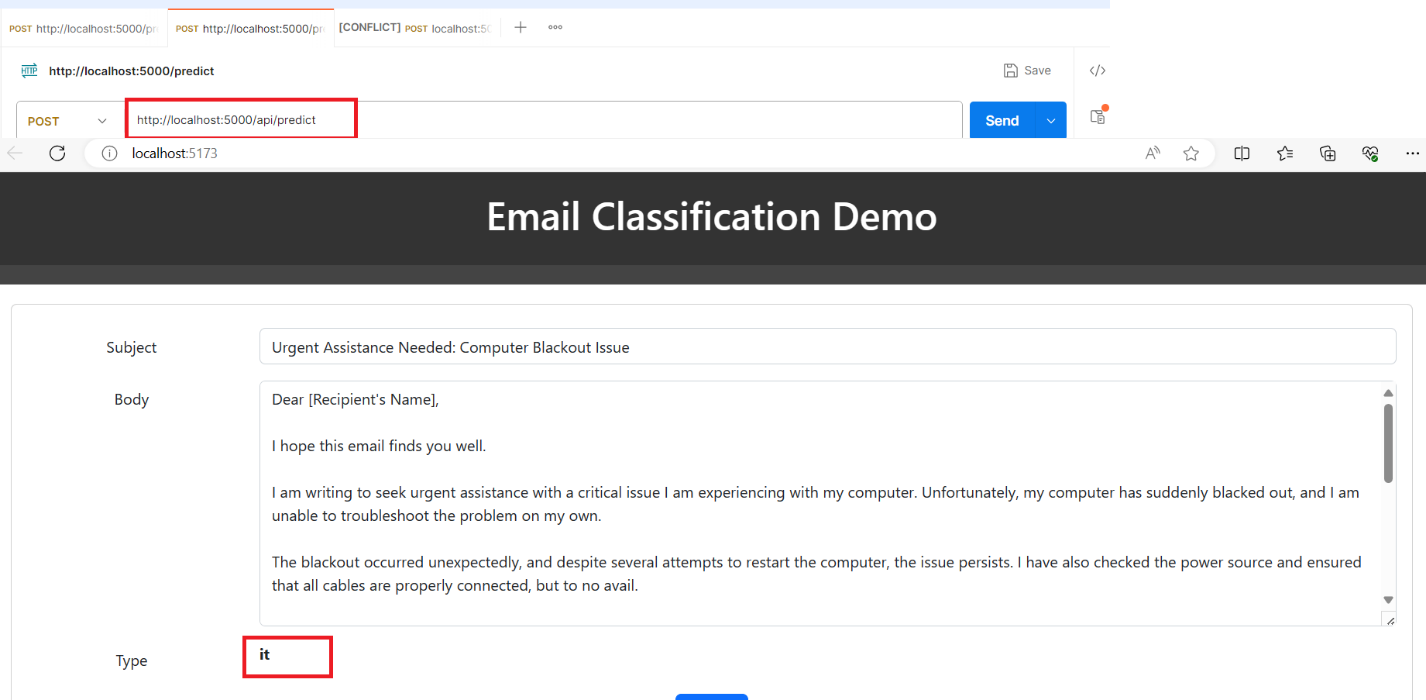
See the below screenshot for detail:



**10.3. Frontend Server:** check README.md to run the server

**A screenshot of a computer program

Description automatically generated**

****

1. **Limitation**

**12.1 Frontend UI**: Frontend application depends on the Backend to clarify email type. But the current dependency is hard code in the EmailForm.tsx file. The dependency should be configurable by using environment variables. The UI application can fetch backend information from the environment setting.

**12.2 Backend**: Communication between backend and Frontend should be encrypted by using HTTPS and Authentication and Authorization should be applied in backend to make sure that backend only accepts requests from specified users via frontend application. Implement a feedback loop where misclassified emails are flagged by users and used to improve the model in subsequent iterations.

**12.3 Model**: continuous monitoring the model performance and retrain on regular basic with information collected from feedback loop and changes in the email domain knowledge.

**Reference**:

Carnegie Mellon University. (n.d.). Enron Email Dataset. Retrieved from https://www.cs.cmu.edu/~./enron/