**📧 Email Intelligence System using Machine Learning and Telegram Bot**

An end-to-end system that:

* i) Automates fetching and categorizing emails from Gmail (Inbox and Spam),
* ii) Cleans and clusters them using **unsupervised learning**,
* iii) Manually maps clusters to labels,
* iv) Trains a **supervised spam classification model**,
* v) Sends real-time email alerts on **Telegram** with actionable buttons (Delete, Archive, Snooze, AI Reply),
* vi) Takes action on emails directly from Telegram using custom integrations.

**📁 Project Structure**

i) fetch\_emails.py: Fetch emails from Gmail Inbox and save to CSV

ii) fetch\_spam\_emails.py: Fetch emails from Gmail Spam and save to CSV

iii) SpamCleaner.py: Clean spam emails

iv) cluster\_emails.py: Cluster emails into 5 groups using unsupervised ML

v) config.py: Cluster to Label mapping

vi) find\_label.py: Assign labels to clustered emails

vii) merger\_csv.py: Merge cleaned and labeled datasets

viii)spam\_detector.py: Train spam detection model (Supervised ML)

ix) email\_notifier\_tele.py: Check new emails and send Telegram alerts

x) telegram\_api.py: Handle Telegram actions

xi) email\_utils.py: Email helper functions (delete, archive, etc.)

xii) clusters/ : Directory containing cluster CSVs

xiii)emails\_dataset.csv: Final labeled dataset for training

xiv) spam\_model.pkl: Trained spam classifier model

xv) vectorizer.pkl: Fitted vectorizer

xvi) README.md: Project documentation

xvii) License: This project is licensed under the [MIT License](./LICENSE).

**⚙️ Workflow Description**

**1. Fetch Inbox Emails**

fetch\_emails.py

* Connects with Gmail using IMAP
* Extracts email subject and content
* Saves data to emails\_data.csv

**2. Fetch Spam Emails**

fetch\_spam\_emails.py

* Fetches emails from Spam folder
* Saves to clusters/spam.csv

**3. Clean Spam Emails**

SpamCleaner.py

* Cleans unwanted characters, whitespace
* Saves to clusters/spam\_cleaned.csv
* Deletes original spam.csv

**4. Cluster Emails (Unsupervised Learning)**

cluster\_emails.py

* Applies TF-IDF vectorization and KMeans clustering (n=5)
* Splits emails\_data.csv into:
* cluster\_0.csv to cluster\_4.csv in clusters/ folder

**5. Manual Labeling**

config.py

* After analyzing each cluster, mapped them like:

CLUSTER\_LABEL\_MAP = {

0: "Career Update",

1: "Daily Updates",

2: "Banking",

3: "Newsletter",

4: "Promotions"

}

**6. Assign Labels**

find\_label.py

* Adds a label column to each cluster CSV based on the config mapping

**7. Merge All Emails**

merger\_csv.py

* Combines all cluster CSVs and spam\_cleaned.csv
* Final dataset saved as emails\_dataset.csv

**8. Train Spam Detection Model**

spam\_detector.py

* Trains **two supervised learning models**:
* One for spam detection
* One for category classification (optional)
* Saves:
* spam\_model.pkl (classifier)
* vectorizer.pkl (text transformer)

**9. Real-time Email Notifications via Telegram**

email\_notifier\_tele.py

* Polls inbox every 1 minute
* On new mail:
* Uses trained model to predict if spam
* Sends a Telegram message with:
* ✅ Subject
* 🏷️ Predicted Label
* 📊 Confidence Score
* 📝 Email Summary
* 🔘 Buttons: Delete, Archive, Snooze, AI Reply

**10. Telegram Interactions**

telegram\_api.py & email\_utils.py

* Handle user actions:
* Archive → Prompts label and moves email
* Delete → Moves email to trash
* AI Reply → Placeholder for future enhancement
* Snooze → Delays rechecking
* Hosted using **ngrok** for public Telegram webhook access

**Note: Please refer to the attached document on how to create a bot on Telegram.**

**🌐 Language Switching Support**

The bot supports multiple languages. You can change the interface language by simply sending:

/language

After this command, you will be presented with options such as:

* English 🇬🇧
* Hindi 🇮🇳

Once selected, all future messages from the bot will be in your chosen language.



**🛠️ Technologies Used**

* Python
* Gmail IMAP/SMTP
* scikit-learn (KMeans, Logistic Regression)
* Pandas / NumPy
* Telegram Bot API
* Ngrok (for webhook exposure)
* Joblib (for model serialization)

**📌 How to Run**

* Clone the repository:  
    
  git clone <https://github.com/your-username/email-intelligence-system.git>

cd email-intelligence-system

* Set up Gmail IMAP access and credentials securely.
* Run sequentially:  
    
  a) python fetch\_emails.py  
  

b) python fetch\_spam\_emails.py 

c)python SpamCleaner.py 

d)python cluster\_emails.py 

# Manually define cluster-labels in config.py

e)python find\_label.py



f)python merger\_csv.py 

g)python spam\_detector.py 

* Start the Telegram Notifier:  
  a) python email\_notifier\_tele.py



b) python telegram\_api.py

c) And start the ngrok: ngrok http {your\_port} on which telegram\_api is running.



Use Telegram to interact with your emails in real-time!

**📍 Future Enhancements**

* i) Implement better spam filtering using **deep learning**
* ii) Use **OAuth-based Gmail access** instead of basic authentication
* iii) Enable **"Email as a Database"** feature (Advanced querying/filtering like SQL)
* iv) Add **"Auto-Unsubscribe"** feature for newsletters
* v) Introduce **"Schedule Send" Reminders** for timely replies and actions
* vi) Integrate **AI-generated email replies** (GPT-based)
* vii) Build a **web dashboard** for visualizing emails and cluster stats

**📄 License**

This project is licensed under the MIT License.

**FLOWCHART** 