# AI-Powered Email Classification – Detailed Report

Prepared by: Luiz Fernando Alves & Romulo Gomes da Silva

Course: Engineering and Evaluating Artificial Intelligence Systems

Instructor: Dr. Abdul Razzaq

National College of Ireland

## 1. Introduction

This report details the development of an AI-based email classification system designed to automatically categorize IT service requests. The project utilizes Natural Language Processing (NLP) and Machine Learning (ML) techniques to classify emails based on their content, improving response efficiency and accuracy.

### 1.1 Problem Statement

Manual classification of support emails is time-consuming and prone to errors. This project aims to automate the process by implementing a supervised machine learning model that classifies emails based on predefined categories.

### 1.2 Goals and Objectives

- Translate multilingual emails into English for standardization.  
- Clean and preprocess the text for better ML performance.  
- Convert text data into numerical format using TF-IDF.  
- Train a machine learning model to classify emails based on their content.  
- Evaluate the model and improve its performance.

## 2. Data Preprocessing

### Steps performed:

1. \*\*Translation:\*\* Converted multilingual text into English.

2. \*\*Cleaning:\*\* Removed special characters, numbers, and punctuation.

3. \*\*Tokenization:\*\* Split text into individual words.

4. \*\*Stopword Removal:\*\* Removed common words that do not add meaning.

5. \*\*TF-IDF Vectorization:\*\* Converted text into numerical format.

### 2.1 Data Preprocessing Code

import pandas as pd  
import re  
import string  
import nltk  
from nltk.corpus import stopwords  
from nltk.tokenize import word\_tokenize  
from sklearn.feature\_extraction.text import TfidfVectorizer  
  
nltk.download('punkt')  
nltk.download('stopwords')  
  
# Load the cleaned dataset  
file\_path = "../data/AppGallery\_cleaned.csv"  
df = pd.read\_csv(file\_path)  
  
stop\_words = set(stopwords.words('english'))  
  
def preprocess\_text(text):  
 if pd.isna(text):  
 return ""  
 text = text.lower()  
 text = re.sub(r'\d+', '', text)  
 text = text.translate(str.maketrans("", "", string.punctuation))  
 words = word\_tokenize(text)  
 words = [word for word in words if word not in stop\_words]  
 return " ".join(words)  
  
df["Processed\_Text"] = df["Ticket Summary"].apply(preprocess\_text)  
  
df.to\_csv("../data/AppGallery\_preprocessed.csv", index=False)

## 3. Model Training

We trained a Naïve Bayes classification model using 'Type 2' as the target variable. This approach allows us to categorize emails into general classes (Problem/Fault, Suggestion, Others).

### 3.1 Model Training Code

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.preprocessing import LabelEncoder  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# Load the dataset  
file\_path = "../data/AppGallery\_final\_preprocessed.csv"  
df = pd.read\_csv(file\_path)  
  
df = df.dropna(subset=["Type 2"])  
label\_encoder = LabelEncoder()  
df["Type 2 Encoded"] = label\_encoder.fit\_transform(df["Type 2"])  
  
X = df["Processed\_Text"]  
y = df["Type 2 Encoded"]  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
vectorizer = TfidfVectorizer(max\_features=5000)  
X\_train\_tfidf = vectorizer.fit\_transform(X\_train)  
X\_test\_tfidf = vectorizer.transform(X\_test)  
  
model = MultinomialNB()  
model.fit(X\_train\_tfidf, y\_train)  
  
y\_pred = model.predict(X\_test\_tfidf)  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Model Accuracy: {accuracy:.4f}")  
print("Classification Report:")  
print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))

## 4. Model Evaluation

After training, the model was evaluated using accuracy and precision-recall metrics.

### \*\*Final Model Performance:\*\*

- \*\*Model Accuracy:\*\* 68%

- \*\*Category Breakdown:\*\*

- \*\*Problem/Fault:\*\* High recall (100%) but moderate precision (64%).

- \*\*Others:\*\* High precision (100%) but low recall (22%).

- \*\*Suggestion:\*\* Limited data (only 2 samples).

### 4.1 Model Evaluation Code

from sklearn.metrics import classification\_report  
  
print("Classification Report:")  
print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))

## 5. Model Improvement Strategies

To improve the model, we considered the following techniques:

1. \*\*Balancing the dataset\*\* (ensuring equal representation of all categories).

2. \*\*Trying alternative models:\*\* SVM, Random Forest, and BERT.

3. \*\*Hyperparameter tuning\*\* to improve Naïve Bayes performance.

## 6. Conclusion

This project successfully demonstrated how Natural Language Processing and Machine Learning can automate email classification for IT service requests. Future work includes testing additional models and optimizing hyperparameters to improve accuracy.