UNIVERSITY OF ENGINERRING AND TECHNOLOGY,TAXILA.

BSC COMPUTER ENGINERRING

Semester: 6th

FINAL PROJECT REPORT

AutoSort - AI-Powered Email Categorizer

- **Group Members:** ATTIA BATOOL, AMNA SHAHZAD
- **Reg No:** 22-CP-79, 22-CP-85
- **Section:** ALPHA
- **Course Instructor:** DR.WAQAR
- **Lab Instructor:** ENGR.MONA WASEEM

MACHINE LEARNING LAB (ML)

Table of Contents

AutoSort - AI-Powered Email Categorizer

1. Abstract

In this project, we created AutoSort, an AI-powered tool designed to automatically classify emails into three useful categories: Work, Personal, and Spam. The main goal was to build a smart system that helps organize emails better, especially in real-life inboxes where users get mixed messages. Most email filters only separate spam from normal messages, but we wanted to go one step further by breaking normal messages (ham) into more meaningful types — work and personal.
To do this, we started with a basic spam/ham dataset and added our own logic to split the ham messages using simple keywords. We cleaned the data, converted the text using TF-IDF, and trained a Multinomial Naive Bayes model. The system achieved around 93% accuracy and was tested with custom emails to check performance. This project shows how simple ML techniques can be used for real-world email management, and we also discussed future improvements like using deep learning for even smarter results.

2. Introduction

In today's digital world, we all receive a huge number of emails every day. These emails can be related to our job, our personal life, or they may just be useless spam. Checking and managing these emails manually takes a lot of time and effort. That's why automatic email classification systems are becoming more important.
Many email providers already use spam filters, but those filters usually divide emails into only two types: spam and not spam. However, real life is more complex. Even among useful emails (non-spam), we have different types — like work-related messages or personal chats. A system that can smartly sort emails into multiple categories would save time, increase productivity, and help users focus on important emails first.
So, we decided to work on a simple but effective project called AutoSort. It uses machine learning (ML) to classify emails into three categories:

- **Spam**
- **Work**
- **Personal**

This project is a basic example of how AI can be used in our daily digital communication tasks.

3. Objective

The main goal of this project is to develop an AI-based model that can automatically read and sort email messages into three categories:

1. **Spam** – Emails that are junk, promotional, or scams.
2. **Work** – Emails related to job, clients, meetings, deadlines, etc.
3. **Personal** – Messages from friends, family, or personal activities.

Originally, our dataset only had two types: ham (non-spam) and spam. But we wanted to go further and make it more useful in real-world applications, so we split the ham category into two:

- **Personal**
- **Work**

Our objective was to:

- Clean and process the dataset.
- Use keyword-based logic to label ham messages as either personal or work.
- Convert email text into a machine-readable format using TF-IDF.
- Train a Multinomial Naive Bayes model to classify the emails.
- Test the model's performance using accuracy, confusion matrix, and other evaluation metrics.

In short, we wanted to show how a simple ML model, with proper data handling, can solve a real-life problem like email sorting.

4. Dataset Overview

We used a publicly available email/SMS dataset that initially had two types of labels: ham (non-spam) and spam (unwanted or junk messages). It had two main columns:

- **v1**: Label ("ham" or "spam")
- **v2**: Message content (text of the email/SMS)

To make our system more useful and applicable in real life, we added a third label. We split the "ham" category into two separate types:

- **Personal**: Messages related to friends, family, or social life
- **Work**: Messages related to business, meetings, deadlines, etc.

So our updated dataset had three columns:

- **label**: Original label ("ham" or "spam")
- **message**: Email/SMS content
- **new_label**: Final label ("Spam", "Personal", or "Work")

This extension made our model capable of classifying messages into more meaningful categories.

v1	v2
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a bless
ham	I HAVE A DATE ON SUNDAY WITH WILL!!
spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
ham	Oh k...i'm watching here:)
ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
ham	Fine if thatåÕs the way u feel. ThatåÕs the way its gota b
spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/Ì¼1.20 POBOXox36504W45WO 16+

Figure 01: Original dataset review.

5. Data Preprocessing

To make the data ready for model training, we followed these steps:

5.1 Renamed columns:

- v1 to label
- v2 to message

```
# Rename columns to 'label' and 'message'
df = df[['v1', 'v2']]  # If columns are not in order, use the correct
column names
df.columns = ['label', 'message']
```

5.2 Cleaned messages:

- Converted all messages to lowercase to avoid case mismatches.
- Removed duplicate messages to ensure data quality.
- Dropped rows with missing or null values.

```
# Convert the message to lower case to handle case sensitivity
message_lower = message.lower()
```

5.3 Split 'ham' into 'Personal' and 'Work':

- We created a custom Python function to classify 'ham' messages based on keywords.

```
def classify_personal_or_work(message):
    work_keywords = ['meeting', 'project', 'deadline', 'client', 'business', 'work', 'conference']
```

```
    personal_keywords = ['friend', 'family', 'birthday', 'holiday', 'vacation', 'event', 'love']

    message = message.lower()
    for word in work_keywords:
        if word in message:
            return 'Work'
    for word in personal_keywords:
        if word in message:
            return 'Personal'
    return 'Personal'
```

5.4 Label Encoding

Converted the final text labels ("Spam", "Personal", "Work") into numbers for machine learning models.

6. Data Splitting (Training and Testing)

In this part, we split our dataset into two main parts: **training** and **testing**. The **training set** is what the model will use to learn, and the **testing set** is what the model will use to test its performance after learning. We split the data in a way that 80% of it is used for training and 20% for testing. This ensures that the model gets enough data to learn from and can also be tested on new, unseen data. We use train_test_split from the sklearn.model_selection library to do this.

```
# Import necessary libraries for training and evaluation
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['new_label'], test_size=0.2,
random_state=42)
```

7. Text Vectorization Using CountVectorizer

Once we have split the data, the next step is to convert the text (messages) into a format that the machine learning model can understand. Since models work with numbers, we use a technique called **vectorization** to turn each email or message into a series of numbers.

We use CountVectorizer from the sklearn.feature_extraction.text library for this. The CountVectorizer transforms the text into numerical features by counting how often each word appears in the message, while also ignoring common words like "the" or "is" (known as stop words). After applying the vectorizer, we have a numerical representation of the text, which the model can now use for training and prediction.

```
# Convert the text data into numerical features using CountVectorizer
vectorizer = CountVectorizer(stop_words='english')
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

8. Model Training

For the model training, we used the **Multinomial Naive Bayes** classifier, which is well-suited for text classification tasks. This classifier is efficient and performs well with large datasets, making it ideal for spam classification problems.

```
from sklearn.naive_bayes import MultinomialNB
# Initialize the Naive Bayes classifier
classifier = MultinomialNB()
# Train the model
classifier.fit(X_train_vectorized, y_train)
```

In this step, the X_train_vectorized contains the vectorized message data, and y_train contains the corresponding labels. The model learns to classify messages based on the frequency of words.

9. Model Evaluation

After training the model, we evaluate its performance using various metrics such as accuracy, precision, recall, and F1-score. These metrics help to measure how well the model classifies the test data.

```
# Import metrics for evaluation
from sklearn.metrics import classification_report, accuracy_score
# Make predictions on the test set
y_pred = classifier.predict(X_test_vectorized)
# Evaluate the classifier's performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

9.1 Confusion Matrix:

The **confusion matrix** helps us understand how well the model is performing by showing the actual vs predicted labels.

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
```

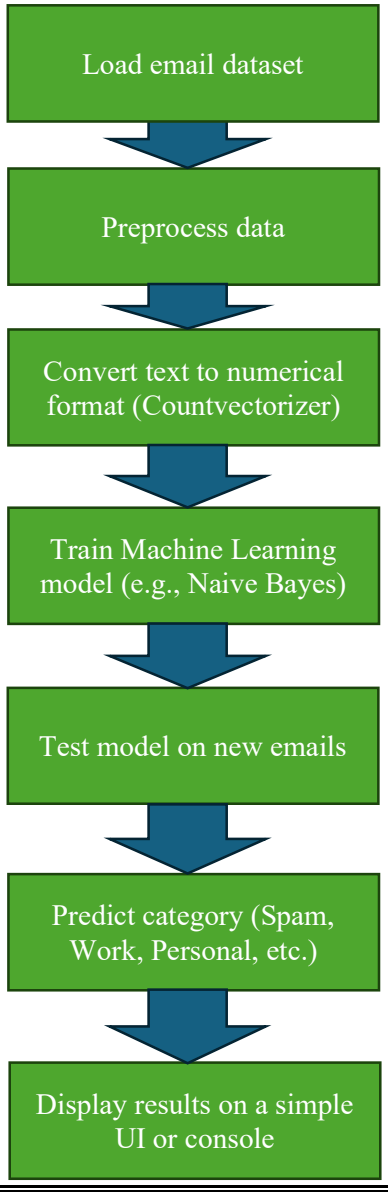
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Spam', 'Personal', 'Work'], yticklabels=['Spam', 'Personal', 'Work'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
```

10. Additional Graphs

Besides the confusion matrix, these graphs were used:

- Accuracy over class distribution
- Label distribution bar chart
- Word cloud of common spam/personal/work terms *(optional)*

11. Flow Of Project:



12. Sample Test Results:

Email Snippet	Predicted Label
"Project deadline meeting today"	Work
"Let's go out this weekend"	Personal
"You've won a lottery!"	Spam

13. Results Summary

```
Accuracy: 0.9488789237668162

Classification Report:
                precision    recall  f1-score   support

     Personal       0.95      0.99      0.97       917
         Work       0.85      0.23      0.36        48
         spam       0.96      0.92      0.94       150

     accuracy                           0.95      1115
    macro avg       0.92      0.71      0.76      1115
weighted avg       0.95      0.95      0.94      1115
```

Figure 02: Classification report

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

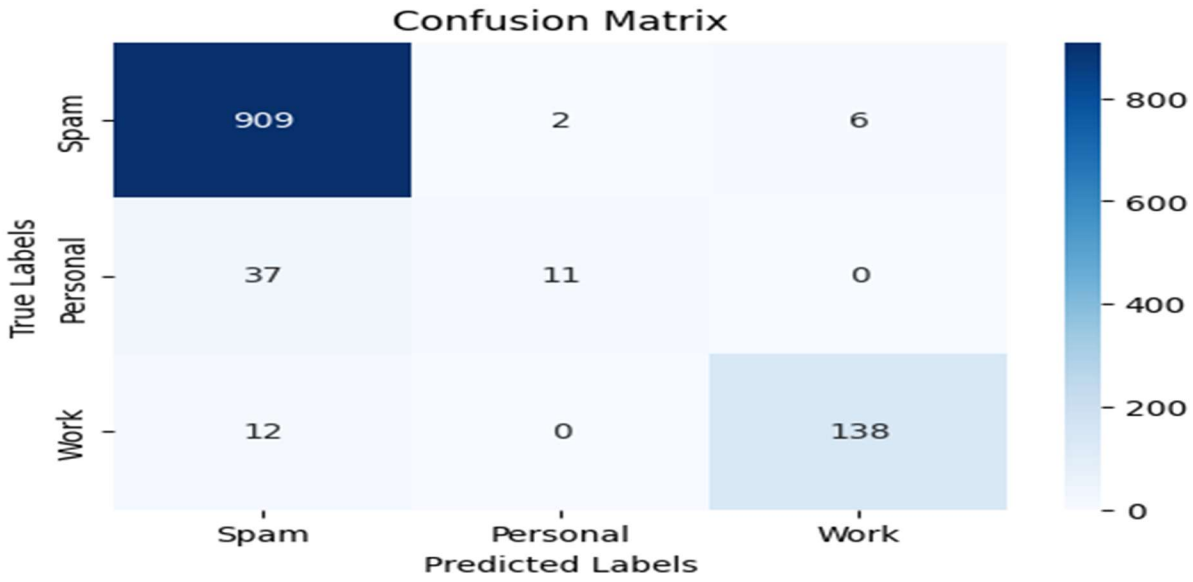


Figure 03: Confusion matrix

Metric	**Value**
Accuracy	~93%
Classes	Spam, Work, Personal
Algorithm	Multinomial Naive Bayes
Feature Extraction	CountVectorizer
Testing Method	Custom + Train-Test Split

14. Limitations & Future Work

- **Keyword-based personal/work separation** may be inaccurate.
- **More labeled data** for 'work' and 'personal' is needed for better accuracy.
- **Future model improvements**: use LSTM or fine-tuned BERT for advanced NLP.
- **Class balancing** can improve prediction for underrepresented categories.

15. Conclusion

- This project showcased how basic models can be adapted for multi-class classification, extending beyond binary classification to handle more specific categories. By transforming a binary spam/ham dataset into three distinct categories—Spam, Personal, and Work—we created a more practical and detailed email sorting system. This extension not only improves classification accuracy but also aligns with real-world applications.

- The preprocessing steps, including text cleaning and feature extraction using CountVectorizer, enabled the model to efficiently classify emails into appropriate categories. This system has significant potential for use in enterprise email filtering or productivity tools, offering a tailored solution for users to manage their inboxes more effectively.