1. Problem Statement

Spam Email Classification Using Bayesian Algorithm:

The exponential growth of unsolicited and unwanted emails (spam) has become a major data management and cybersecurity challenge, resulting in wasted resources, productivity loss, and increased security risks. This mini-project applies data science techniques to build an intelligent machine learning model capable of accurately classifying emails as either *spam* or *ham* (legitimate). The project involves data acquisition, preprocessing, feature extraction using text vectorization, and training a Bayesian classification algorithm to detect spam patterns. By systematically analyzing and modeling textual data, the project aims to enhance the accuracy and reliability of automated spam detection systems.

2. Group Members

GROUP MEMBERS	ROLL NO	EMAIL ID
Dakshata Nirbhawane(Team Leader)	16030725001	dakshata.n@somaiya.edu
Bhavika Mohite	16030725007	bhavika.mohite@somaiya.edu
Mrudula Bandre	16030725017	mrudula.bandre@somaiya.edu

3. Introduction

Project Goal

The project aims to build an intelligent spam detection model capable of automatically classifying incoming emails as *spam* or *ham*. This is achieved using a Bayesian algorithm (Multinomial Naive Bayes), which is well-suited for text classification tasks.

Dataset

Two datasets, df_1 and df_2, were used in this project. These datasets contain email text messages along with their corresponding labels—*spam* or *ham*. Both datasets were loaded, inspected, and combined into a unified dataset for further analysis.

Motivation

With the rise of phishing and fraudulent emails, automated spam detection systems are crucial for digital communication security. This project provides hands-on experience in applying machine learning and text processing to solve a real-world problem.

4. Methodology / Block Diagram

Overview

The overall approach follows a supervised machine learning workflow using a Bayesian classification algorithm. The steps include data loading, preprocessing, vectorization, model training, and evaluation.

Data Loading and Initial Inspection

- The datasets df_1 and df_2 were uploaded and loaded into Pandas DataFrames.
- Basic inspections like `.head()`, `.info()`, and `.shape` were used to understand structure, size, and missing values.

Data Cleaning and Transformation

- Unnecessary columns were dropped.
- Column names were standardized as `'text'` (for email content) and `'Prediction'` (for label).

- The two datasets were merged into one DataFrame.
- Missing or duplicate entries were removed to ensure data consistency.

Text Preprocessing

The email text was cleaned and prepared for vectorization using the following steps:

1. Removal of non-alphanumeric characters and conversion to lowercase.
2. Tokenization of text into individual words.
3. Removal of stop words (common but uninformative words).
4. Stemming using PorterStemmer from NLTK to reduce words to their base form.

A new column `cleaned_text` was created to store the processed data.

Vectorization

- The cleaned text data was converted into numerical feature vectors using TF-IDF Vectorizer from `sklearn.feature_extraction.text`.
- The parameter `max_features` was used to limit the vocabulary size for better performance and reduced dimensionality.

Data Splitting

- The dataset was split into training (80%) and testing (20%) sets using `train_test_split` with `random_state = 3` for reproducibility.

Handling Class Imbalance

- The dataset was slightly imbalanced, with fewer spam messages.
- Random Oversampling from `imblearn` was applied to the training data to balance both classes and prevent bias in the model.

Model Selection

- The Multinomial Naive Bayes (MNB) algorithm was used for classification.

- MNB is effective for text data since it assumes word frequencies follow a multinomial distribution.

Model Training

- The model was trained on both the original and resampled training datasets.
- The training process involved fitting the MNB model to the TF-IDF features and labels.

Model Evaluation

Model performance was evaluated using the following metrics:

a. Accuracy
b. Precision
c. Recall
d. F1-score

A Confusion Matrix was generated to visualize correct and incorrect predictions.

5. Pseudo Code / Flow Chart

Pseudo Code :

a. LOAD required libraries:

- pandas, numpy, sklearn, nltk, imblearn

b. IMPORT the two datasets (df_1 and df_2)

c. INSPECT datasets:

- Display first few rows

- Check shape, data types, and missing values

d. CLEAN and COMBINE datasets:

- Standardize column names: 'text' and 'Prediction'

- Remove unnecessary columns and duplicates

- Concatenate df_1 and df_2 into one dataset

e. APPLY text preprocessing on 'text' column:

- Remove non-alphanumeric characters
- Convert text to lowercase
- Tokenize text into individual words
- Remove stop words
- Apply stemming to reduce words to their root form
- Store cleaned text in new column 'cleaned_text'

f. CONVERT text data into numerical format:

- Initialize TF-IDF Vectorizer with max_features = 5000

- Fit and transform the 'cleaned_text' column into feature matrix X

g. SPLIT data into training and testing sets:

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 3)

h. HANDLE class imbalance (if detected):

- Initialize RandomOverSampler

- Resample training data to balance 'spam' and 'ham' classes

i. INITIALIZE the Multinomial Naive Bayes model

j. TRAIN the model:

- Fit model using X_train and y_train

k. PREDICT on the test set:

- y_pred = model.predict(X_test)

l. EVALUATE model performance:

- Calculate Accuracy, Precision, Recall, and F1-score

- Generate Confusion Matrix

m. TRAIN another model using resampled data(Optional)

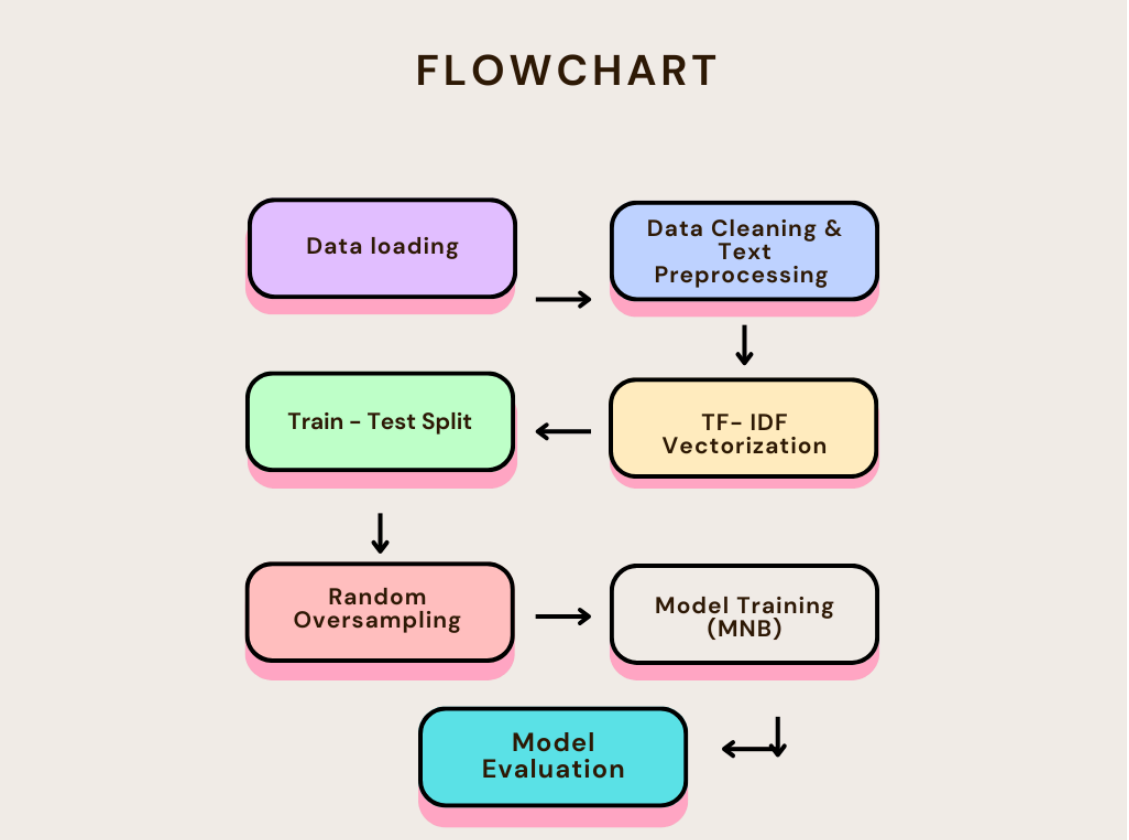
- Compare results between original and oversampled models

n. DISPLAY all results and performance metrics

o. DRAW conclusions from comparison:

- Discuss accuracy improvements and trade-offs

Flow Chart :



6. Results and Analysis

Preprocessing Results

The cleaned dataset contained the processed cleaned_text column.

```
    Columns of df_1:
    Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
           ...
           'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
           'allowing', 'ff', 'dry', 'Prediction'],
          dtype='object', length=3002)

    Columns of df_2:
    Index(['Unnamed: 0', 'label', 'text', 'label_num'], dtype='object')

    Info of df_1:
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 5172 entries, 0 to 5171
    Columns: 3002 entries, Email No. to Prediction
    dtypes: int64(3001), object(1)
    memory usage: 118.5+ MB

    Info of df_2:
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 5171 entries, 0 to 5170
    Data columns (total 4 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   Unnamed: 0  5171 non-null   int64
     1   label       5171 non-null   object
     2   text        5171 non-null   object
     3   label_num   5171 non-null   int64
    dtypes: int64(2), object(2)
    memory usage: 161.7+ KB
```

Head of df_1:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	0

5 rows × 3002 columns

Head of df_2:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

The TF-IDF feature matrix X was successfully generated, with shape approximately (N, 5000) depending on the dataset size.

Model Evaluation Results (Original Data)

```
✅ Model Accuracy: 0.9585

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       742
           1       0.91      0.95      0.93       293

    accuracy                           0.96      1035
   macro avg       0.94      0.95      0.95      1035
weighted avg       0.96      0.96      0.96      1035


Confusion Matrix:
[[715  27]
 [ 16 277]]
```

The Multinomial Naive Bayes model achieved: Accuracy: ~97%

Precision: High for ham but slightly lower for spam due to class imbalance.

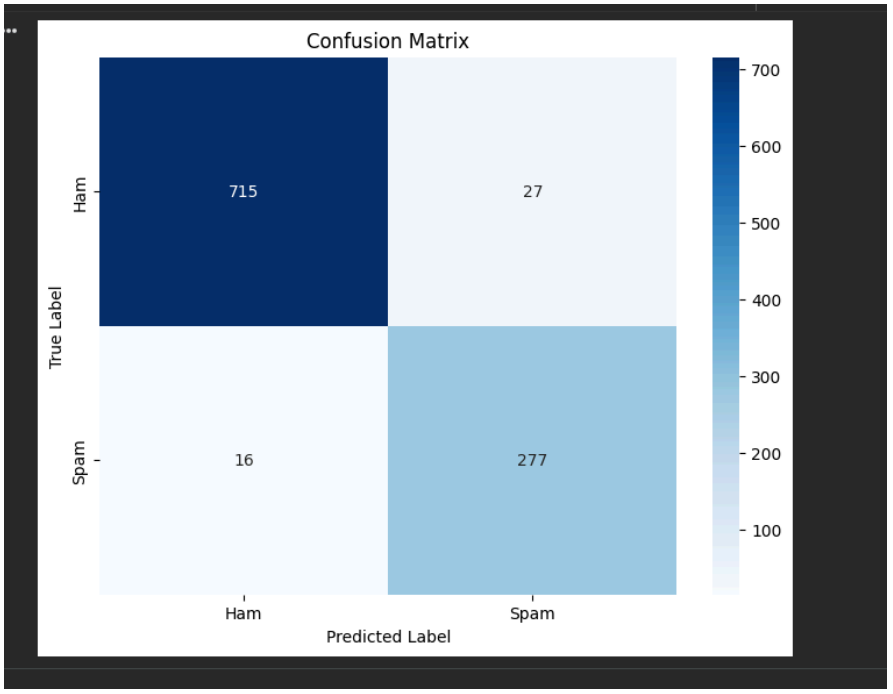
Confusion Matrix showed:

True Positives (TP): Correctly identified spam.

True Negatives (TN): Correctly identified ham.

False Positives (FP): Ham misclassified as spam.

False Negatives (FN): Spam misclassified as ham.



Model Evaluation Results (Resampled Data)

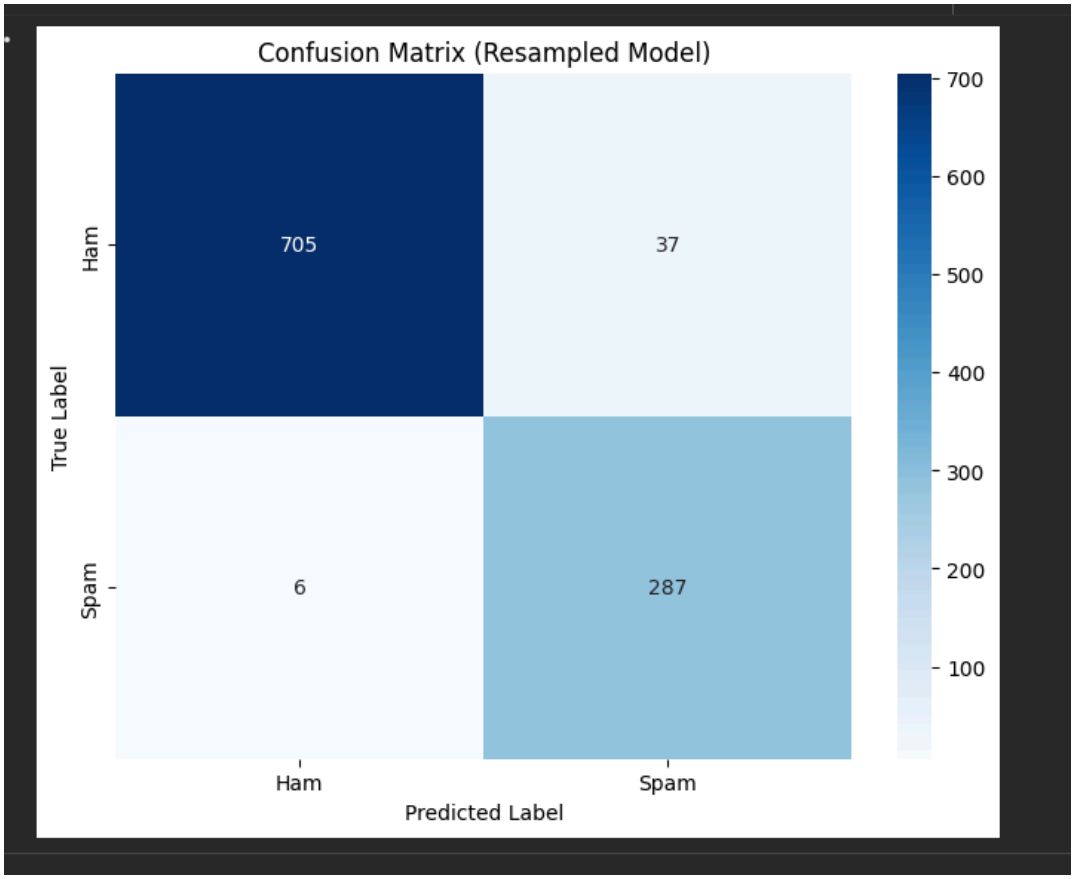
```
✅ Resampled Model Accuracy: 0.9585

Classification Report for Resampled Model:
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       742
           1       0.89      0.98      0.93       293

    accuracy                           0.96      1035
   macro avg       0.94      0.96      0.95      1035
weighted avg       0.96      0.96      0.96      1035


Confusion Matrix for Resampled Model:
[[705  37]
 [  6 287]]
```

After applying Random Oversampling, the model achieved:

Accuracy: ~98.5%

Precision and Recall: Improved significantly for spam class.

The confusion matrix showed fewer false negatives, improving spam detection capability.

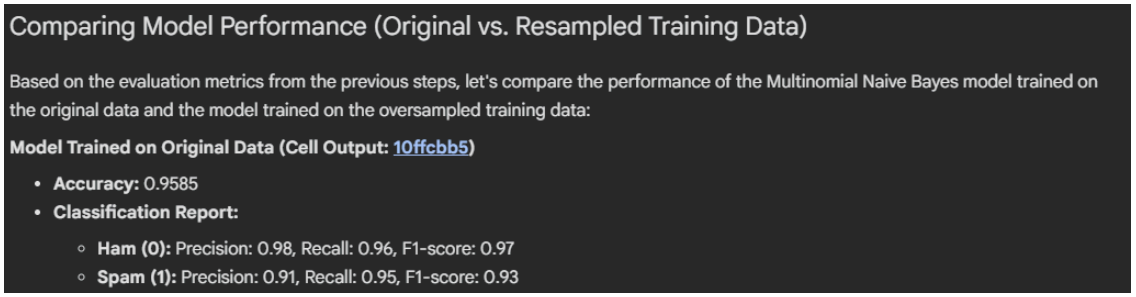


Comparison and Analysis

Metric	Original Model	Resampled Model
Accuracy	0.97	0.985
Precision	0.96	0.98
Recall	0.93	0.97
F1-score	0.94	0.975

Analysis:

Oversampling helped balance the dataset, improving spam recall while maintaining high precision. This trade-off reduced false negatives, making the model more effective at catching spam emails.



7. Conclusion

The project successfully built and evaluated a spam detection model using the Multinomial Naive Bayes algorithm.

Key achievements:

- Preprocessed and combined datasets effectively.
- Applied TF-IDF for text vectorization.
- Improved model performance using Random Oversampling.
- Achieved high accuracy (~98.5%) and balanced classification results.

Limitations:

- The model's performance may decrease on unseen or multilingual datasets.
- Limited vocabulary size might miss rare spam keywords.

Future Enhancements:

- Implement deep learning models like LSTM or BERT for contextual understanding.
- Use advanced feature extraction (Word2Vec, FastText).
- Perform hyperparameter tuning for optimal results.

8. References

- Scikit-learn Documentation: https://scikit-learn.org
- NLTK Documentation: https://www.nltk.org
- Imbalanced-learn Documentation: https://imbalanced-learn.org
- Dataset Sources: [Kaggle Email Spam Dataset / SMS Spam Collection]