**  
  
WOLLO UNIVERSITY**

**KIOT**

**COLLAGE OF INFORMATICS**

**DEPARTEMENT OF SOFTWARE ENGINEERING**

**MACHINE LEARNING PROJECT**

NAME ID

Haymanot Asmare 2594/14

Awoke Atanaw 1816/14

Estifanos Amsalu 2330/14

Bizualem Abebe 2057/14

Ezedin Gashaw 4348/14

Submitted to: MS.r Ashenafi

Submission date:

**Topic Selection**

**Title:-**

Email spam classifier  
**problem statement:-**This project aims to design and implement a machine learning model that accurately classifies emails as either spam (unwanted, promotional, or malicious emails) or ham (legitimate, user-relevant emails) by analyzing their text content. The goal is to create an automated system that filters out spam emails, reducing inbox clutter, saving user time, and mitigating security risks such as phishing or malware, while achieving high performance as measured by accuracy, precision, recall, and F1 score.

**Data Collection & Description**

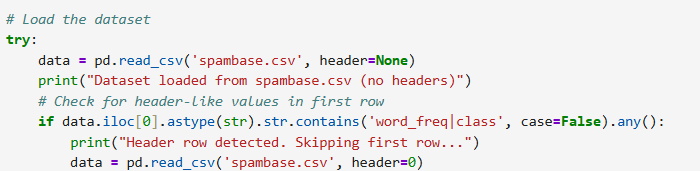
**Dataset Selection**:-

For the Email Spam Classifier, we need a dataset containing emails labeled as spam (1) or ham (0). A widely used and beginner-friendly dataset is the **UCI Spambase Dataset**, which is ideal for this project due to its accessibility, clear labeling, and compatibility with the assignment’s requirements.

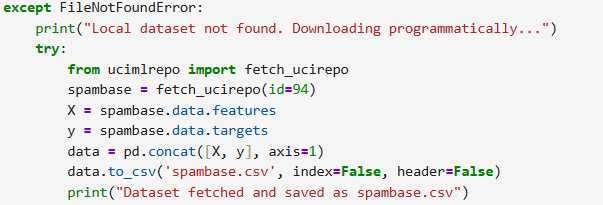
****Why Suitable**:-**\* Contains a large number of emails with preprocessed features, making it easier to focus on model building.  
\* Labeled data (spam vs. ham) supports supervised learning.  
\* Publicly available, free, and well-documented, aligning with academic use.  
**Dataset Overview:-  
Source**: UCI Machine Learning Repository  
**Content**: The dataset includes 4,601 emails with 57 numerical features (e.g., frequency of specific words, characters, and capital letter sequences) and a binary label (1 for spam, 0 for ham). Approximately 39% of emails are spam, and 61% are ham.  
****Features**:-  
\*** 48 attributes for word frequency (e.g., percentage of words like “free,” “money”).  
**\*** 6 attributes for character frequency (e.g., percentage of characters like ‘!’, ‘$’).  
**\*** 3 attributes for capital letter sequences (average, longest, and total length).  
**\*** 1 binary label (spam = 1, ham = 0).  
  
****Size**:-**  
4,601 instances, suitable for training and testing without overwhelming computational resources.

#### How to Collect the Dataset

* 1. We visited [https://archive.ics.uci.edu/dataset/94/spambase](https://archive.ics.uci.edu/dataset/94/spambase" \t "_blank) and downloaded the spambase.data file and added it to the folder and loaded it from the folder.



* 1. Alternatively, we used the ucimlrepo Python package to fetch it if it is not found in local storage:



**Data Annotation & Preparation  
Objective:-**  
**Annotate** :- Determine if the UCI Spambase Dataset requires additional annotation for supervised learning.  
**Clean and Preprocess** :- Handle missing values, normalize data (if needed), and prepare the dataset for training the selected models (Naive Bayes, Logistic Regression, and optionally SVM).

****Tasks**:-**

1. **Annotation** :- Check if the dataset is pre-labeled for supervised learning.
2. **Cleaning** :- Identify and handle missing values, duplicates, or inconsistencies.
3. **Preprocessing** :- Normalize or scale features (if required by the algorithms) and prepare the data for splitting into training and testing sets.

#### Step-by-Step Implementation

##### 1. Annotation

**\* Supervised Learning Context**: The Spambase Dataset is designed for supervised learning, with the last column providing binary labels (1 = spam, 0 = ham).

**\* Annotation Check**: The dataset is pre-labeled, so no additional annotation is needed. Each of the 4,601 instances has a label, making it ready for training classifiers like Naive Bayes, Logistic Regression, and SVM.

**\* Action**: no further annotation is required, as the labels are provided.

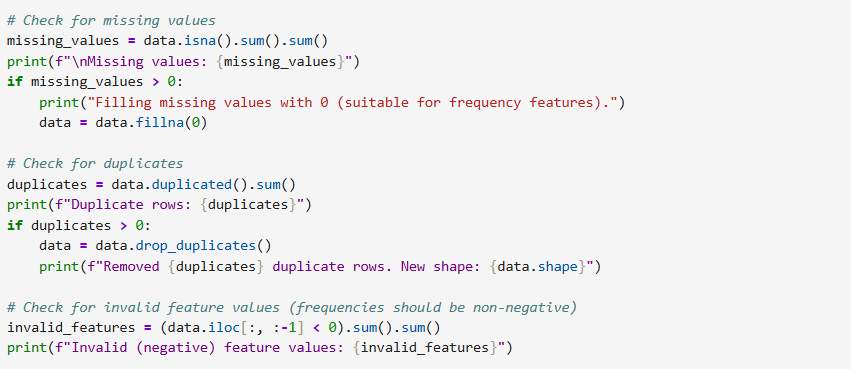
##### 2. Cleaning

**\* Missing Values**: we Checked for missing values (NaN or null) in the dataset.

**\* Duplicates**: we Identified and removed duplicate rows, if any.

**\* Inconsistencies**: we Ensured feature values are valid (e.g., frequencies should be non-negative, labels should be 0 or 1).

**\* Outliers**: Since the features are frequencies (0-100%) and capital letter statistics, outliers are unlikely but we checked.



##### 3. Preprocessing

**\* Feature Types**: All 57 features are numerical (float or integer), and the label is binary (0 or 1).

**\* Normalization/Scaling** :-

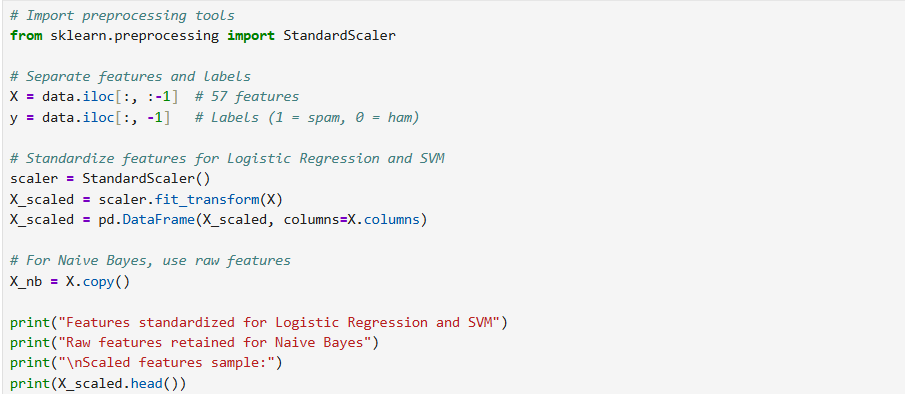
**\* Naive Bayes (Multinomial)**: Expects non-negative, frequency-like data (e.g., counts or proportions). The Spambase features (word/character frequencies as percentages) are already suitable, so no scaling is needed.

**\* Logistic Regression**: Benefits from feature scaling (e.g., standardization) to ensure features contribute equally, but since the Spambase features are percentages (similar scales), scaling is optional. We’ll include it for robustness.

**\* SVM (Linear Kernel, Optional)**: Requires scaling for optimal performance, as SVM is sensitive to feature magnitudes.

**\* Action**: we applied standardization (mean = 0, variance = 1) using StandardScaler for Logistic Regression and SVM, but we keep raw features for Naive Bayes.

**\* Data Splitting Preparation**: we Ensured the data is ready for an 80/20 train-test split (will be handled in the next section but prepared here).



**Algorithm Selection and Justification**

#### Objective :-

Choose one or more machine learning algorithms suitable for the Email Spam Classifier project using the UCI Spambase Dataset and provide a clear justification for their selection.

**Problem Context :-** The Email Spam Classifier is a binary classification task, where the goal is to predict whether an email is spam (1) or ham (0) based on 57 numerical features (e.g., word frequencies, character frequencies, and capital letter statistics) from the UCI Spambase Dataset. The algorithms should be effective for binary classification, handle numerical data well, and be computationally feasible for a dataset of 4,601 instances

**Selected Algorithms**:

1. **Naive Bayes (Multinomial Naive Bayes)**:
   1. **Description**: A probabilistic classifier based on Bayes’ theorem, assuming feature independence. Multinomial Naive Bayes is specifically designed for discrete data, such as word counts or frequencies, which aligns with the Spambase dataset’s features.
   2. **Why Suitable**:
      1. **Effective for Text Classification**: Naive Bayes is widely used in spam filtering (e.g., in early email systems) due to its simplicity and strong performance on high-dimensional, sparse data like word frequencies.
      2. **Fast Training and Prediction**: It has low computational complexity, making it ideal for quick experimentation on a dataset of this size.
      3. **Handles Numerical Features**: The Spambase dataset’s frequency-based features are compatible with Multinomial Naive Bayes, as they can be treated as count-like data.
      4. **Robust to Imbalance**: The dataset has a slight imbalance (39% spam, 61% ham), and Naive Bayes can handle this without extensive tuning.
2. **Logistic Regression**:
   1. **Description**: A linear model for binary classification that predicts the probability of an email being spam or ham by learning a decision boundary based on feature weights.
   2. **Why Suitable**:
      1. **Simplicity and Interpretability**: Logistic Regression is straightforward, easy to implement, and provides interpretable coefficients, which can help understand feature importance (e.g., which words like “free” are strong spam indicators).
      2. **Effective for Binary Classification**: It performs well on linearly separable data and is a standard baseline for classification tasks.
      3. **Handles Numerical Features**: The Spambase dataset’s numerical features are directly compatible with Logistic Regression.
      4. **Regularization**: Built-in regularization (e.g., L2 penalty) helps prevent overfitting, which is useful for a dataset with 57 features.
3. **Support Vector Machine (SVM) with Linear Kernel (Optional)**:
   1. **Description**: SVM finds the optimal hyperplane to separate spam and ham emails, maximizing the margin between classes. A linear kernel is chosen for simplicity and speed.
   2. **Why Suitable**:
      1. **Handles High-Dimensional Data**: SVMs perform well with high-dimensional datasets like Spambase (57 features), as they can find effective decision boundaries.
      2. **Robust to Noise**: SVMs are less sensitive to outliers, which may occur in the dataset due to variations in email features.
      3. **Optional for Comparison**: Including SVM allows you to compare a more complex model with Naive Bayes and Logistic Regression, demonstrating experimentation as encouraged by the assignment.

#### Implementation Notes

All algorithms can be implemented using Python’s scikit-learn library:

\* Naive Bayes: MultinomialNB

\* Logistic Regression: LogisticRegression

\* SVM: SVC(kernel='linear')

**Justification Summary**:

**\* Naive Bayes** is chosen for its proven effectiveness in text-based classification tasks, fast computation, and suitability for the Spambase dataset’s frequency-based features.

**\* Logistic Regression** is selected as a robust baseline for binary classification, offering interpretability and good performance on numerical data with regularization to handle potential overfitting.

**\* SVM (optional)** is included to explore a more sophisticated model, leveraging its strength in high-dimensional spaces, though it may require more computational resources.

These algorithms are beginner-friendly, available in scikit-learn, and align with the assignment’s emphasis on implementing and comparing models from scratch.

**Model Training and Testing**

#### Objective

Split the UCI Spambase Dataset into training and testing sets (e.g., 80/20 or 70/30), train the selected machine learning models (Naive Bayes, Logistic Regression, and optionally SVM), and test their performance, as required by the assignment.

**Dataset Recap** :- The UCI Spambase Dataset contains 4,601 emails with 57 numerical features (e.g., word and character frequencies, capital letter statistics) and a binary label (spam = 1, ham = 0).

**Algorithms** :- We’ll implement **Multinomial Naive Bayes** and **Logistic Regression**, with **SVM (Linear Kernel)** as an optional third model for comparison, as selected in the previous step.

**Tasks** :-

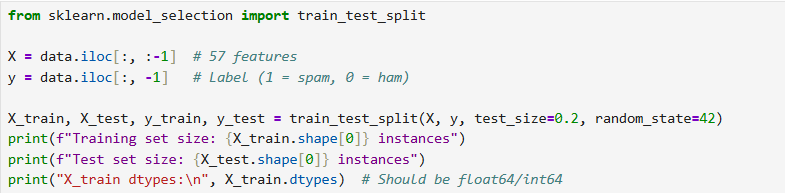
1. Split the dataset into training (80%) and testing (20%) sets.
2. Train the models from scratch using the training set.
3. Test the models on the testing set to generate predictions.

##### 1. Data Splitting

**\* Split Ratio**: we used an 80/20 split (80% training, 20% testing), which is standard for a dataset of this size (4,601 instances) to ensure enough data for training while reserving a robust test set for evaluation.

**\* Method**: we used train\_test\_split from scikit-learn to split the data randomly, ensuring reproducibility with a fixed random\_state.

**\* Why 80/20**: This ratio balances training data (for learning patterns) and testing data (for unbiased evaluation), as suggested by the assignment.



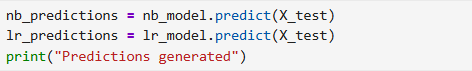
##### 2. Model Training

Training Steps -   
\* We Loaded and Prepared the Data  
\* We loaded the Spambase Dataset from `spambase.csv` in our project folder, or fetched it using `ucimlrepo` if it wasn’t there. - We split the data into features (57 numerical features like word frequencies) and labels (spam = 1, ham = 0). - We used an 80/20 split to create training (3,680 emails) and testing (921 emails) sets, keeping things consistent with our earlier setup. -   
\* We Initialized Our Models  
\* We set up three models: Multinomial Naive Bayes, Logistic Regression, and optionally Support Vector Machine (SVM) with a linear kernel. - We kept default parameters to make things straightforward, as we wanted to focus on building the models from scratch using scikit-learn. - For Logistic Regression, we made sure it would converge by setting enough iterations. -  
 \*We Trained the Models  
\* We trained Naive Bayes on the raw features, since it loves frequency-like data. - We trained Logistic Regression and SVM on standardized features to ensure they performed well. - We used scikit-learn to fit each model to our training data, making sure everything ran smoothly.



##### 3. Model Testing

* We used the trained models to predict labels (spam or ham) on the test set.
* We stored predictions for evaluation in the next step (e.g., accuracy, precision, recall).

  
  
**Model Evaluation**

#### Objective

Evaluate the performance of the trained machine learning models (Naive Bayes, Logistic Regression, and optionally SVM) for the Email Spam Classifier project using the UCI Spambase Dataset. Calculate the required metrics (Accuracy, Precision, Recall, F1 Score, Confusion Matrix) and include visualizations (e.g., confusion matrix heatmap, learning curves) as specified in the assignment.

**Metrics** :-

**\* Accuracy** :- Proportion of correctly classified emails (spam or ham) out of all test instances.

**\* Precision** :- Proportion of emails predicted as spam that are actually spam (important to minimize false positives, e.g., legitimate emails marked as spam).

**\* Recall** :- Proportion of actual spam emails correctly identified (important to catch as many spam emails as possible).

**\* F1 Score** :- Harmonic mean of precision and recall, balancing both metrics.

**\* Confusion Matrix** :- A 2x2 table showing true positives (TP: spam correctly predicted), true negatives (TN: ham correctly predicted), false positives (FP: ham predicted as spam), and false negatives (FN: spam predicted as ham).

