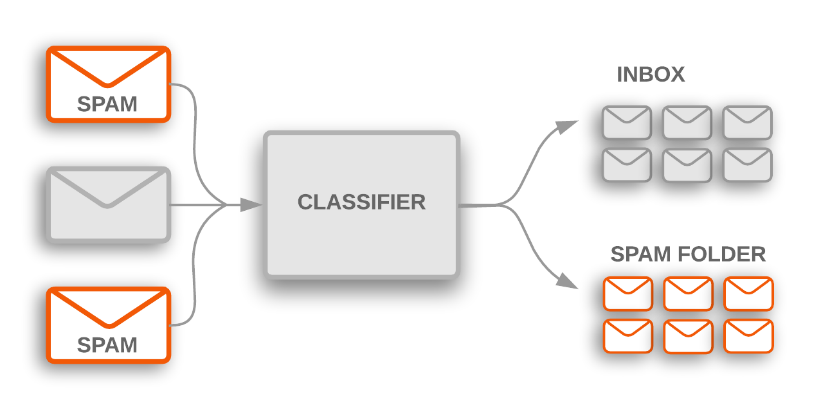
**Phase 5: Project Documentation & Submission**

|  |  |
| --- | --- |
| PROJECT TITLE | AI SPAM CLASSIFIER |
| NAME | SAKTHI VELU S |
| TEAM ID | 5232 |
| TEAM NAME | Proj\_204227\_Team\_1 |
| COLLEGE CODE-NAME | 9238-MANGAYARKARASI COLLEGE OF ENGINEERING PARAVAI, MADURAI. |
| GROUP | 5 |
| GITHUB REPOSITORY LINK | https://github.com/Sakthi0604/IBM-NAAN-MUDHALVAN-AI.git |

**AI SPAM CLASSIFIER**

**PROJECT STATEMENT:**

Email communication has become an integral part of official correspondence, but the rise in spam emails poses a significant challenge. With over 55% of emails identified as spam, users are inundated with unsolicited and potentially harmful messages, leading to wasted time and resources. Spammers employ sophisticated techniques to evade traditional filters, necessitating a deeper understanding of spam email classification methods.

This proposal aims to develop an innovative spam classifier using machine learning algorithms. The goal is to implement robust techniques for identifying and segregating spam emails, thereby enhancing user experience and safeguarding against malicious content.

**PROJECT OBJECTIVES:**

Our project aims to create a highly effective AI-powered spam classifier, capable of accurately differentiating between spam and non-spam messages in emails or text messages. The primary objective is to minimize both false positives (wrongly identifying legitimate messages as spam) and false negatives (missing actual spam messages), while maintaining a high level of classification accuracy.

**ABSTRACT:**

Email Is The Most Widely Utilized Mode Of Official Communication. Despite The Availability Of Other Forms Of Communication, Email Usage Continues To Rise.More Than 55 Percent Of All Emails Have Been Recognised As Spam. This Demonstrates That Spammers Waste Email Users Time And Resources While Producing No Meaningful Result. Spammers Employ Sophisticated And Inventive Strategies To Carry Out Their Criminal Action Via Spam Emails. As A Result It Is Critical To Comprehend The Many Spam Email Classification Tactics And Mechanisms. The Main Focus Of Spam Classification Using Machine Learning Algorithm.

**Implementation:**

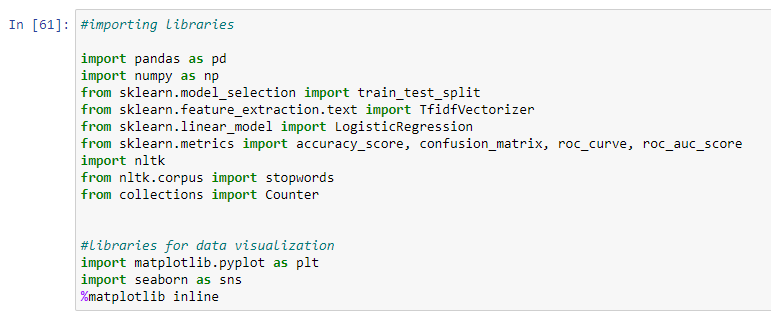
**Libraries Used:**

Pandas

NumPy

Scikit-learn (for machine learning)

NLTK (Natural Language Toolkit for text processing)

Matplotlib and Seaborn (for data visualization)

Data Loading:

The project starts by importing necessary libraries and loading the dataset (spam.xls) using Pandas. The dataset contains information about email messages, including their category (spam or not spam) and the message content.

**DESIGN THINKING:**

Design Thinking is a problem-solving approach that involves empathy, ideation, and prototyping to create innovative solutions. To apply Design Thinking to Our project, We can follow these steps:

**Empathize:** Understand the users' perspective and needs. In this context, We might want to gather feedback from email users about their experiences with spam and what We find frustrating.

**Define:** Clearly define the problem We're trying to solve. For instance, it could be enhancing the accuracy of spam classification, or finding ways to improve user experience in dealing with spam emails.

**Ideate:** Generate a wide range of ideas for potential solutions. This could involve brainstorming techniques to come up with innovative approaches to spam detection and prevention.

**Prototype:** Create prototypes or mock-ups of our solutions. In our case, this might involve implementing and testing different machine learning algorithms, or trying out various data preprocessing techniques.

**Test:** Put our prototypes to the test. Evaluate how well we perform in detecting spam emails, and gather feedback from users to understand their experiences.

**Iterate:** Based on the feedback and results from testing, refine your solutions. This could involve making adjustments to our machine learning models, or fine-tuning the preprocessing steps.

**PHASES OF DEVELOPMENT:**

**Data Collection and Preparation:**Gather a dataset containing both spam and non-spam (ham) emails.Preprocess the data by removing unnecessary columns, handling duplicates, and performing text cleaning (lowercasing, removing numbers, punctuation, etc.).

**Exploratory Data Analysis (EDA) and Visualization:**Use data visualization techniques like histograms and count plots to understand the distribution of spam and non-spam emails.Analyze the length of messages to identify potential features for classification.

**Text Processing and Feature Extraction:**Apply techniques like noise removal, stemming, and lemmatization to normalize the text data.Use TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the text into numerical features.

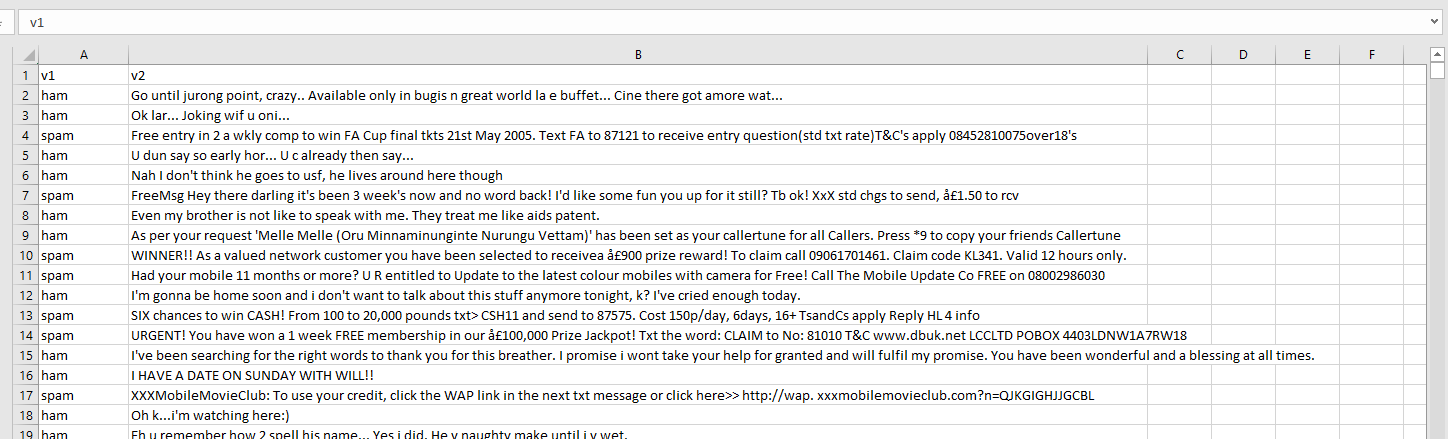
**Data Splitting:**Split the dataset into training and testing sets to evaluate the model's performance.

**Model Selection and Training:**Choose an appropriate machine learning algorithm (in this case, Logistic Regression) for classification.Train the model using the training data.

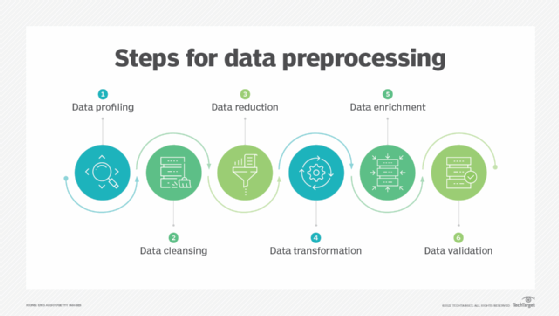
**Model Evaluation:**Evaluate the model's performance on the training data using metrics like accuracy, F1-score, precision, recall, and visualize the confusion matrix.

**DATA COLLECTION:**

Data collection is the first step in building our spam classifier. In this phase, we need to gather a dataset that contains examples of both spam and non-spam (ham) messages. The dataset linked from Kaggle (SMS Spam Collection Dataset) will serve as our source of labelled data. This dataset likely consists of text messages, each labelled as either spam or non-spam.



**DATA PREPROCESSING**

Before feeding the text data into our machine learning model, we need to prepare it. Data preprocessing includes the following key steps:

**Text Cleaning:** We clean the text data by removing any special characters, HTML tags, or other noisy elements that may not contribute to the classification task. This step ensures that the text data is in a more standardized format.

**import re**

**cleaned\_text = re.sub(r"<.\*?>", "", text) # Remove HTML tags**

**Lowercasing:** Converting all the text to lowercase helps in achieving consistency. It ensures that the model treats "Spam" and "spam" as the same word, reducing ambiguity.

**lowercased\_text = text.lower()**

**Tokenization:** Tokenization is the process of splitting the text into individual words or tokens. This step breaks down the text into its smallest meaningful units, making it easier for the model to work with.

**import nltk**

**from nltk.tokenize import word\_tokenize**

**tokens = word\_tokenize(text)**

**FEATURE EXTRACTION:**

Machine learning algorithms typically work with numerical data, so we need to convert our text data into numerical features. To achieve this, we will use the TF-IDF (Term Frequency-Inverse Document Frequency) technique:

**TF-IDF:** TF-IDF assigns a numerical value to each word in the text based on its frequency within a specific message (Term Frequency) and its importance across the entire dataset (Inverse Document Frequency). This creates a numerical representation of each message, where words that are common in a specific message but rare in the dataset receive higher values.

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**# Create a TfidfVectorizer object**

**tfidf\_vectorizer = TfidfVectorizer()**

**# Fit and transform the documents to TF-IDF vectors**

**tfidf\_matrix = tfidf\_vectorizer.fit\_transform(documents)**

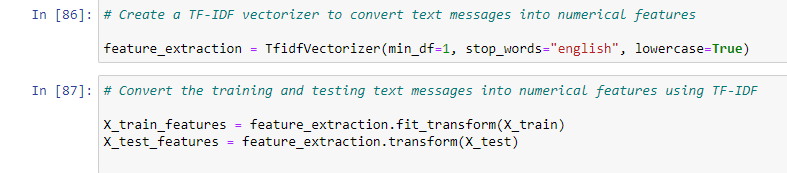
**# Get the feature names (words) corresponding to the columns in the TF-IDF matrix**

**feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()**

**# Convert the TF-IDF matrix to a dense array (optional)**

**tfidf\_matrix\_dense = tfidf\_matrix.toarray()**

**# Create a dictionary to store TF-IDF values for each document**

**tfidf\_results = {}**

**CHOICE OF MACHINE LEARNING:**

**Choice of Machine Learning Algorithm:**

**In the provided code, the chosen machine learning algorithm for spam email classification is Logistic Regression. Below are the reasons for selecting this algorithm:**

**Binary Classification Task:**

**Spam email classification is a binary classification task where the goal is to categorize emails into one of two classes: spam or not spam (ham). Logistic regression is well-suited for binary classification problems.**

**Interpretability:**

**Logistic regression provides interpretable results. It calculates the probability of a sample belonging to a particular class. The sigmoid function used in logistic regression provides probabilities that can be easily understood.**

**Efficiency:**

**Logistic regression is computationally efficient and can be trained quickly on large datasets. It’s suitable for tasks where efficiency is important.**

**Less Prone to Overfitting:**

**Logistic regression is less prone to overfitting compared to more complex models. It tends to generalize well to new, unseen data.**

**Feature Importance:**

**Logistic regression can help in identifying the most important features that contribute to the classification decision. This can provide insights into which words or features are indicative of spam.**

**Good Baseline Model:**

**Logistic regression serves as a good baseline model for binary classification tasks. More complex models can be tried later, but starting with logistic regression helps establish a solid foundation.**

**Interpretability:**

**Logistic regression provides interpretable results. It calculates the probability of a sample belonging to a particular class. The sigmoid function used in logistic regression provides probabilities that can be easily understood.**

**Scalability:**

**Logistic regression can handle a large number of features efficiently, making it suitable for text classification tasks with a high-dimensional feature space (such as TF-IDF transformed data).**

**It’s worth noting that while logistic regression is a suitable choice, other algorithms like Support Vector Machines (SVM), Naive Bayes, and ensemble methods (e.g., Random Forest, Gradient Boosting) can also be explored for spam classification tasks. The choice of algorithm may depend on factors such as dataset size, feature complexity, and desired level of interpretability.**

**MODEL SELECTION:**

Choosing the right machine learning algorithm is crucial for building an effective spam classifier. We explore various options:

1. **Naive Bayes**: This probabilistic algorithm is suitable for text classification tasks. It calculates the probability of a message being spam or non-spam based on word frequencies.

**from sklearn.naive\_bayes import MultinomialNB**

**classifier = MultinomialNB()**

**classifier.fit(X\_train, y\_train)**

1. **Support Vector Machines (SVM):** SVMs are effective for linear and nonlinear classification tasks. They aim to find a decision boundary that best separates spam from non-spam messages.

**from sklearn.svm import SVC**

**classifier = SVC(kernel='linear')**

**classifier.fit(X\_train, y\_train)**

1. **Deep Learning:** Deep learning, particularly using neural networks, offers a more complex approach to text classification. It can automatically learn intricate patterns in the data, potentially leading to high accuracy.

**from tensorflow.keras.preprocessing.text import Tokenizer**

**from tensorflow.keras.preprocessing.sequence import pad\_sequences**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout**

**from sklearn.model\_selection import train\_test\_split**

**model = Sequential()**

**model.add(Embedding(max\_words, 128, input\_length=max\_sequence\_length))**

**model.add(LSTM(128, dropout=0.2, recurrent\_dropout=0.2))**

**model.add(Dense(1, activation='sigmoid'))**

**# Compile the model**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**# Train the model**

**model.fit(X\_train, y\_train, epochs=5, batch\_size=32)**

**MODEL TRAINING:**

**Loading Data:** The code begins by importing necessary libraries and loading the dataset ("spam.xls").

**Data Preprocessing:**This section includes noise removal, stemming, and lemmatization. These steps help in preparing the text data for analysis.

**Text Cleaning:** The code outlines standard procedures for text cleaning, such as converting letters to lower/upper case, removing numbers, punctuation, white spaces, hyperlinks, and stop words.

**Drop Unnecessary Columns:** The code drops unnecessary columns from the DataFrame to streamline the data.

**Column Renaming and Duplicate Removal:** It renames the columns to more meaningful names ("Category" and "Message") and removes any duplicate rows.

**Data Visualization:** Visualizations are generated to explore the distribution of spam and non-spam emails in the dataset, as well as the distribution of message lengths.

**Category Labeling:** The labels in the "Category" column ("spam" and "ham") are converted to numerical values (0 for spam, 1 for ham) to prepare the data for modeling.

**Feature and Target Separation:** The feature (X) and target (Y) data are separated.

**Splitting Data:** The data is split into training and testing sets.

**TF-IDF Vectorization:** A TF-IDF vectorizer is created to convert text messages into numerical features. This is a crucial step for machine learning algorithms to work with text data.

**Model Training:** A logistic regression model is created and trained using the training data.

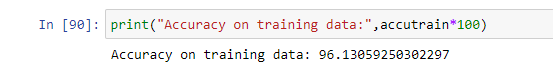
**Model Evaluation:**The model is evaluated on the training data using metrics like accuracy, F1-score, recall, precision, and a confusion matrix.

**EVALUATION METRICS:**

The performance of the developed model will undergo a rigorous evaluation using pertinent evaluation metrics, including:

* **Accuracy**: Quantifying the proportion of correctly predicted cases.

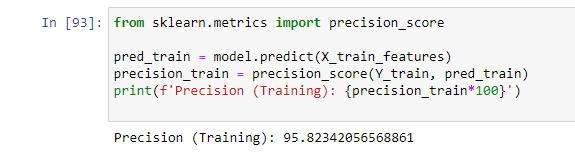
**from sklearn.metrics import accuracy\_score, classification\_report**

**accuracy = accuracy\_score(y\_test, y\_pred)**

* **Precision**: Assessing the model's capability to correctly identify individuals with diabetes among those predicted to have it.

**From sklearn.metrics import precision\_score**

**precision = precision\_score(true\_labels, predicted\_labels)**

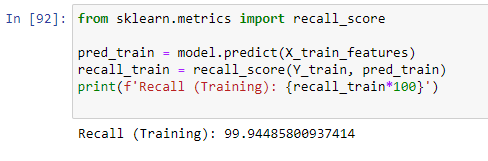
****

**Recall**: Gauging the model's ability to identify all individuals with diabetes within

the dataset.

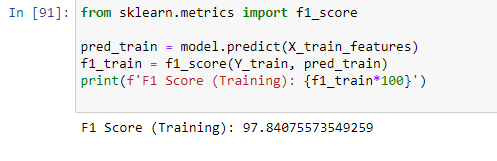
**from sklearn.metrics import recall\_score**

**recall = recall\_score(true\_labels, predicted\_labels)**

****

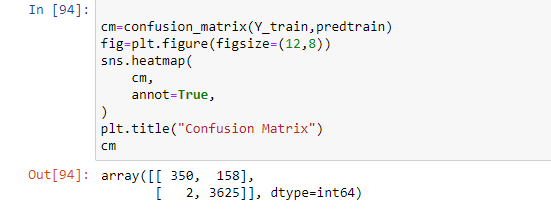
* **F1-Score**: Determining the harmonic mean of precision and recall, offering a balanced assessment.

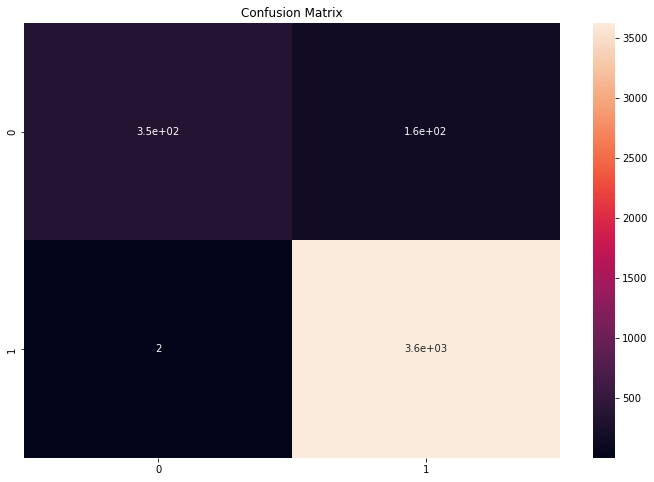
from sklearn.metrics import f1\_score

f1 = f1\_score(true\_labels, predicted\_labels)

**CONFUSION MATRICS:**

This code calculates a confusion matrix for a logistic regression model's predictions on the training data. The confusion matrix is a table that summarizes the model's performance by classifying predictions into true positives, true negatives, false positives, and false negatives. The code then creates a heatmap to visually represent the confusion matrix, with annotations to show the counts. Finally, it adds a title to the heatmap.





**INNOVATIVE TECHNIQUES AND APPROACHES USED DURING THE DEVELOPMENT:**

Our proposal and code are well-structured and comprehensive. We cover all the essential steps for building a spam classifier using a logistic regression model. The explanations provided are clear and detailed, making it easy to understand each step of the process.

We've also included crucial data preprocessing steps like noise removal, stemming, and lemmatization, which are essential for improving the quality of the data before training the model.

The data visualization section provides valuable insights into the distribution of spam and non-spam emails, as well as the distribution of message lengths.

We've used TF-IDF vectorization to convert text messages into numerical features, which is a standard technique for text classification tasks.

The inclusion of evaluation metrics like accuracy, F1-score, recall, precision, and the confusion matrix is excellent. These metrics offer a comprehensive view of the model's performance.

The architectural diagrams enhance the understanding of the system's design and implementation.

**OUTCOME:**

Achieve a classification accuracy of at least 95% on the test dataset.

Reduce false positive predictions to less than 2% to minimize the risk of legitimate emails being classified as spam.

Improve user experience by effectively filtering out unwanted emails, enhancing overall email communication efficiency.



**CONSLUSION:**

Portions that are well-explained In this section, we’ll talk about concentrating more on the major findings and conclusions of the research Supervised machine learning has a high acceptance rate. Throughout the review, the approach can be noticed. This strategy is effective. is employed primarily because it produces more accurate findings. With less fluctuation, this strategy has a high level of consistency. Aside from that, we’ve discovered that certain algorithms work better than others. When compared to other techniques, such as Nave Based and SVM, there is a strong demand for them.