**Building A Smarter Ai Spam Classifier**

**Phase 5**

**PROBLEM STATEMENT**:

Unlike emails, which have a variety of large datasets available, real databases for SMS spams are very limited. Additionally, due to the small length of text messages, the number of features that can be used for their classification is far smaller than the corresponding number in emails.



**Design thinking process:**

1. Empathize:

- Understand the user's needs and pain points regarding spam emails.

- Gather user feedback and analyze existing spam filtering solutions to identify shortcomings.

2. Define:

- Clearly define the problem and establish goals for the spam classifier.

- Define the criteria for success, such as accuracy, false positives, and false negatives.

3. Ideate:

- Brainstorm potential AI-powered spam filtering methods.

- Consider various AI algorithms, feature extraction techniques, and data sources.

- Explore ideas for user interfaces and user experiences.

4. Prototype:

- Create a working prototype of the spam classifier using a subset of the data.

- Experiment with different AI models, such as decision trees, support vector machines, or deep learning neural networks.

- Design a basic user interface for testing.

5. Test:

- Evaluate the prototype with actual users, collecting feedback on the classifier's performance and usability.

- Make improvements based on user feedback and iterate on the design.

6. Implement:

- Develop the final AI-powered spam classifier using the selected model.

- Integrate it into an email system or application.

- Ensure scalability and performance optimization.

7. Launch:

- Deploy the spam classifier to a limited user group to monitor its performance in a real-world environment.

- Continuously refine the model using user feedback and additional data.

8. Learn:

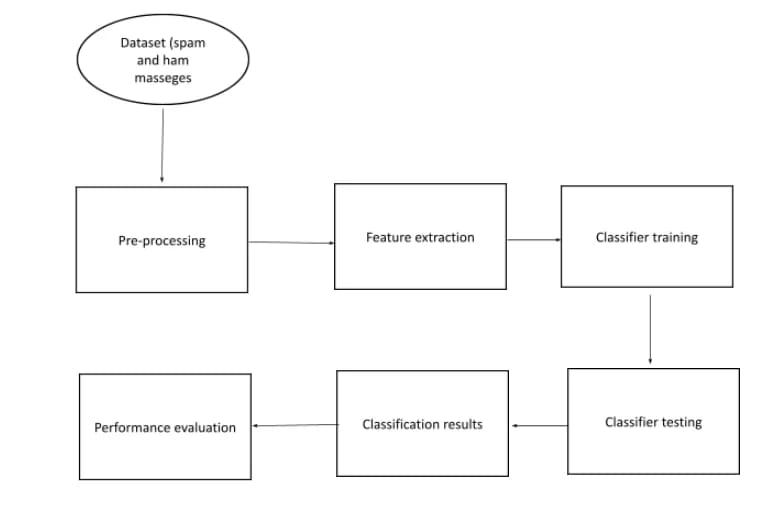
- Gather data and user feedback over time to improve the spam

**Phase of development:**

60% of data is used for the training phase and 40% of data is used for the testing phase. After completing the step feature selection and extraction, we get the feature and the feature that is selected is considered spam. In this classification phase, our datasets will be trained based on the naïve bayes algorithm.

To filter out these messages, a spam filtering system is used that marks a message spam on the basis of its contents or sender. We will be using the Email Spam Classification Dataset dataset which has mainly 2 columns and 5572 rows with spam and non-spam messages. You can download the dataset from

**Spam classification using AI:**



**Data preprocessing steps for creating an AI-powered spam classifier using a database:**

1. Data Collection:

- Gather a labeled dataset containing both spam and non-spam (ham) messages.

2. Data Cleaning:

- Remove any irrelevant metadata or information.

- Handle missing values if any.

- Remove duplicates.

- Correct any encoding issues.

3. Text Preprocessing:

- Tokenization: Split messages into individual words or tokens.

- Lowercasing: Convert all text to lowercase for consistency.

- Stop Word Removal: Remove common words like "the," "and," "in" that do not carry much information.

- Special Character and Punctuation Removal: Eliminate symbols, punctuation, and non-alphanumeric characters.

- Stemming or Lemmatization: Reduce words to their root form (e.g., "running" to "run").

4. Feature Extraction:

- Convert text data into numerical representations, such as TF-IDF (Term Frequency-Inverse Document Frequency) vectors or word embeddings like Word2Vec or GloVe.

5. Data Splitting:

- Split the preprocessed data into training and testing sets for model development and evaluation.

6. Model Development:

- Train a machine learning or deep learning model (e.g., Naive Bayes, SVM, or a neural network) using the preprocessed text data.

7. Model Evaluation:

- Evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and ROC AUC.

8. Model Fine-Tuning:

- Refine the model and its hyperparameters based on performance.

9. Data Integration:

- If necessary, integrate new incoming data into your database and preprocess it using the same steps.

10. Deployment:

- Deploy the AI-powered spam classifier to your application or system.

11. Real-time Processing:

- Ensure the system can handle real-time data and apply the same preprocessing steps to new messages as they arrive.

12. Continuous Monitoring:

- Regularly monitor the model's performance and retrain it as needed to adapt to evolving spam patterns.

These preprocessing steps are essential for creating an effective AI-powered spam classifier that can accurately differentiate between

**Feature extraction techniques:**

1. Bag of Words (BoW): BoW represents text as a collection of unique words in the dataset, ignoring their order. Each word becomes a feature, and the frequency of each word in a document is used as its value.

2. TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF is a variation of BoW that takes into account the importance of words. It assigns a weight to each word based on its frequency in a document and its rarity in the entire dataset.

3. Word Embeddings: Word embeddings like Word2Vec, GloVe, or fastText represent words as dense vectors in a continuous vector space. These vectors capture semantic relationships between words and can be used as features.

4. N-grams: N-grams are sequences of 'n' consecutive words in a text. They capture local word order and can be used as features. Common choices are unigrams (single words), bigrams (two-word sequences), or trigrams (three-word sequences).

5. Character-level Features: You can use character-level n-grams or embeddings to capture patterns at the character level, which can be useful for identifying misspelled words or obfuscated text.

6. Part-of-Speech Tags: Extracting part-of-speech tags for words in a text can provide information about grammatical structure, which can be useful in spam detection.

7. Sentiment Analysis: Analyzing the sentiment of the text can be a helpful feature. Spam messages often exhibit different sentiment patterns compared to legitimate messages.

8. Text Length and Structure: Features such as message length, the presence of special characters, and structural elements like URLs or email addresses can be indicative of spam.

9. Domain-specific Features: Depending on the type of messages you're dealing with, you can extract domain-specific features. For example, if you're classifying email spam, sender information and email headers can be useful.

10. Topic Modeling: Using techniques like Latent Dirichlet Allocation (LDA), you can identify topics within a corpus, and the topic distribution of a message can serve as a feature.

It's important to experiment with various feature extraction techniques and evaluate their performance with your specific dataset and machine learning model. Often, a combination of these techniques can yield

**Machine learning algorithm:**

1. \*\*Data Collection\*\*: Gather a labeled dataset of both spam and non-spam emails. This dataset will be used to train and evaluate your model.

2. \*\*Data Preprocessing\*\*: Clean and preprocess the text data. This may involve removing HTML tags, special characters, and normalizing the text (e.g., converting to lowercase).

3. \*\*Feature Extraction\*\*: Convert the text data into numerical features that machine learning algorithms can understand. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings like Word2Vec or GloVe.

4. \*\*Algorithm Selection\*\*: Choose a suitable machine learning algorithm. Common choices include Naive Bayes, Support Vector Machines, Random Forest, or more advanced methods like deep learning with neural networks.

5. \*\*Model Training\*\*: Split your dataset into training and testing sets. Train your chosen algorithm on the training data to learn the patterns that distinguish spam from non-spam messages.

6. \*\*Hyperparameter Tuning\*\*: Fine-tune the model's hyperparameters to optimize its performance. This may involve cross-validation.

7. \*\*Evaluation\*\*: Evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. You may also use a confusion matrix to understand false positives and false negatives.

8. \*\*Deployment\*\*: Once the model performs well, deploy it as part of your email system to automatically classify incoming emails as spam or ham.

9. \*\*Monitoring and Maintenance\*\*: Continuously monitor the model's performance and update it as new spam patterns emerge.

10. \*\*Feedback Loop\*\*: Implement a feedback mechanism where users can report false positives and false negatives to further improve the model.

The choice of the specific algorithm will depend on the characteristics of your dataset and the level of accuracy you want to achieve. It's also common to combine multiple algorithms or use deep learning techniques for more complex spam classification tasks.

**Appropriate evaluation metrics:**

1. \*\*Accuracy:\*\* This measures the overall correctness of the classifier's predictions. However, in the context of spam classification, accuracy can be misleading if the dataset is imbalanced (i.e., there are many more non-spam messages than spam messages).

2. \*\*Precision:\*\* Precision calculates the proportion of correctly classified spam messages among all messages predicted as spam. It is a good metric to use when you want to minimize false positives, as it tells you how precise the model is in identifying spam.

3. \*\*Recall (Sensitivity):\*\* Recall measures the proportion of correctly classified spam messages out of all actual spam messages. It is important when you want to minimize false negatives, as it tells you how well the model is at capturing spam.

4. \*\*F1 Score:\*\* The F1 score is the harmonic mean of precision and recall. It provides a balanced measure that takes into account both false positives and false negatives. It's often used when you need to strike a balance between precision and recall.

5. \*\*Specificity:\*\* Specificity measures the proportion of correctly classified non-spam (ham) messages out of all actual non-spam messages. It's valuable when you want to