**problem definition**

Design and create a more intelligent AI-powered classifier that can correctly classify a variety of dynamic datasets using cutting-edge machine learning algorithms. The objective is to develop a robust, adaptable, and effective categorization system that outperforms current approaches and gives stakeholders useful insights and actionable information.

**Design thinking**

1. Data gathering: Compile a broad and representative dataset of labeled emails and texts, containing samples of both spam and non-spam.
2. Data Cleaning and Preprocessing: To normalize the text, remove special characters, stopwords, and perform stemming or lemmatization on the text data.
3. Feature engineering: To represent communications quantitatively, significant characteristics from the text data, such as word frequency, n-grams, and text length, are extracted.
4. Model selection: Pick the best deep learning or machine learning models for classification, such as Naive Bayes, SVMs, or neural networks like CNN or LSTM.To assess the performance of the model, divide the dataset into training and testing sets.
5. Model training: Use the training set of data to train the selected model, then adjust the hyperparameters for the optimum performance. Employ techniques like cross-validation to prevent overfitting.
6. Evaluation Metrics: To quantify false positives and false negatives, measure the model's performance using evaluation metrics like precision, recall, F1-score, and accuracy.
7. Threshold Optimization: Depending on the requirements of the application, change the classification threshold to strike a balance between false positives and false negatives.
8. Update the model frequently with fresh information to better account for changing spam patterns.
9. Implement tools that will allow users to report false positives and false negatives, which can be utilized to improve and tweak the model.

Dataset used link:<https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

**Description of dataset**

The "SMS Spam Collection Dataset" is a popular dataset that is often used for text classification and natural language processing (NLP) tasks. It is typically used for building and training machine learning models to classify text messages as either spam or not spam (ham). Here are some key characteristics of this dataset:

1. Data Source: This dataset is often available on platforms like Kaggle and can be used for educational and research purposes.

2. Data Content: The dataset contains a collection of SMS messages, which are labeled as either "spam" or "ham" (not spam). Each message in the dataset is associated with a class label indicating whether it is a legitimate message (ham) or a spam message.

3. Data Format: The dataset is usually provided in a structured format, often as a CSV file, with two main columns: one for the text of the SMS message and another for the label (spam or ham).

4. Size: The dataset typically contains a few thousand SMS messages, with a relatively balanced distribution of spam and ham messages.

5. Purpose: This dataset is commonly used for training and evaluating text classification models, such as machine learning algorithms and NLP techniques, to automatically identify and filter out spam messages in text messages.

6. Applications: The applications of this dataset include spam detection in SMS messages, creating spam filters for mobile devices, and improving user experience by reducing unwanted messages.

**Data preprocessing steps**

1. Loading the Dataset: Import the dataset into your preferred data analysis environment, such as Python with libraries like Pandas, or R.

2. Data Inspection: Examine the dataset to get a sense of its structure and content. Check for missing values, data types, and any issues that need addressing.

3. Data Cleaning: - Handle Missing Data: If there are any missing values in the dataset, decide on a strategy to handle them, which might include removal, imputation, or other methods.

Text Cleaning: Depending on the quality of the text data, you might need to perform text cleaning tasks like removing special characters, lowercasing, and stemming/lemmatization.

4. Data Label Encoding: Convert the class labels (spam/ham) into numerical values. For instance, you can map "spam" to 1 and "ham" to 0.

5. Data Splitting: Divide the dataset into training and testing sets to evaluate the performance of your machine learning model.

**Feature Extraction Techniques**

1. Bag of Words (BoW):

Tokenization: Split text into individual words (tokens).

Vocabulary Building: Create a vocabulary of unique words from the entire dataset.

Count Vectorization: Represent each text message as a vector, where each element corresponds to the count of a word in the message. Tools like CountVectorizer in scikit-learn can be used.

2. TF-IDF (Term Frequency-Inverse Document Frequency):

This technique assigns a weight to each word in a document based on its frequency in the document and its inverse frequency in the entire dataset. It helps to identify words that are important for a specific document but not common across all documents.

3. Word Embeddings :

Word embeddings represent words as dense vector representations, capturing semantic relationships. Pre-trained word embeddings like Word2Vec or GloVe can be used to transform words into vectors.

4. Feature Engineering:

Create custom features like message length, number of special characters, and other domain-specific features that might help improve classification.

5. Text Representation Normalization:

Normalize the feature vectors to have a consistent scale. Techniques like scaling or standardization can be applied to numerical features.

6. Dimensionality Reduction:

If the dataset has many features, you may consider applying dimensionality reduction techniques like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to reduce the number of features.

**Machine Learning Algorithms**

Naive Bayes:

Naive Bayes algorithms, such as Multinomial Naive Bayes, are commonly used for text classification tasks, including spam detection. They work well with text data and are relatively simple to implement.

Support Vector Machines (SVM):

SVMs can be effective for text classification. They aim to find a hyperplane that best separates spam and ham messages.

Deep Learning Models (e.g., Neural Networks):

You can also explore deep learning techniques using neural networks, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for text classification.

**Model Training Process**

Data Splitting: Divide your dataset into training and testing sets to train and evaluate your model's performance.

Feature Extraction: Apply the feature extraction techniques mentioned earlier to represent text data as numerical features.

Model Selection: Choose one or more of the machine learning algorithms mentioned above.

Model Training: Train the selected model(s) on the training data. This involves feeding the algorithm with the labeled examples (spam/ham messages) to learn the patterns in the data.

Model Evaluation: After training, evaluate the model's performance on the testing dataset using appropriate evaluation metrics.

**Evaluation Metrics**

Accuracy: Measures the overall correctness of the model's predictions. However, accuracy may not be the best metric when classes are imbalanced.

Precision: Calculates the ratio of true positive predictions to all positive predictions. It's a measure of how many of the predicted spam messages are actually spam.

Recall (Sensitivity): Measures the ratio of true positive predictions to all actual positive instances. It's a measure of how many spam messages were correctly identified.

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics.

Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC): These are used for binary classification problems and provide a graphical representation of the model's performance and an associated metric for its quality.

Confusion Matrix: A table showing true positive, true negative, false positive, and false negative predictions, which can be used to compute various metrics.

**Innovative Techniques**

1. Word Embeddings and Pre-trained Models:

Utilize pre-trained word embeddings like Word2Vec, GloVe, or fastText to capture semantic relationships in the text. These embeddings can improve the model's understanding of the context and meaning of words.

2. BERT and Transformers:

Consider leveraging transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) for text classification. Fine-tuning pre-trained BERT models on your dataset can lead to state-of-the-art results.

3. Data Augmentation:

Apply data augmentation techniques to artificially increase the size of your training dataset. This can include adding synonyms, changing word order, or introducing minor textual variations to existing samples.

4. Ensemble Models:

Create ensemble models that combine the predictions of multiple models. Stacking different types of models (e.g., Naive Bayes, SVM, and deep learning) can often improve overall performance.

5. Active Learning:

Implement active learning strategies to intelligently select the most informative samples for labeling. This can help improve the model with fewer labeled examples.

6. Anomaly Detection:

-Explore anomaly detection techniques to identify unusual and potentially harmful messages that don't fit typical spam patterns. This can complement traditional spam detection methods.

7. Transfer Learning:

Investigate transfer learning techniques where models trained on one text classification task can be fine-tuned for your specific task, potentially requiring less labeled data.

8. Feature Engineering:

Experiment with domain-specific feature engineering. For instance, you can create features that capture the use of non-standard characters or formatting, which are common in spam messages.

SUBMISSION

**IMPORT THE MODULES**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import re

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import CountVectorizer

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

**LOAD THE DATASET**

df = pd.read\_csv('spam.csv', encoding='latin-1')

df = df[['v1', 'v2']]

df.tail()

**PREPROCESS THE DATASET**

def preprocess\_text(text):

    text = re.sub(r'[^a-zA-Z]', ' ', text)

    text = text.lower()

    text = text.split()

    text = ' '.join(text)

    return text

df['v2'] = df['v2'].apply(preprocess\_text)

df['v2']

**MODEL TRAINING**

X = df['v2']

y = df['v1']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

vectorizer = CountVectorizer()

X\_train = vectorizer.fit\_transform(X\_train)

X\_test = vectorizer.transform(X\_test)

**MODEL EVALUATION**

from sklearn.naive\_bayes import MultinomialNB

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

classifier = MultinomialNB()

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

y\_pred = classifier.predict(X\_test)

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

print("Accuracy:", accuracy)

print(classification\_report(y\_test, y\_pred))

**VISUALIZATION**

count\_Class=pd.value\_counts(df['v1'], sort=True)

count\_Class.plot(kind='bar', color=['blue', 'orange'])

plt.title('Bar chart')

plt.show()

count\_Class.plot(kind='pie', autopct='%1.0f%%')

plt.title('Pie chart')

plt.ylabel('')

plt.show()

**DESCRIPTION OF THE DATASET**

The "SMS Spam Collection Dataset" is a popular dataset that is often used for text classification and natural language processing (NLP) tasks. It is typically used for building and training machine learning models to classify text messages as either spam or not spam (ham). Here are some key characteristics of this dataset:

1. Data Source: This dataset is often available on platforms like Kaggle and can be used for educational and research purposes.

2. Data Content: The dataset contains a collection of SMS messages, which are labeled as either "spam" or "ham" (not spam). Each message in the dataset is associated with a class label indicating whether it is a legitimate message (ham) or a spam message.

3. Data Format: The dataset is usually provided in a structured format, often as a CSV file, with two main columns: one for the text of the SMS message and another for the label (spam or ham).

4. Size: The dataset typically contains a few thousand SMS messages, with a relatively balanced distribution of spam and ham messages.

5. Purpose: This dataset is commonly used for training and evaluating text classification models, such as machine learning algorithms and NLP techniques, to automatically identify and filter out spam messages in text messages.

6. Applications: The applications of this dataset include spam detection in SMS messages, creating spam filters for mobile devices, and improving user experience by reducing unwanted messages.