SPAM SMS DETECTION

The project aims to build an AI model to classify SMS messages as spam or legitimate. Using text preprocessing techniques such as TF-IDF vectorization and word embeddings, we employ classifiers like Naive Bayes, Logistic Regression, and Support Vector Machines to identify spam messages. The models are evaluated based on accuracy, precision, recall, and F1 score. Ultimately, the best performing model is identified by comparing these metrics.

Steps:

1. Import Necessary Libraries:

The code imports essential libraries such as numpy, pandas, matplotlib, seaborn, re, string, warnings, and sklearn for various data processing, visualization, and machine learning tasks. These libraries provide functions and tools needed to load, clean, process, visualize, and build machine learning models.

2. Load and Clean Data:

The dataset is loaded using the pd.read_csv() function with the specified encoding 'ISO-8859-1'. This step reads the data from the CSV file into a pandas DataFrame. The relevant columns are kept, specifically 'v1' and 'v2', which are then renamed to 'label' and 'message' respectively for clarity. The first few rows of the dataset are displayed to verify that the data has been loaded correctly.

3. Print Summary Statistics:

Summary statistics of the 'message' column are printed to understand the data better. This includes metrics like the count of messages, the number of unique messages, the most frequent message, and its frequency. Numerical and categorical features are identified by checking the data types of each column, and these features are printed to confirm the columns that will be used in further analysis and modeling.

4. Stemming Function for Text Preprocessing:

A stemming function is defined using PorterStemmer to preprocess the text data. Stemming reduces words to their root form, which helps in standardizing the text data. The function converts text to lowercase, removes non-alphabetic characters, and applies stemming to each word. The stemming function is applied to the 'message' column, transforming the text data into a new preprocessed text column.

5. Visualize Data Distribution:

The distribution of spam and legitimate messages is visualized using a count plot. This plot helps in understanding the class balance in the dataset by showing the number of spam and legitimate messages. This step is crucial for assessing if there is any class imbalance that might affect model training and evaluation.

6. Preprocess Text Data:

A text cleaning function is defined to preprocess the text data further. The function converts text to lowercase, removes URLs, digits, punctuation, and extra spaces. This cleaning helps in normalizing the text data by removing unnecessary characters and spaces. The cleaned text data is stored in a new column called 'cleaned_message'.

7. Split Data into Training and Testing Sets:

The data is split into training and testing sets using the train_test_split() function from sklearn. An 80-20 split is used, where 80% of the data is used for training the models and 20% is reserved for testing. This step is essential for evaluating the model's performance on unseen data.

8. Evaluate Model Function:

A function is defined to evaluate a model's performance. This function calculates accuracy, precision, recall, and F1 score using the predictions made by the model on the test set. These metrics are printed and returned for further analysis. Accuracy measures the overall correctness, precision measures the correctness of positive predictions, recall measures the ability to identify positive instances, and F1 score is the harmonic mean of precision and recall.

9. TF-IDF Vectorization and Model Training:

The TF-IDF vectorizer is used to transform the text data into numerical features. TF-IDF stands for Term Frequency-Inverse Document Frequency, which weights the importance of words in the text. Three different models (Naive Bayes, Logistic Regression, and Support Vector Machine) are trained using the TF-IDF features. Each model is evaluated using the previously defined function, and their performance metrics are printed.

10. Visualize and Compare Results:

The performance metrics of all models are stored in a DataFrame for easier comparison. The results are visualized using a bar plot, which compares the performance of different models across various metrics like accuracy, precision, recall, and F1 score. This visualization helps in understanding which model performs best overall.

11. Determine Best Model:

The best model is determined based on the highest F1 score, which balances precision and recall. The name and F1 score of the best model are printed, indicating which model performed the best in classifying spam and legitimate messages based on the given dataset.