Spam Email detection

Problem Statement:

- A common problem that arises in the communication through emails is the situation of spam . A spam mail consists of unnecessary, unwanted, and usually malicious messages.
- This project is concerned with implementation, evaluation and comparison of various machine learning classifier algorithms to detect spam mail.
- The main task of our project is to develop an effective spam filter or a spam detector which classifies messages received as spam or ham.

Algorithms used for implementation:

- Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- Multinomial Naive Bayes
- Decision Tree
- Logistic Regression
- Random Forest
- AdaBoost
- Gradient Boost and
- XGBoost.

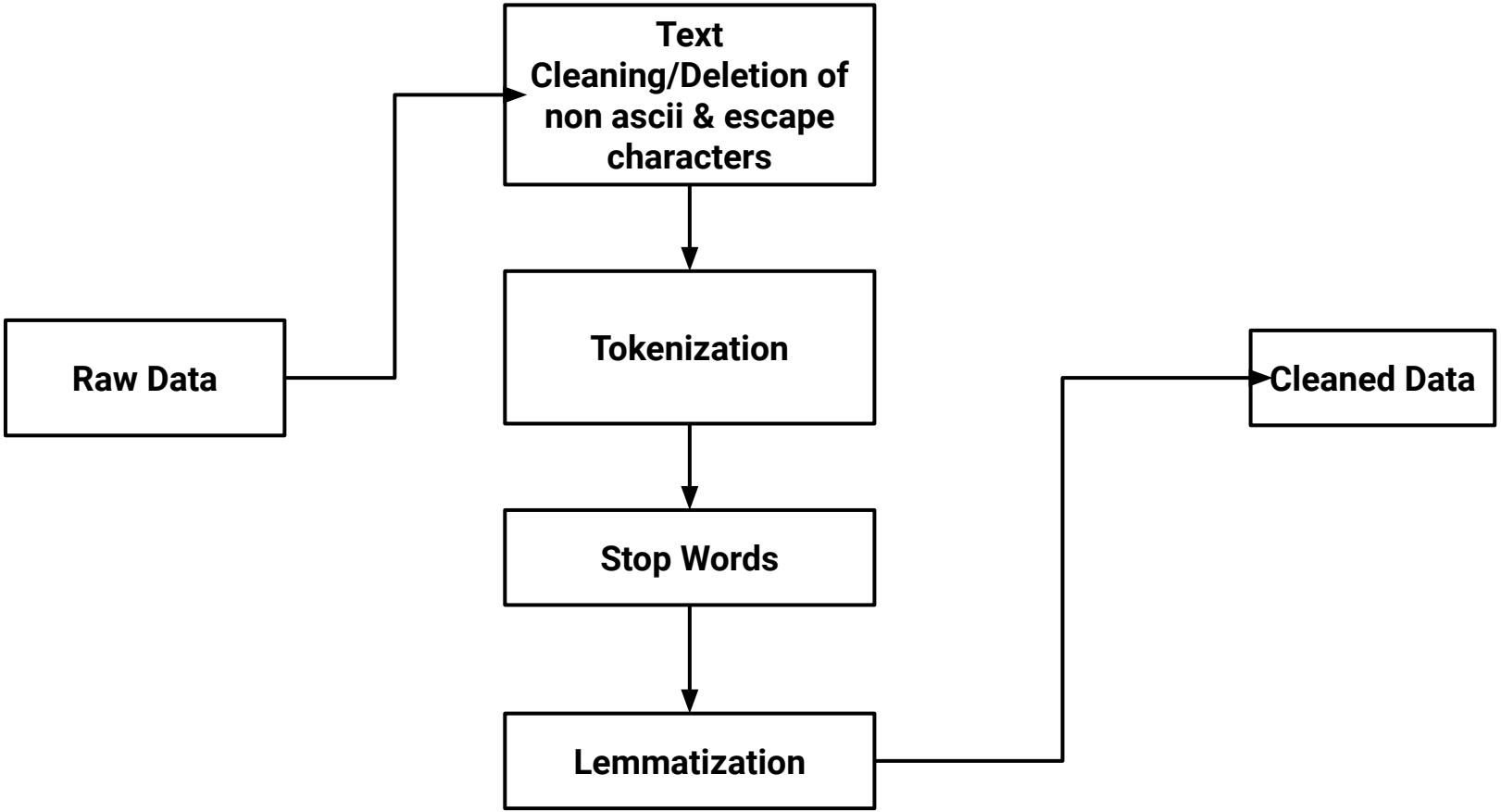
Data collection:

- The dataset we used, consists of emails which are classified and labelled as spam or not. It has entities like category and message, with more than 5000 entries.
- Dataset Source: Email Spam Detection (kaggle.com)
- The quality of the data appears to be reasonable, with structured information of all the emails considered which makes the spam detection become easy.

Data preprocessing:

Below NLP techniques are used for data preprocessing:

- **Cleaning**: Removing HTML tags, metadata, and handling special characters.
- **Tokenization**: Breaking text into individual words or tokens.
- **Lowercasing**: Converting text to lowercase for uniformity.
- **Stopwords Remova**l: Eliminating common, non-informative words.
- **Lemmatization**: Reducing words to their root forms.


Text Cleaning/Deletion of non ascii & escape characters
Raw Data
Tokenization
Cleaned Data
Stop Words
Lemmatization

Feature engineering:

Feature engineering is the process of selecting, transforming, and creating features (variables) from raw data that are suitable for machine learning models.

Here, we have used several steps for the feature engineering

- **Text Preprocessing:** The raw text data in the 'email' column is preprocessed to make it modeling-ready. This involves changing the text to lowercase, deleting non-ASCII letters, escape characters, stopwords, and punctuation. These techniques assist in cleaning the text data and extracting useful features from it.
- **TF-IDF Vectorization:** The text data is converted into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization approach. The TF-IDF is measure of originality of word by comparing the number of times a word appears in the document with the number of documents the word appears in. It helps in converting text data into a numerical format that machine learning algorithms can understand.

- **Feature Extraction:** The length of each email message is estimated and included as a feature. This feature measures the length of the text, which could be beneficial for identifying spam from non-spam emails.
- **Label Encoding:** The target variable 'Category' is encoded with label encoding. Label encoding transforms categorical labels into numerical representations, which are required for training machine learning models.
- **Dimensionality Reduction:** Limiting the number of features in TF-IDF vectorization (via'max_features') reduces dimensionality.

```
max_features=2000
```

Model selection and architecture:

Machine learning models used for the task of spam email classification are mentioned below:

- Support Vector Machine (SVM): SVM is a supervised machine learning algorithm used for classification, regression and outlier detection tasks.For optimal performance we have used below hyperparameters for this model

  kernel="sigmoid", gamma=1.0

- K-Nearest Neighbors (KNN): This algorithm also belongs to supervised learning domain. It is used in data mining, pattern recognition applications. This algorithm assumes that the similar things exist in close distance. Euclidean distance is used for calculating the distance between data points. We considered the default parameters for KNeighborsClassifier() , where in it considers 5 nearest neighbours.

Model selection and architecture (continuation)

- Multinomial Naive Bayes: It is a probabilistic Naive Bayes classifier used to calculate the probability distribution of text data, which makes it well-suited for analyzing data containing discrete frequency features or event counts in different natural language processing tasks. This algorithm is commonly used for text classification tasks.
  We have used default parameters of MultinomialNB().

- Decision Tree: It is a supervised learning algorithm. According to the input features, it divides the feature space into regions, and at each internal node, it decides which branch to proceed depending on the feature values. This process continues recursively until a stopping criterion is met, such as reaching a maximum depth, minimum number of samples in a node, or no further improvement in impurity reduction.
  To prevent overfitting and to balance model complexity and performance we have chosen the below parameter:
  Maximum depth as 5.

Model selection and architecture (continuation)

- Logistic Regression: It is a linear classification algorithm for binary classification tasks that estimates based on the features that the probability of the given input belongs to particular class

  The parameters we have used here are “solver =liblinear” and “penalty=l1”.
- Random Forest: It is an ensemble learning. Prediction of multiple decision trees is combined to reduce overfitting and improve accuracy.
  The parameters used here “n_estimators = 50” and “random_state=2”.

Model selection and architecture (continuation)

- AdaBoost: It is an ensemble learning. This is used to create a strong classifier combining multiple weak learners
  The parameters used here are “n_estimators = 50” where it indicates 50 weak learners which is decision trees by default and “random_state=2”.

- Gradient Boosting: It is an ensemble learning.This constructs the predictive models i.e. it combines multiple weak learners which is decision tree in default sequentially.
  The parameters used here are “n_estimators = 50” and “random_state=2”.

Model selection and architecture (continuation)

- XGBoost: It is an advanced implementation of Gradient Boosting mainly used for handling large datasets. It incorporates regularization techniques,built-in support for the handling missing data providing high accuracy.
  The default parameters used here are “n_estimators = 50” and “random_state=2”

Model architecture:

Model architecture:

- All models are implemented using Python’s scikit-learn library.
- Each model is instantiated with either the default or supplied parameters.
- The models are trained using the TF-IDF transformed text data ('X_train') and related labels ('y_train').
- Following the training, the models are assessed on the test dataset ('X_test', 'y_test').
- Precision and Accuracy are used as evaluation parameters.

Hyperparameter tuning:

Hyperparameter Tuning:

- Grid search cross-validation is used to find the optimal hyperparameters for each model.
- For each model, a grid of hyperparameters is defined, and the values that optimize the chosen evaluation metric (accuracy or precision) are chosen. 5-fold cross validation is performed.
- The hyperparameter grids are created using common selections for each model, with the goal of identifying the best-performing configuration.

Ensemble learning:

Ensemble Learning:

- Following the evaluation of individual models, the top-performing models are used to build a Voting Classifier.
- Voting Classifier is a ensemble technique used for combining predictions from numerous base models to provide the most frequent prediction.
- Hard voting mechanism is used, in which the class label with the highest frequency among the predictions of all classifiers is chosen as the final prediction.
- The Voting Classifier is trained using the same training data ('X_train' and 'y_train').

Final model selection:

The optimal model is chosen based on its performance on the validation dataset (X_test, y_test) following hyperparameter adjustment.

Training and evaluation:

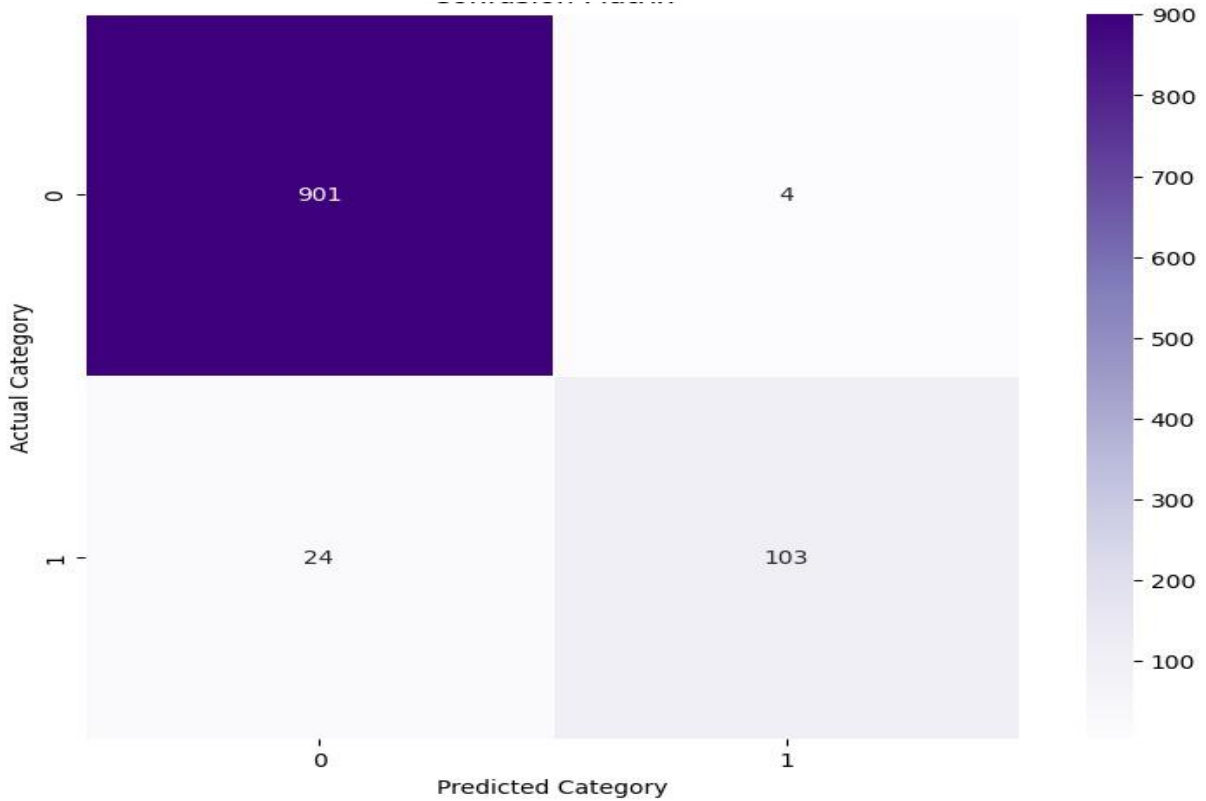
- The dataset is loaded and preprocess is performed by removing duplicates, encoding labels and preprocessing text data.
- TF-IDF vectorization is used to extract features from text data and create additional features.
- Splitting the dataset into training and testing sets.
- Multiple classifiers are initialised and each machine learning classifier is trained using TF-IDF vectorizer .
- Hyperparameter tuning using grid search cross-validation is performed to find the best hyperparameters for each classifier.

- Voting classifier is used to determine the best performing model and is trained using the training data.
- Once the Voting classifier is done each classifier is evaluated on the test data.
- Accuracy and precision for each classifier is calculated.
- Best performing model is selected from the evaluation metrics, such as validation accuracy after hyperparameter tuning.

Results:

- After the grid search the best performing model is “SVM classifier” with an accuracy and precision of 0.9826 and 0.9827 respectively
- Accuracy of the best model is 0.9728682170542635
- Precision of the best model is 0.9726465652786018
- Recall of the best model is 0.9728682170542635
- F1-score of the best model is 0.9718570002595928

Confusion Matrix



Thank you....

Divya Reddy Sadhula

Vasudha Maddi

Swathi Priya Mardi

Meghana Kanakam

Manikanta V