Report on Email Spam Classifier

Snehal MA23M020

November 3, 2024

Introduction

The assignment deals with classifying emails into two categories: ham and spam mails. This report shows several machine learning models implemented to perform spam detection. The models are The models include:

- Logistic Regression
- Naive Bayes Classifier
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Perceptron

Combing all the models and taking the majority count i.e. the Voting Ensemble Each model has been trained and tested on a dataset of emails, and their performances have been evaluated to determine their effectiveness in classifying spam.

Libraries

- **os**: Provides functions to interact with the operating system, such as file and directory handling.
- **pandas**: A data manipulation and analysis library, used for handling and processing data in DataFrame format.
- **numpy**: A library for numerical operations in Python, providing support for arrays, matrices, and mathematical functions.
- **string**: Contains common string operations and constants, such as punctuation removal.
- **math**: Provides mathematical functions, such as logarithms, used for Naive Bayes calculations.
- **sklearn.svm.SVC**: Support Vector Classification from scikit-learn, a machine learning library, used for training SVM models.
- **sklearn.model_selection.train_test_split**: A utility function from scikit-learn that splits data into training and test sets for model evaluation.

Dataset

The dataset used to train the model consists of the email and the label i.e. whether it is spam or ham. The summary of the dataset is:

```
d.columns
```

```
Index(['target', 'text'], dtype='object')
```

+ Code + Text

```
[19] d.size
```

```
11144
```

```
[25] d[d['target']=='spam'].value_counts()
```

target	text	count
spam	Please call our customer service representative on FREEPHONE 0808 145 4742 between 9am-11pm as you have WON a guaranteed â£1000 cash or â£5000 prize!	4
	I don't know u and u don't know me. Send CHAT to 86688 now and let's find each other! Only 150p/Msg rcvd. HG/Suite342/2Lands/Row/W1J6HL LDN. 18 years or over.	3
	Camera - You are awarded a SiPix Digital Camera! call 09061221066 fromm landline. Delivery within 28 days.	3
	Loan for any purpose â£500 - â£75,000. Homeowners + Tenants welcome. Have you been previously refused? We can still help. Call Free 0800 1956669 or text back 'help'	3
	HMV BONUS SPECIAL 500 pounds of genuine HMV vouchers to be won. Just answer 4 easy questions. Play Now! Send HMV to 86688 More info:www.100percent-real.com	3
	...	...
	Hello darling how are you today? I would love to have a chat, why dont you tell me what you look like and what you are in to sexy?	1
	Hello from Orange. For 1 month's free access to games, news and sport, plus 10 free texts and 20 photo messages, reply YES. Terms apply: www.orange.co.uk/ow	1
	Hello. We need some posh birds and chaps to user trial prods for champneys. Can i put you down? I need your address and dob asap. Ta r	1
	Here is your discount code RP176781. To stop further messages reply stop. www.regalportfolio.co.uk. Customer Services 08717205546	1
	we tried to contact you re your response to our offer of a new nokia fone and camcorder hit reply or call 08000930705 for delivery	1

653 rows × 1 columns

```
[26] d[d['target']=='ham'].value_counts()
```

target	text	count
ham	Sorry, I'll call later	30
	I cant pick the phone right now. Pls send a message	12
	Ok...	10
	Okie	4
	Ok.	4
	...	...
	I forgot 2 ask Ì_ all smth.. There's a card on da present lei... How? ÏÍ all want 2 write smth or sign on it?	1
	I get out of class in bsn in like <#> minutes, you know where advising is?	1
	I got another job! The one at the hospital doing data analysis or something, starts on monday! Not sure when my thesis will got finished	1
	I got arrested for possession at, I shit you not, <TIME> pm	1
	ÏÍ wait 4 me in sch i finish ard 5..	1

4516 rows × 1 columns

```
Training data: 4457 emails
Test data: 1115 emails
8337
vocab size: 8337 words.
```

Data Loading and Preprocessing

Data Loading

The data loading function performs the following steps:

	target	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Figure 1: a sample from dataset

```
Ham emails:
  target                                                text  label
0    ham  Go until jurong point, crazy.. Available only ...      0
1    ham                      Ok lar... Joking wif u oni...      0
3    ham  U dun say so early hor... U c already then say...      0
4    ham  Nah I don't think he goes to usf, he lives aro...      0
6    ham  Even my brother is not like to speak with me. ...      0

Spam emails:
   target                                                text  label
2    spam  Free entry in 2 a wkly comp to win FA Cup fina...      1
5    spam  FreeMsg Hey there darling it's been 3 week's n...      1
8    spam  WINNER!! As a valued network customer you have...      1
9    spam  Had your mobile 11 months or more? U R entitle...      1
11   spam  SIX chances to win CASH! From 100 to 20,000 po...      1
```

Figure 2: some samples of spam and ham emails

1. **Reading the Dataset**: The CSV file is read using `pandas.read_csv`.
2. **Selecting Relevant Columns**: Only the 'target' and 'text' columns are retained.
3. **Renaming Columns**: The columns are renamed to 'label' and 'text' for clarity.
4. **Mapping Labels**: The labels are mapped to numerical values: 0 for 'ham' and 1 for 'spam'.
5. **Shuffling the Data**: The dataset is shuffled to ensure random distribution.
6. **Splitting the Data**: The data is split into training and testing sets based on a specified `test_size` which is 0.2.

Text Preprocessing

The raw text can not be fed to machine learning algorithms as it is , So text preprocessing is crucial for converting raw text into a format suitable for machine learning algorithms. The preprocessing steps include:

- **Removing Punctuation**: All punctuation marks are removed using `str.translate`.
- **Converting to Lowercase**: Text is converted to lowercase to ensure uniformity.
- **Removing Stopwords**: Common English stopwords (e.g., 'the', 'and', 'is', 'in', 'to', 'of', 'a', 'for', 'on', 'with', 'as', 'this', 'that', 'it', 'at') are removed to reduce noise.
- **Tokenization**: The text is split into individual words (tokens).

Vocabulary Building

A vocabulary of the most frequent words which is found to be 5000 words is built from the training data:

1. **Word Counting**: The frequency of each word in the dataset is counted.
2. **Sorting Words**: Words are sorted based on their frequency in descending order.
3. **Selecting Top Words**: The top `max_features` words are selected to form the vocabulary.

Feature Extraction

Till now we have data in form of text only which needs to be converted into numerical features for further analysis. Text data is transformed into numerical feature matrices:

1. **Initializing Matrices**: A zero matrix X of shape (number of emails, vocabulary size) is created.
2. **Mapping Words to Indices**: Each word in the vocabulary is assigned a unique index.
3. **Term Frequency Calculation**: The term frequency (count of each word in an email) is calculated and stored in X.
4. **Label Extraction**: The labels are extracted into an array y.

Models

Since all the pre-processing is done now , so the training data is fed to models.

K-Nearest Neighbors (KNN)

Introduction

The K-Nearest Neighbors (KNN) algorithm is a non-parametric, instance-based learning method employed for classification tasks. It classifies emails as spam or legitimate (ham) by assessing the similarity between a new email and previously labeled examples.

Implementation Details

- **Training Data** $(X_{\text{train}}, y_{\text{train}})$: The feature matrix and corresponding labels for the training set.
- **Test Data** X_{test}: The feature matrix for the test set.
- **Number of Neighbors** (k): The number of nearest neighbors considered, in this model it is taken 5.
- **Iterating Over Test Samples**: For each test sample x_{test}, compute the Euclidean distance to all training samples:

$$\text{distance}_i = \sqrt{\sum_{j=1}^{n} (x_{\text{test},j} - x_{\text{train},i,j})^2} \tag{1}$$

- **Identifying Neighbors**: Sort distances and select the indices of the k smallest distances.
- **Making Predictions**: Use majority voting to predict labels:
- **Returning Results**: Convert `predictions` to a NumPy array and return it.

Why KNN should be used in Email Spam Classification ?

- Non-Parametric Nature: KNN does not assume any specific underlying distribution in the data, which is advantageous for email classification, as spam emails can exhibit varied patterns.
- **Adaptability to Complex Boundaries**: Spam and non-spam emails are often not linearly separable, and they may form clusters within the feature space. KNN's ability to adapt to non-linear decision boundaries allows it to classify emails accurately by considering local patterns in the data.
- **Minimal Training Time**: KNN has a low training time since it memorizes the dataset and applies it directly during prediction. This is beneficial if training time is limited and memory resources are sufficient.

Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. It models the probability that a given input belongs to a particular class.

Implementation Details

- **Sigmoid Function**: The sigmoid function is used to map any real-valued number into the [0,1] interval, representing probability.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- **Weights and Bias Initialization**: Weights and bias are initialized to zeros.
- **Gradient Descent**: The model is trained using gradient descent to minimize the logistic loss function.
- **Regularization**: L2 regularization is applied to prevent overfitting.
- **Imbalanced Dataset**: Class weights are calculated to handle imbalanced datasets, assigning higher importance to the minority class.

Why Logistic Regression is Suitable for Email Spam Classification

Logistic Regression is a simple but powerful algorithm suitable for linearly separable data. Its probabilistic output is useful for understanding the confidence of predictions.

Binary Classification:

Logistic regression is inherently a binary classifier, making it well-suited for distinguishing between two classes, such as "spam" and "not spam."

Interpretability:

Logistic regression is relatively easy to interpret. Each feature weight indicates how much it contributes to the probability of an email being classified as spam.

Handling Imbalanced Data:

Logistic regression, especially when combined with class weights, can handle imbalanced datasets effectively by adjusting the importance given to each class during training. This helps in situations like email spam detection, where spam emails are often outnumbered by legitimate emails.

Efficiency:

Logistic regression is computationally efficient for large datasets, which is often the case in email filtering tasks where the volume of data can be significant.

SVC

This model is implemented using scikit learn library.

Implementation Details

The `SVC` class from Scikit-learn is initialized with a linear kernel and a regularization parameter C set to 1.0. The choice of a linear kernel indicates that the model will try to find a hyperplane that separates the two classes in the feature space.

Model Training and Prediction

The `fit` method is called on the `svm_model` object with the training dataset $(X_{\text{train}}, y_{\text{train}})$. This step involves learning the optimal hyperplane that separates the classes in the feature space defined by the input features. The trained model is used to make predictions on the test dataset (X_{test}) by calling the `predict` method, which outputs predicted labels for the test samples.

Why Use SVC for Email Spam Classification?

Effective in High Dimensions

SVMs are particularly effective in high-dimensional spaces, which is common in text classification tasks like email spam detection where the feature space may consist of a large number of words or phrases.

Robustness to Overfitting

The regularization parameter C helps control overfitting.

Linear and Non-Linear Classification

Although a linear kernel is used in the code, SVM can easily switch to non-linear kernels (like RBF or polynomial) if the data is not linearly separable.

Interpretability

The hyperplane defined by the SVM model can provide insights into the features that influence the classification decisions, helping to understand what characteristics make an email more likely to be spam.

Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic model based on Bayes' theorem with the "naive" assumption of Class independence.

Implementation details

The `word_frequency` function calculates word frequencies for both spam and ham emails. The process works as follows:

- **Initialize Dictionaries:** It creates dictionaries to store word counts (`no_of_spam_word` and `no_of_ham_word`) for each word in the vocabulary (`vocab`) and sets initial counts to zero.

- **Count Spam and Ham Words:** The function iterates over each email in the dataset. It preprocesses the text . For each email, if it's spam (`label == 1`), it increments the total spam count and updates the count for each word found,in `no_of_spam_word`. If it's ham, it does the same in `no_of_ham_word`.
- **Output:** This function returns dictionaries for word frequencies in spam and ham emails, as well as the total number of spam and ham emails.

The `naive_bayes_train` function uses the frequencies calculated to determine the prior and likelihood probabilities needed for Naive Bayes classification.

- **Prior Probabilities:**
  - The prior probability of spam (or ham) is calculated as the fraction of spam emails over the total messages.
- **Likelihood Probabilities:**
  - The total number of words in spam (`total_words_spam`) and ham (`total_words_ham`) emails are computed.
  - Using these totals, the likelihood probability for each word in the vocabulary is calculated. To prevent zero probabilities (which would cause issues in the Naive Bayes classifier), Laplace smoothing is applied by adding +1 to the numerator and adding the vocabulary size to the denominator.
  - The function returns prior probabilities (`prior_spam`, `prior_ham`) and dictionaries of likelihood probabilities for each word (`spam_word_probs`, `ham_word_probs`).

The `naive_bayes_predict` function makes predictions on new emails based on the computed priors and likelihoods.

- **Logarithmic Transformation:** To avoid underflow (very small numbers from multiplying many probabilities), the function computes the log-probabilities of spam and ham for each email.
- **Prediction:** After processing all words, the class with the higher log-probability is chosen as the prediction: if `log_prob_spam` is higher, the email is classified as spam (1), otherwise as ham (0). This function returns an array of predictions for each email in `X`.

Why Naive Bayes is Suitable for Email Spam Classification

Naive Bayes is highly effective for email spam classification, particularly because it is a probabilistic model that makes strong, simplifying assumptions.

- **Text Classification:** Naive Bayes is commonly used in text classification problems where the data consists of words and their frequencies. Its probabilistic nature aligns well with text data and allows it to utilize word frequencies to classify emails accurately.

- **Assumption:** Naive Bayes assumes that the presence of each word in an email is independent of the presence of any other word (the "naive" assumption). Although this assumption may not be strictly true, it works well in practice for tasks like spam filtering where individual words can carry significant weight (e.g., "offer," "free," "buy now").
- **Good Performance with Sparse Data:** The model performs well with sparse data where only a subset of the vocabulary appears in any given email. This characteristic is typical in text datasets and helps Naive Bayes to excel in email classification.
- **Handling of Imbalanced Data:** Naive Bayes inherently considers the probabilities of each class and is less prone to biases from class imbalance.

Perceptron

The Perceptron is a foundational algorithm in machine learning, particularly for binary classification tasks.

Implementation Details

- **Training Data** (X, y): The feature matrix X containing the training samples, and the label vector y indicating whether each sample is spam (1) or not spam (0).
- **Learning Rate**: A hyperparameter controlling the step size during weight updates.
- **Epochs**: The number of complete passes over the training dataset.

Initialization

The algorithm begins by initializing weights wts to zero and setting the bias to zero. Labels are transformed from $\{0, 1\}$ to $\{-1, 1\}$ for compatibility with the Perceptron algorithm.

Training Loop

For each epoch, the algorithm iterates over all training samples:

- **Linear Output Calculation**: For each sample x_i,

$$\text{linear\_output} = \text{dot}(x_i, wts) + \text{bias}$$

- **Prediction**:

$$\text{prediction} = \begin{cases} 1 & \text{if linear\_output} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- **Weight Update**: If the prediction does not match the actual label, the weights and bias are updated using the learning rate:

Why perceptron is suitable for Email Spam Classification

Binary Classification

The Perceptron is inherently designed for binary classification, making it ideal for the spam classification problem.

Simplicity and Efficiency

The Perceptron is computationally efficient and straightforward to implement. It requires minimal resources, making it suitable for real-time applications.

Adaptability

The Perceptron can learn from new data continuously by updating weights based on incoming samples, allowing it to adapt to evolving spam patterns.

The Perceptron algorithm offers a simple yet effective method for email spam classification. Its ability to learn from labeled data and create linear decision boundaries makes it a valuable tool in the spam detection .

Voting Ensemble

Ensemble methods combine the predictions of multiple models to improve generalization performance.

Implementation Details

- **Collecting Predictions**: Predictions from all individual models are collected.
- **Majority Voting**: The final prediction is determined by majority voting among the models.
- **Performance Evaluation**: The ensemble's performance is compared to individual models.

Why voting ensemble ?

Ensembles often perform better than individual models by reducing variance and leveraging the strengths of different algorithms.

Model Evaluation

Accuracy

The metric used for evaluation is accuracy, calculated as the proportion of correct predictions over the total number of samples.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Results

Accuracy of all the models

```
Training data: 4457 emails
Test data: 1115 emails
8337
vocab size: 8337 words.
Feature matrices created.

Model Accuracies:
                     Model  Accuracy
       K-Nearest Neighbors  0.904036
       Logistic Regression  0.931839
Support Vector Machine  0.979372
                Naive Bayes  0.982960
                 Perceptron  0.982063
           Voting Ensemble  0.982960
```

Prediction on New Emails

The trained models are used to predict the labels of new, unseen emails:

- The code reads new emails from a specified test folder (`test`), where each email file starts with "email" and ends with ".txt".
- Each email's text is preprocessed into a feature vector, denoted as $\mathbf{x}_{\text{new}}$, using a predefined vocabulary.For each word in the email, the code updates its term frequency in $\mathbf{x}_{\text{new}}$.

Model Predictions

For each email, predictions are obtained from a set of models: **Naive Bayes,Logistic Regression,K-Nearest Neighbors,Support Vector Machine, Perceptron**

Each model outputs a binary prediction (spam or ham).

The individual predictions from each model are combined using majority voting. The most common prediction among the models is selected as the final ensemble prediction for each email.

Results Display

For each email, the code displays the final prediction from the ensemble:

- Spam (+1) if the ensemble predicts spam.
- Ham (0) if the ensemble predicts non-spam (ham).

```
Predictions on New Emails:
    Filename Voting Ensemble Prediction
 email5.txt                    Ham (0)
 email6.txt                   Spam (+1)
 email2.txt                   Spam (+1)
 email7.txt                    Ham (0)
 email8.txt                    Ham (0)
 email4.txt                    Ham (0)
 email3.txt                    Ham (0)
email10.txt                   Spam (+1)
 email9.txt                   Spam (+1)
 email1.txt                    Ham (0)
```

Figure 3: screenshot of the output

To run the code :

The train1.csv file is provided to tain the model and code is provided in an ipynb format , Model can be tested on any test set containing file names in the prescribed format.