Email Spam Classifier

Machine Learning Project

KRISHNAPRAKASH K R

Dated : 03/06/2024

Abstract

The paper details the development and implementation of an Email Spam Classifier using the k-Nearest Neighbours algorithm. This project aimed to give a Shiny web application that is empowered with the capacity to sort email messages into spam or not as spam. The application gives a representation of a text preprocessing strategy in combination with a machine learning algorithm concerning a real-time email classification setting. In the following parts of the report, the scope, methodology, observations, and results are described.

Introduction

Email has become an integral part of everyday communication, serving both personal and professional purposes. However, many people encounter a significant problem: unwanted and sometimes harmful spam emails. These spam messages not only clutter inboxes but can also pose serious risks, such as phishing scams designed to steal personal information or malware that can infect systems.

Traditional spam filters rely on simple, rule-based approaches to identify spam. These methods use fixed rules or keyword lists to flag suspicious messages. While effective to some extent, such filters often fall short when dealing with more sophisticated spam techniques that can easily bypass these basic rules. This creates a pressing need for more advanced methods that can better identify and filter spam emails.

To address this issue, our project focuses on developing an Email Spam Classifier using the k-Nearest Neighbors (kNN) algorithm. This machine learning approach offers a more dynamic and adaptive solution to spam Vectorization: Representing the words as numbers using some text vectorization techniques. One such technique is TF-IDF, which stands for Term Frequency-Inverse Document Frequency.

detection. Unlike traditional filters, the kNN algorithm can learn from data patterns and make more nuanced predictions about whether an email is spam or not.

The classifier is integrated into a Shiny web application, which provides a user-friendly interface for real-time email classification. Users can simply enter email content into the app and receive immediate feedback on whether the email is classified as spam.

The use of machine learning technology is crucial in this context. By leveraging advanced techniques such as kNN, the classifier improves accuracy in identifying spam and reduces the likelihood of legitimate emails being incorrectly flagged. This not only enhances email security but also helps in managing and mitigating the impact of spam on users' inboxes.

Overall, this project aims to advance email security by applying cutting-edge technology to filter out unwanted messages, making email communication safer and more efficient.

Procedure

Data

The classifier uses a dataset of email messages which are classified as either spam or not a spam. Preprocessing had to be run on the data in order to apply the kNN algorithm. Major preprocessing applied were:

Text Cleaning: Removal of special characters, numbers, and other irrelevant text.

Tokenization: Breaking down text into words or tokens.

Stop-word Removal: Removal of commonly used words that are not adding value to the classification.

The dataset was split to train a classifier and test its performance.

Platform Used

The Email Spam Classifier was developed using the following platform and tools:

R Programming Language: Used for implementing the kNN algorithm and data preprocessing.

Shiny Framework: Used for developing the web application. It provides an interactive user interface to the Web application.

RStudio IDE: Used as a development environment for coding and testing the application.

Method Used

The Classification process involved the following steps:

Dataset loading the dataset was read into R and prepared. Text preprocessing Cleaning and transforming the textual data into a form suitable for the kNN algorithm. Training the kNN model based on the pre-processed training dataset. Proper choice of k was optimized using cross-validation. Testing a trained model for its accuracy and efficiency on the testing dataset.

Web Application Development: A shiny web app is developed that provides an interface to the user for feeding email text input and getting the results for spam classification.

R Code

```
# Load necessary libraries
library(shiny) # For creating the web app
library(class) # For kNN implementation
library(tm) # For text preprocessing
library(shinythemes) # For Shiny themes

# Load and preprocess the data
data <- read.csv('email_classification.csv')

# Preprocess the text data
corpus <- Corpus(VectorSource(data$email))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)

# Create Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)
```

```
dtm <- removeSparseTerms(dtm, 0.99)  # Keep only the most frequent terms

# Convert the Document-Term Matrix to a data frame
dtm_data <- as.data.frame(as.matrix(dtm))
dtm_data$label <- data$label

# Split data into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(dtm_data), 0.8 * nrow(dtm_data))
train_data <- dtm_data[train_index, ]
test_data <- dtm_data[-train_index, ]

# Define the kNN model (We use k=3 for this example)
k <- 3
# Define UI for the Shiny app
ui <- navbarPage(
    theme = shinytheme("cosmo"),
    title = "Email Spam Classifier",

    # Home Tab
    tabPanel(
        "Home",
        fluidPage(
            tags$head(
                tags$style(HTML(
                    " .sidebar { background-color: #f2f2f2; }
                      .main { background-color: #e6e6e6; }
                      .content { height: 100%; }
                      .btn-primary { background-color: #007bff; border-
color: #007bff; } "
                ))
            ),
            titlePanel("Email Spam Classifier"),
            sidebarLayout(
                sidebarPanel(
                    class = "sidebar",
```

```
                    textInput("email", "Enter an email text:", "",
placeholder = "Type your email content here..."),
                    actionButton("predict", "Predict", class = "btn-
primary"),
                    br(),
                    br(),
                    p("Submit your email text and click 'Predict' to
classify it as spam or not spam.")
                ),
                mainPanel(
                    class = "main",
                    div(
                        class = "content",
                        h3("Prediction Result"),
                        h4("This is a machine learning model prediction.
Human crosscheck for strange predictions is advisable."),
                        verbatimTextOutput("result"),
                        style = "padding: 15px;"
                    )
                )
            )
        )
    ),
```

About Tab

```
    tabPanel(
        "About",
        fluidPage(
            tags$head(
                tags$style(HTML(" .main { background-color: #e6e6e6; } "))
            ),
            titlePanel("About This App"),
            mainPanel(
                class = "main",
                div(
                    class = "content",
                    h4("Overview"),
```

```
                    p("This Shiny app uses a k-Nearest Neighbors (kNN)
algorithm to classify emails as spam or not spam."),
                    p("The app demonstrates text preprocessing techniques
and the application of the kNN algorithm for classification."),
                    p("Created using R and Shiny."),
                    br(),
                    h4("Key Features"),
                    tags$ul(
                        tags$li("Simple and interactive interface"),
                        tags$li("Real-time spam classification"),
                        tags$li("Based on kNN algorithm")
                    )
                )
            )
        )
    ),

    # Contact Tab
    tabPanel(
        "Contact",
        fluidPage(
            tags$head(
                tags$style(HTML(" .main { background-color: #e6e6e6; } "))
            ),
            titlePanel("Contact Information"),
            mainPanel(
                class = "main",
                div(
                    class = "content",
                    h4("Get in Touch"),
                    p("For more information, please contact us:"),
                    tags$ul(
                        tags$li("Email: example@example.com"),
                        tags$li("Phone: (123) 456-7890")
                    ),
                    br(),
                    p("We look forward to hearing from you!"),
```

```
                    style = "padding: 15px;"
                )
            )
        )
    )
)
```

Define server logic for the Shiny app

```
server <- function(input, output) {
    observeEvent(input$predict, {
        email_input <- input$email
        email_corpus <- Corpus(VectorSource(email_input))
        email_corpus <- tm_map(email_corpus, content_transformer(tolower))
        email_corpus <- tm_map(email_corpus, removePunctuation)
        email_corpus <- tm_map(email_corpus, removeNumbers)
        email_corpus <- tm_map(email_corpus, removeWords,
stopwords("english"))
        email_corpus <- tm_map(email_corpus, stripWhitespace)
        email_dtm <- DocumentTermMatrix(email_corpus, list(dictionary =
Terms(dtm)))
        email_df <- as.data.frame(as.matrix(email_dtm))
```

Ensure email_df has the same columns as the training data

```
        missing_cols <- setdiff(colnames(train_data), colnames(email_df))
        email_df[missing_cols] <- 0
        email_df <- email_df[, colnames(train_data)[-ncol(train_data)]]
```

Use kNN to predict the label

```
        prediction <- knn(train = train_data[, -ncol(train_data)], test =
email_df, cl = train_data$label, k = k)
        output$result <- renderText({
            paste("The email seems to be", ifelse(prediction == "spam",
"Spam. Better not click any links associated with this email.", "Not Spam.
No need to worry much!"))
        })
    })
}
```

Run the Shiny app

```
shinyApp(ui = ui, server = server)
```

Observations

Performance: The kNN algorithm did quite well in the task of email classification and has accuracy that is close to other methods for the filtering of spam emails.

User Interface: The Shiny web application provided a clear view and user-friendly interface for inputs to be fed into the email text and the classification results to be viewed.

Challenges: Text preprocessing and vectorization were the steps influencing the classifier's performance. Fine-tuning of parameters such as the number of neighbours limited to k was essential in achieving the best results.

Results

The final Email Spam Classifier turned out with the following:

Accuracy: The classifier showed high accuracy in identifying spam and non-spam emails.

User feedback: The first user feedback indicated satisfaction with the usability and simplicity of the application.

There is scope for improvements by using additional machine learning algorithms for comparison, including an advancement of the text preprocessing techniques and generally expanding the dataset in a better way for better generalization.

References

1. *Practical Statistics for Data Scientists: 50 Essential Concepts, Peter Bruce and Andrew Bruce*
2. *Programmed statistics, B. L. Agarwal.*
3. **Internet Sources*