



DATA SCIENCE PROJECT

"The goal is to turn data into information, and information into insight."-Carly Fiorina



by Sunny Mishra 2k17/CS/19

Garima Chawla 2k17/CS/28

Priya Aditya 2k17/CS/34

on May 15th, 2020

to Professor Geetika Vashisth

OUTLINE

1. Introduction
2. Problem statement
3. Approach
4. Life Cycle
 - Data Acquisition
 - Data Preparation
 - Data Modelling
 - Data Evaluation
 - Data Visualisation
5. Conclusion

INTRODUCTION

Data science is an [interdisciplinary](#) field that uses scientific methods, processes, algorithms and systems to extract [knowledge](#) and insights from many structural and [unstructured data](#).

Data science is a "concept to unify [statistics](#), [data analysis](#), [machine learning](#) and their related methods" in order to "understand and analyze actual phenomena" with data. It uses techniques and theories drawn from many fields within the context of [mathematics](#), [statistics](#), [computer science](#), and [information science](#).

In this project, we have incorporated the concepts and techniques of [Data Science](#), [Data Mining](#) and [Machine Learning](#).

PROBLEM

STATEMENT

To demonstrate various approaches of predicting overall sentiments of gathered live tweets for a user-defined Twitter Hashtag.

We have presented two approaches-

A. Model Training

To compare performance of the following techniques of Machine Learning-

1. Support Vector Machine Algorithm (SVM)
2. K-Nearest Neighbour Algorithm (KNN)

using Data Visualisation as well as Evaluation techniques and illustrate the results in a well documented structure.

B. Lexicon Approach

The aim of this is to build a sentiment analysis using the R syuzhet library which will allow us to categorize words based on their sentiments

Approach

Data Visualisation is the essence of a Data Science Project. It is easier to visualise techniques and draw conclusions than deciphering them from spreadsheets / tables.

We have used the **Shiny package in R** and developed a dynamic web page. It provides a very powerful way to share our analysis in an interactive manner with the community.

Shiny: Overview

Shiny is an open package from RStudio, which provides a web application framework to create interactive web applications (visualization) called “Shiny apps”. These web applications seamlessly display R objects (like plots, tables etc.) and can also be made live to allow access to anyone.

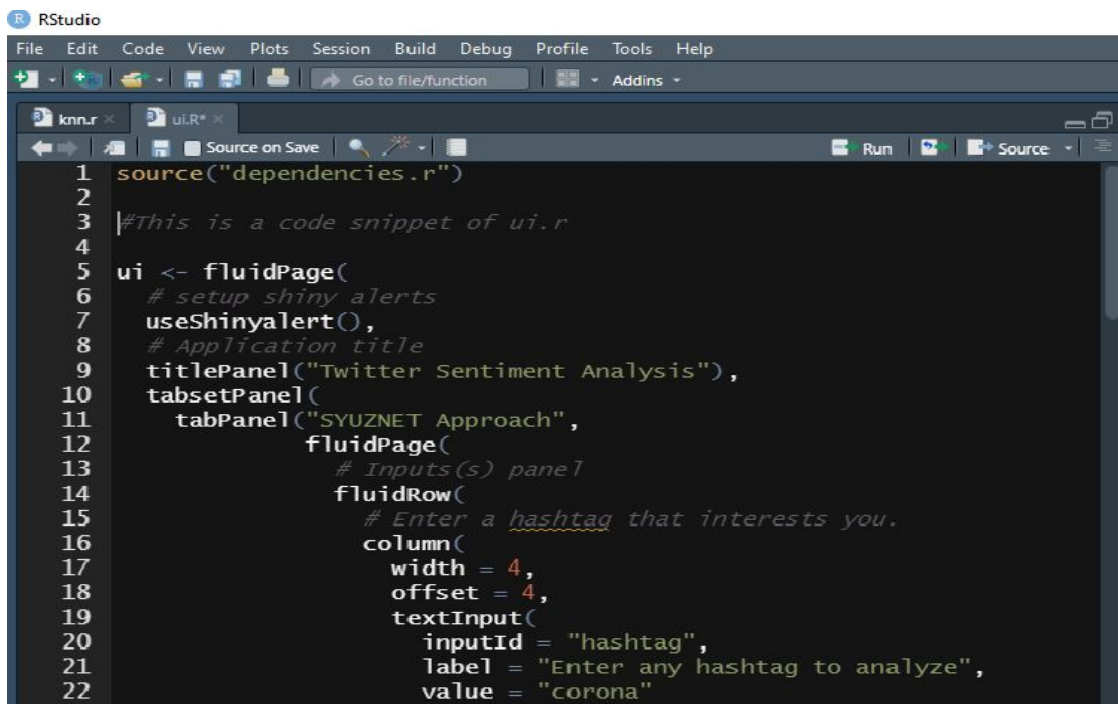
Shiny provides automatic reactive binding between inputs and outputs. It also provides extensive pre-built widgets which make it possible to build elegant and powerful applications with minimal effort.

Any shiny app is built using two components:

1.UI.R: This file creates the user interface in a shiny application. It provides interactivity to the shiny app by taking the input from the user and dynamically displaying

the generated output on the screen. The best way to ensure that the application interface runs smoothly on different devices with different screen resolutions is to create it using fluid page. This ensures that the page is laid out dynamically based on the resolution of each device. The user interface can be broadly divided into three categories:

- **Title Panel:** The content in the title panel is displayed as metadata, as in the top left corner of the above image which generally provides the name of the application and some other relevant information.
- **Sidebar Layout:** Sidebar layout takes input from the user in various forms like text input, checkbox input, radio button input, drop down input, etc. It is represented in the dark background in the left section of the above image.
- **Main Panel:** It is part of the screen where the output(s) generated as a result of performing a set of operations on input(s) at the server.R is / are displayed.



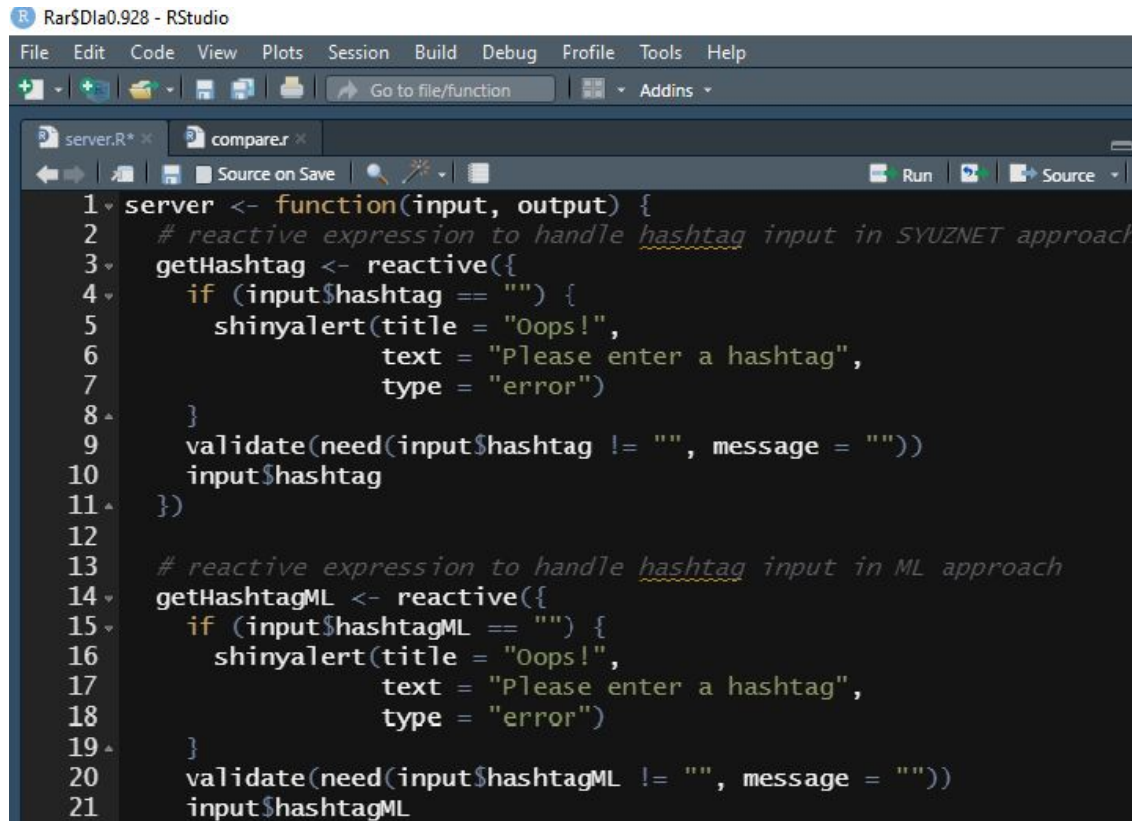
```

1 source("dependencies.r")
2
3 ## This is a code snippet of ui.r
4
5 ui <- fluidPage(
6   # setup shiny alerts
7   useShinyalert(),
8   # Application title
9   titlePanel("Twitter Sentiment Analysis"),
10  tabsetPanel(
11    tabPanel("SYUZNET Approach",
12      fluidPage(
13        # Inputs(s) panel
14        fluidRow(
15          # Enter a hashtag that interests you.
16          column(
17            width = 4,
18            offset = 4,
19            textInput(
20              inputId = "hashtag",
21              label = "Enter any hashtag to analyze",
22              value = "corona"

```

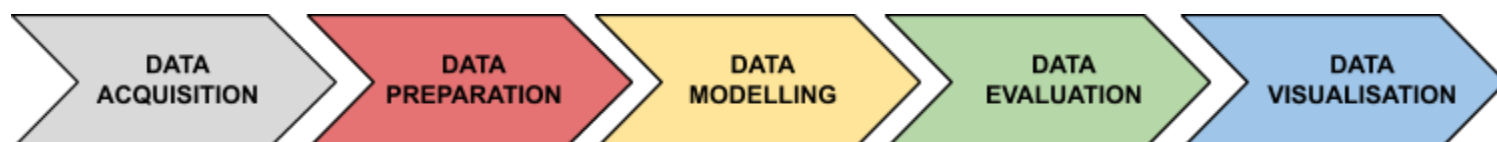
Server.R: This file contains the series of steps to convert the input given by the user into the desired output to be displayed. This acts as the brain of web application. The server.R is written in the form of a function which

maps input(s) to the output(s) by some set of logical operations. The inputs taken in the ui.R file are accessed using \$ operator (input\$InputName). The outputs are also referred using the \$ operator (output\$OutputName).

A screenshot of the RStudio interface. The title bar shows 'Rar\$Dla0.928 - RStudio'. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar has icons for file operations and a 'Go to file/function' search bar. The source editor shows two tabs: 'server.R*' and 'compare.r'. The 'server.R' tab is active, displaying R code. The code defines a 'server' function that takes 'input' and 'output' as arguments. It contains two reactive expressions: 'getHashtag' and 'getHashtagML'. Both expressions check if their respective input is empty. If empty, they trigger a 'shinyalert' with the title 'Oops!' and the text 'Please enter a hashtag', with an 'error' type. They also use 'validate' to ensure the input is not empty. The code is as follows:

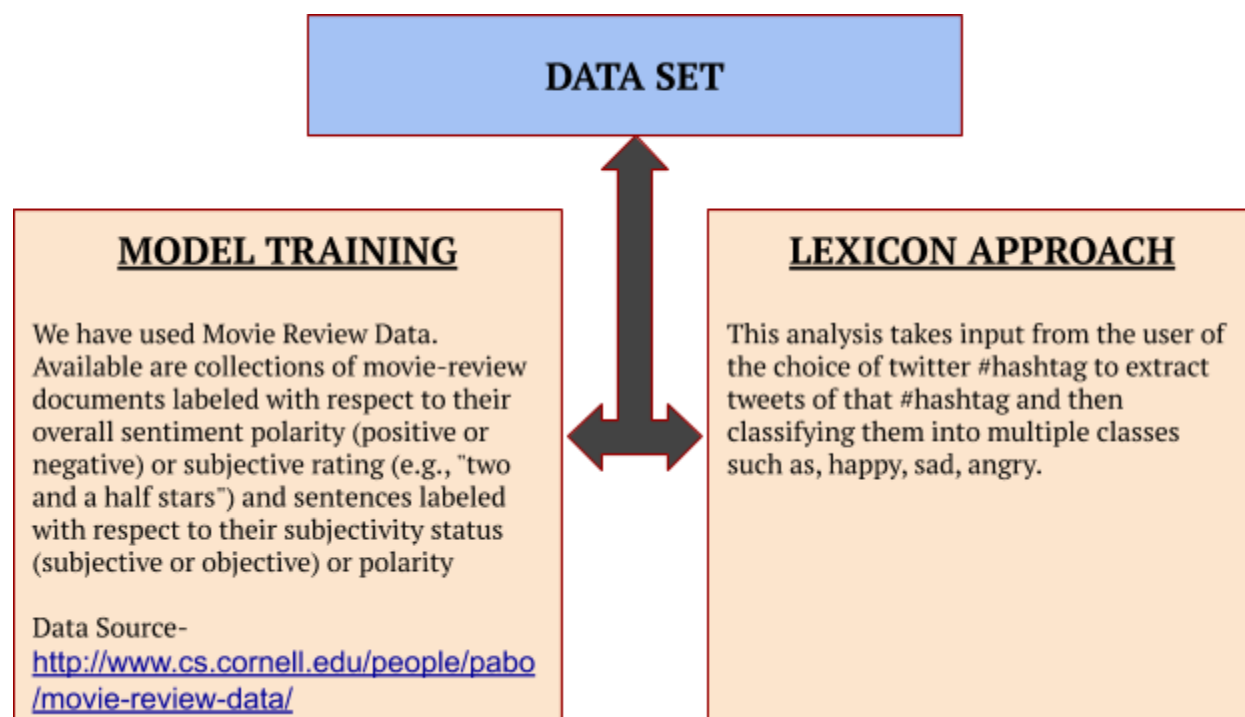
```
1 server <- function(input, output) {  
2   # reactive expression to handle hashtag input in SYUZNET approach  
3   getHashtag <- reactive({  
4     if (input$hashtag == "") {  
5       shinyalert(title = "Oops!",  
6                 text = "Please enter a hashtag",  
7                 type = "error")  
8     }  
9     validate(need(input$hashtag != "", message = ""))  
10    input$hashtag  
11  })  
12  
13  # reactive expression to handle hashtag input in ML approach  
14  getHashtagML <- reactive({  
15    if (input$hashtagML == "") {  
16      shinyalert(title = "Oops!",  
17                text = "Please enter a hashtag",  
18                type = "error")  
19    }  
20    validate(need(input$hashtagML != "", message = ""))  
21    input$hashtagML
```


LIFE CYCLE OF THE PROJECT



I. DATA ACQUISITION:

Data acquisition, or data collection, is the very first step in any data science project. It is important to rightly choose the dataset we wish to work upon. A data set is a collection of data.



II. DATA PREPARATION:

Data acquired in the first step of a data science project is usually not in a usable format to run the required analysis and might contain missing entries, inconsistencies and semantic errors. Therefore, there is a need to clean and reformat the data to make it suitable for further use. This step is usually referred to as [Preprocessing](#). Data preprocessing/preparation/cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset, or and refers to identifying incorrect, incomplete, irrelevant parts of the data and then modifying, replacing, or deleting the dirty or coarse data

The steps used for Data Preprocessing usually fall into two categories:

1. selecting data objects and attributes for the analysis.
2. creating/changing the attributes

Data Preprocessing is a broad area and consists of a number of different strategies:

- Aggregation
- Sampling
- Dimensionality Reduction
- Feature Subset Selection
- Feature Creation
- Discretization and Binarization
- Variable Transformation

Why Data Preprocessing?

Since mistakes, redundancies, missing values, and inconsistencies all compromise the integrity of the set, we need to fix all those issues for a more accurate outcome.

Preprocessing of our datasets

A. Model Training-Movie Reviews Dataset

For the movie reviews data, we undertook the following preprocessing steps to prepare our dataset for data mining:

- [Aggregation](#)- The combining of two or more objects into a single object.

Source Code for Data Aggregation:

```
getMovieSentimentDf <- function() {
  posBase = "./data/txt_sentoken/pos/"
  negBase = "./data/txt_sentoken/neg/"
  posFiles = dir(posBase)
  negFiles = dir(negBase)

  posReviewsVector = vector("character")
  negReviewsVector = vector("character")

  for (pFile in posFiles) {
    posReviewsVector = c(posReviewsVector, read_file(paste(posBase, pFile, sep = "")))
  }

  for(nFile in negFiles) {
    negReviewsVector = c(negReviewsVector, read_file(paste(negBase, nFile, sep = "")))
  }
  # posReviewsVector = sample(posReviewsVector, size = size, replace = FALSE)
  # negReviewsVector = sample(negReviewsVector, size = size, replace = FALSE)
  positiveDf <- data.frame(text = posReviewsVector, sentiment = c("pos"))
  negativeDf <- data.frame(text = negReviewsVector, sentiment = c("neg"))
  reviewDf <- rbind(positiveDf, negativeDf)
  return(reviewDf)
}
```

- [Text Cleaning:](#)

This step involves removal of punctuations, whitespaces, HTML tags, emoticons, URL elements, numbers, special characters, single characters etc. that might be a part of our dataset.

[Source Code for Text Cleaning of the Movie Review Dataset:](#)

```
getCleanMovieSentimentDf <- function () {
  cleanReviewsFilename <- paste("./data/cleanReviews", ".csv", sep="")
  if(file.exists(cleanReviewsFilename)){
    df <- read.csv(cleanReviewsFilename)
    return(data.frame(text=df$text, sentiment=df$sentiment))
  }
  uncleanDf <- getMovieSentimentDf()
  uncleanDf$cleanText <- gsub("(ftp|http(s)*):/.*", "", uncleanDf$text) # remove urls if there any in
the dataset
  uncleanDf$cleanText <- removePunctuation(uncleanDf$cleanText) # remove punctuations
  uncleanDf$cleanText <- replace_internet_slang(uncleanDf$cleanText) # remove internet slang
  uncleanDf$cleanText <- replace_emoji(uncleanDf$cleanText) # replace emoji with their equivalent
words
  uncleanDf$cleanText <- replace_contraction(uncleanDf$cleanText)
  uncleanDf$cleanText <- removeWords(uncleanDf$cleanText, stopwords())
  uncleanDf$cleanText <- gsub(pattern="\\d", replace="", uncleanDf$cleanText) # remove all the
numbers
  uncleanDf$cleanText <- gsub("[[:punct:]]", "", uncleanDf$cleanText) # remove all the special
characters
  uncleanDf$cleanText <- gsub(pattern="\\b[a-z]\\b{1}", "", uncleanDf$cleanText) # remove all single
chars
  uncleanDf$cleanText <- stripWhitespace(uncleanDf$cleanText) # remove whitespaces
  uncleanDf$cleanText <- tolower(uncleanDf$cleanText)
  cleanDf <- data.frame(text=uncleanDf$cleanText, sentiment=uncleanDf$sentiment)
  write.csv(cleanDf, file=cleanReviewsFilename)
  return(cleanDf)
}
```

- [Dimensionality Reduction:](#)

Datasets can have a large number of features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. There are two components of dimensionality reduction: [Feature selection](#) and

Feature Extraction. We have used Feature Selection which is the process of reducing the dimensionality of a dataset by creating new attributes that are a combination of old attributes.

Source Code for Dimensionality Reduction:

```
getTrainingData <- function() {
  df <- getCleanMovieSentimentDf()
  df <- data.frame(text = df$text, s = df$sentiment)
  textCorpus = VCorpus(VectorSource(df$text))
  tdm <- DocumentTermMatrix(textCorpus, control = list(dictionary = getDictionary()))
  trainingMatrix <- as.matrix(tdm)
  trainingDf <- as.data.frame(trainingMatrix)
  trainingDf$s <- df$s
  trainingDf$s <- as.factor(trainingDf$s)
  return(trainingDf)
}
```

B. Lexicon Approach-Live Twitter Data

For the twitter data, we undertook the following preprocessing steps to prepare our dataset for data mining:

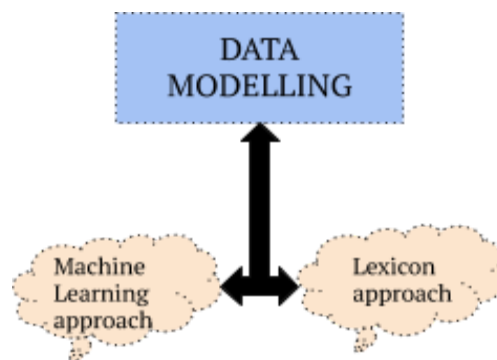
- [Text Cleaning](#):

Source Code for Text Cleaning of Live Twitter Data extracted:

```
cleanText <- function(text) {
  cleanText <- gsub("(ftp|http(s)*):/.?*", "", text) # remove urls if there any in the dataset
  cleanText <- removePunctuation(cleanText) # remove punctuations
  cleanText <- replace_internet_slang(cleanText) # remove internet slang
  cleanText <- replace_emoji(cleanText) # replace emoji with their equivalent words
  cleanText <- replace_contraction(cleanText)
  cleanText <- removeWords(cleanText, stopwords())
  cleanText <- gsub(pattern="\\d", replace="", cleanText) # remove all the numbers
  cleanText <- gsub("[[:punct:]]", "", cleanText) # remove all the special characters
  cleanText <- gsub(pattern="\\b[a-z]\\b{1}", "", cleanText) # remove all single chars
  cleanText <- stripWhitespace(cleanText) # remove whitespaces
  cleanText <- tolower(cleanText)
  return(cleanText)
}
```

III. DATA MODELLING:

This is the core activity of a data science project that requires writing, running and refining the programs to analyse and derive meaningful insights from data. Diverse machine learning techniques are applied to the data to identify the machine learning model that best fits the needs. All the contending machine learning models are trained with the training data sets.



In our project we have used two approaches for data modelling-

- A. Model Training data that implements the [machine learning](#) algorithms [SVM](#) and [KNN](#) on [training dataset](#) of Movie Reviews.
- B. Twitter Sentiment Analysis using the [lexicon based approach](#) applied on [user-defined Twitter Hashtags](#).

A. Model Training- The Machine Learning Approach:

[Highlight of this approach:](#) In this approach, we have trained and compared SVM and KNN classifiers on Movie Reviews Dataset for sentiment prediction and then applied the trained classifier to predict sentiments of live tweets from Twitter.

Steps of this Approach:

- ★ After Data Cleaning, we trained the two models SVM and KNN on Movie Reviews Dataset.
- ★ We then compared the performance of these two models on the basis of their Accuracy and Precision
- ★ On our Shiny Web page, we take input from the user for the Twitter Hashtag (Live Data) and tweets are so collected from Twitter
- ★ Then the Twitter Data is cleaned (as shown in previous steps)
- ★ We then apply the trained models from Step 1 on the live Twitter Data
- ★ The sentiment score of each tweet is then predicted
- ★ To visualise the data, graph is plotted on the number of positive and negative tweets

Source Code for SVM Classifier:

```
getSVMTrainedSentimentAnalysisModel <- function(useCache = TRUE){
  if(file.exists(svmModelCacheFile) && useCache) {
    load(svmModelCacheFile)
    return(fit)
  }
  df <- getTrainingData()
  fit <- train(s~., df, method="svmLinear3")
  save(fit, file=svmModelCacheFile)
  return(fit)
}

svmPredict <- function(text) {
  text <- cleanText(text)
  corpus <- VCorpus(VectorSource(c(text)))
  tdm <- DocumentTermMatrix(corpus, control = list(dictionary = getDictionary()))
  test <- as.matrix(tdm)
  predictions <- predict(getSVMTrainedSentimentAnalysisModel(useCache = TRUE), newdata=test)
  return(predictions)
}
```

Source Code for KNN Classifier:

```
getKNNTrainedSentimentAnalysisModel <- function(useCache = TRUE){
  if(file.exists(knnModelCacheFile) && useCache){
    load(knnModelCacheFile)
    return(fit)
  }
  df <- getTrainingData()
  fit <- train(s~, df, method="knn")
  save(fit, file = knnModelCacheFile)
  return(fit)
}

knnPredict <- function(text) {
  text <- cleanText(text)
  corpus <- VCorpus(VectorSource(c(text)))
  tdm <- DocumentTermMatrix(corpus, control = list(dictionary = getDictionary()))
  test <- as.matrix(tdm)
  predictions <- predict(getKNNTrainedSentimentAnalysisModel(useCache = TRUE), newdata=test)
  return(predictions)
}
```

B. Lexicon Approach-The Syuzhet Package

Highlight of this approach: We have used predefined Syuzhet Library for predicting sentiments of live tweets for user defined Twitter Hashtag.

Steps of this approach:

- ★ We take input from the user on our Shiny Webpage interface
- ★ The tweets are collected from Twitter for the entered Hashtag
- ★ Data Cleaning is performed to prepare the data (as shown in previous steps)
- ★ The Syuzhet Library Functions are applied to the cleaned data to predict sentiments of gathered tweets
- ★ We then plot the various sentiments of the tweets

Source Code for Sentiment Analysis using Syuzhet:

```
sentimentDf <- reactive({
  tweetsText <- tweets()$cleanText
  get_nrc_sentiment(tweetsText)
})
```


IV. DATA EVALUATION:

Evaluating the performance of a data mining technique is a fundamental aspect of machine learning. Determining the efficiency and performance of any machine learning model is hard. The machine learning model will be used for prediction, and for the model to be reliable it is important to choose the right evaluation measure. It is also useful to select the most versatile model or technique. Evaluation measures can differ from model to model. In data mining, classification involves the problem of predicting which category or class a new observation belongs in. The derived model (classifier) is based on the analysis of a set of training data where each data is given a class label. The trained model (classifier)(in our project, Movie Reviews Dataset) is then used to predict the class label for new, unseen data (the live Twitter Data). To understand classification metrics, one of the most important concepts is the **confusion matrix**, shown below:

		Predicted	
		Negative	Positive
Actual	False	True Negative (TN)	False Positive (FP)
	True	False Negative (FN)	True Positive (TP)

Each prediction will fall into one of these four categories. Let's look at what they are.

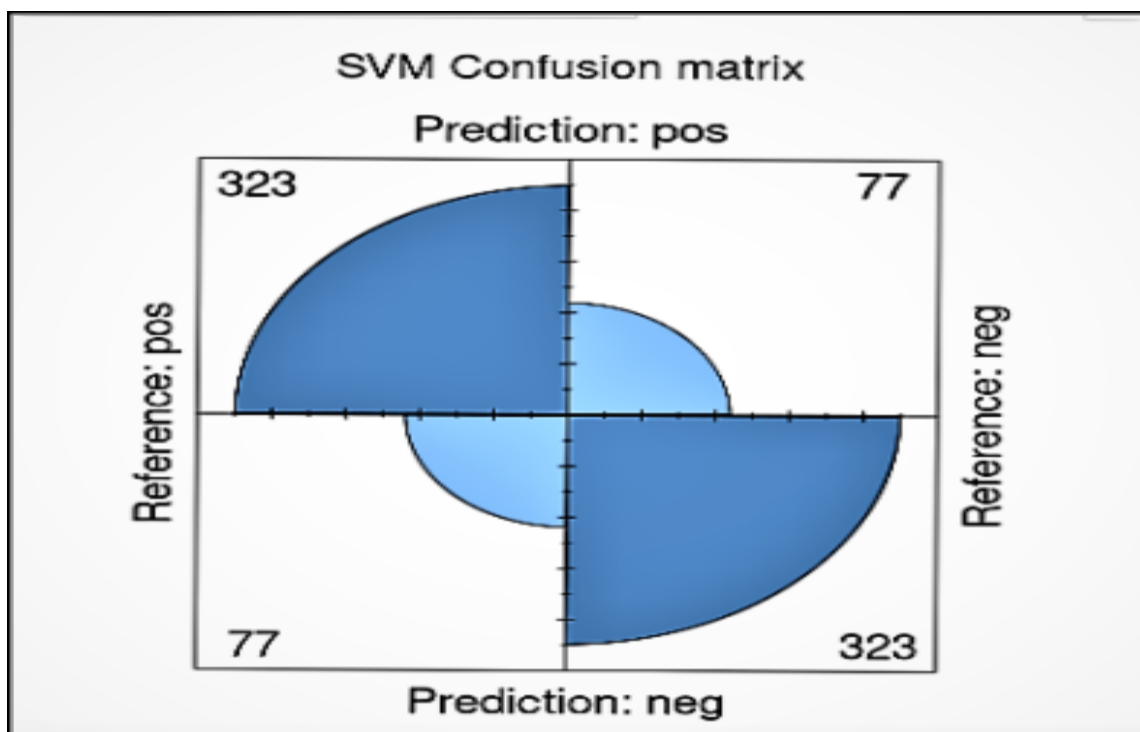
1. **True Negative (TN)**: Data that is labeled false is predicted as false.
2. **True Positive (TP)**: Data that is labeled true is predicted as true.

3. **False Positive (FP)**: Also called "false alarm", this is a type 1 error in which the test is checking a single condition and wrongly predicting a positive.
4. **False Negative (FN)**: This is a type 2 error in which a single condition is checked and our classifier has predicted a true instance as negative.

Source Code for SVM Confusion Matrix:

```
getSVMConfusionMatrix <- function(useCache = TRUE) {
  if(file.exists(svmCMCacheFile) && useCache) {
    load(svmCMCacheFile)
    return(cm)
  }
  df <- getTrainingData()
  trainDataIndex <- createDataPartition(df$s, p=0.6, list = FALSE)
  trainDf <- df[trainDataIndex, ]
  testDf <- df[-trainDataIndex, ]
  fit <- train(s~, data = trainDf, method="svmLinear3")
  predictions <- predict(fit, testDf)
  cm <- confusionMatrix(predictions, testDf$s)
  save(cm, file = svmCMCacheFile)
  return(cm)
}
```

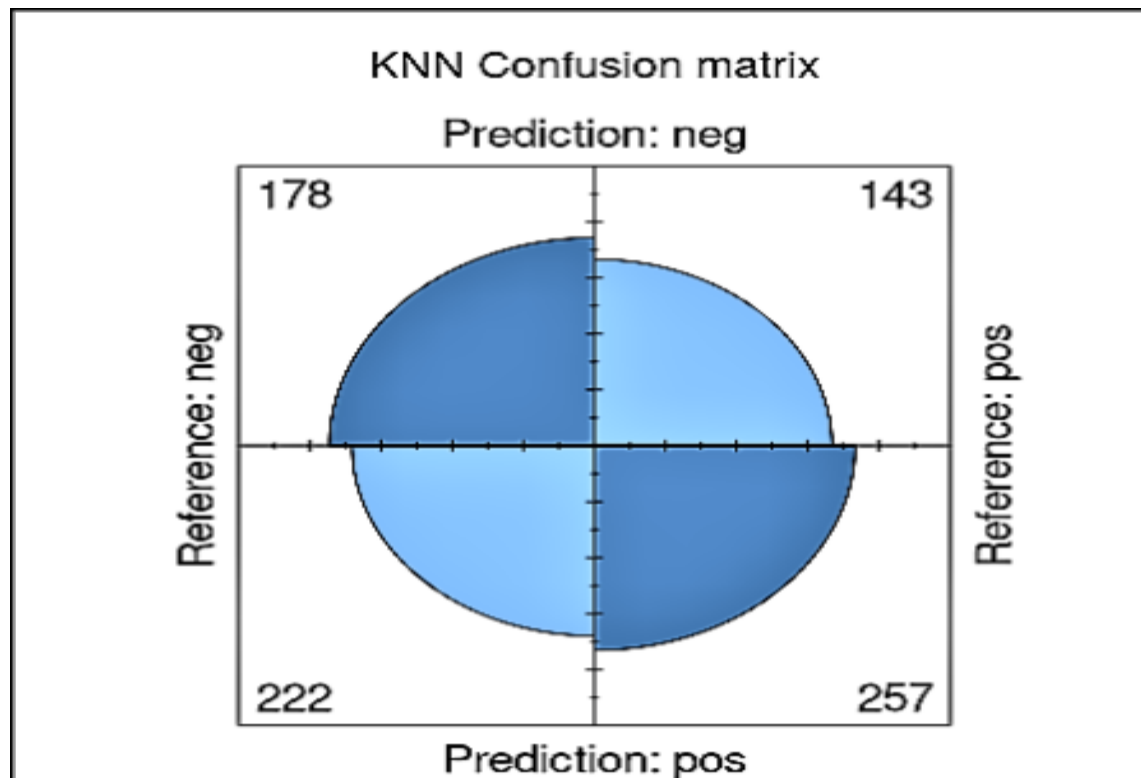
SVM Confusion Matrix:



Source Code for KNN Confusion Matrix:

```
getKNNConfusionMatrix <- function(useCache = TRUE) {
  if(file.exists(knnCNCacheFile) && !useCache) {
    file.remove(knnCNCacheFile)
  }
  if(file.exists(knnCNCacheFile) && useCache) {
    load(knnCNCacheFile)
    return(cm)
  }
  df <- getTrainingData()
  trainDataIndex <- createDataPartition(df$s, p=0.6, list = FALSE)
  trainDf <- df[trainDataIndex,]
  testDf <- df[-trainDataIndex,]
  set.seed(1337)
  fit <- train(s~, trainDf, method="knn")
  predictions <- predict(fit, testDf)
  cm <- confusionMatrix(predictions, testDf$s)
  save(cm, file = knnCNCacheFile)
  return(cm)
}
```

SVM Confusion Matrix:



Evaluation metrics

Accuracy-

The accuracy of a classifier is given as the percentage of total correct predictions divided by the total number of instances.

Mathematically,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

$$\begin{aligned}\text{Accuracy for SVM Classifier} &= (323+323)/(323+77+323+77) \\ &= 80.75\%\end{aligned}$$

$$\begin{aligned}\text{Accuracy for KNN Classifier} &= (257+178)/(257+143+178+222) \\ &= 54.37\%\end{aligned}$$

Recall-

Recall is one of the most used evaluation metrics for an unbalanced dataset. It calculates how many of the actual positives our model predicted as positives (True Positive).

Recall is also known as true positive rate (TPR), sensitivity, or probability of detection.

Mathematically,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Recall for SVM Classifier} = 323/(323+77)$$

$$= 80.75\%$$

$$\text{Recall for KNN Classifier} = 257/(257+222)$$

$$= 53.65\%$$

Precision-

Precision describes how accurate or precise our data mining model is. Out of those cases predicted positive, how many of them are actually positive.?

Precision is also called a measure of exactness or quality, or positive predictive value.

Mathematically,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Precision for SVM Classifier} = 323/(323+77)$$

$$= 80.75\%$$

$$\text{Precision for KNN Classifier} = 257/(257+143)$$

$$= 64.25\%$$

F1 Score

When both recall and precision are necessary, then the F1 score comes into the picture. It tries to balance out both recall and precision. Remember, it is still better than accuracy, as with an F1 score we are not looking for any true negative data.

Mathematically, it is defined as a harmonic mean of recall and precision:

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{F1 Score for SVM Classifier} = (2 \times 0.8075 \times 0.8075)/(0.8075 + 0.8075)$$

$$= 80.74\%$$

$$\text{F1 Score for KNN Classifier} = (2 \times 0.6425 \times 0.5365)/(0.6425 + 0.5365)$$

$$= 58.4\%$$

V. DATA VISUALISATION:

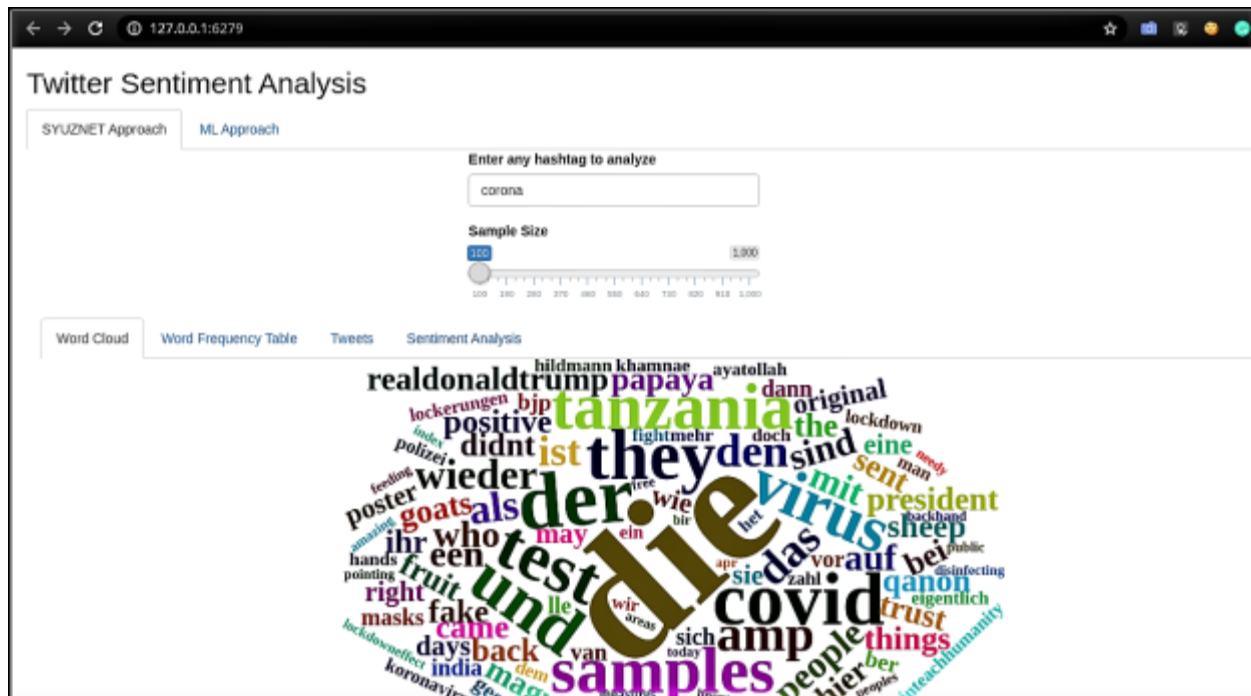
Data visualization is the graphical representation of the data and information extracted from data mining using the visual elements like graph, chart, and maps, data visualization tool, and techniques helps in analyzing massive amounts of information and make decisions on top of it. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It's storytelling with a purpose.

With the use of [Shiny Webpage](#), we have created a very powerful way to share our analysis in an interactive manner with the community.

It is a dynamic webpage which takes input from the user for the Twitter Hashtag. It displays two concepts-

- a. Model Training-Machine Learning Approach
- b. Lexicon Approach- Using the Syuzhet Package

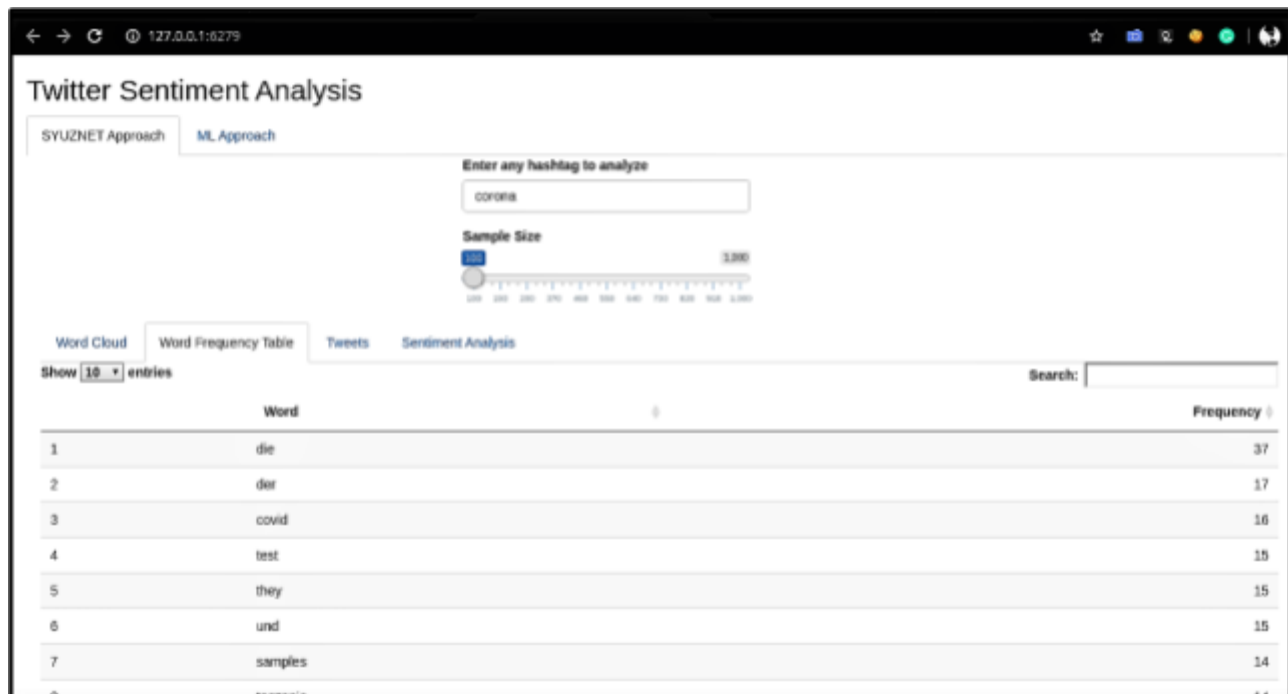




WORD FREQUENCY TABLE

Source Code for displaying the Word frequency table:

```
output$frequencyTable <- renderDataTable(tweetsFreqDf())
```



TWEETS

Source Code for displaying tweets:

```
output$tweets <- renderDataTable({
  df <- tweets()
  compact <-
    data.frame(
      date = df$created_at,
      userid = df$screen_name,
      tweet = df$text
    )
  compact
})
```

Twitter Sentiment Analysis

SYUJNET Approach ML Approach

Enter any hashtag to analyze

corona

Sample Size

100 3280

Word Cloud Word Frequency Table Tweets Sentiment Analysis

Show 10 entries

Search:

	date	userid	tweet
1	2020-05-10T16:49:13Z	iremcicekci	#human #HumanResources #pazar #yangtay #kurallar #annelergunukufutuban #COVID #Corona https://t.co/TVvVjfyw
2	2020-05-10T16:49:13Z	tokitaderos	#GeçiciVergiligin Uzama Sirkulerine engel olan neden: #Corona sürecinde de ciddi gelirler elde etmiş, hatta ek faydalar sağlamış Sektörlerin bulunması. Onlar için bir formül bulunmazsa geriye bir hafta geçecekleri kesin. Umarım Bu sektörler için bulunacak formül karmaşayı artırmaz.
3	2020-05-10T16:49:11Z	JustDeacon	#Tanzania #Magufuli #Corona #test #gapan #realDonaldTrump The President of Tanzania didn't trust the #WHO, so he had fake test samples sent to laboratories. They were samples of papaya fruit, sheep, goats, and other things. They all came back positive for the Corona virus. https://t.co/jp713ufU8
4	2020-05-10T16:49:06Z	Namrata86813190	कहीं #Corona #lockdowneffect ने हो रहा है कि जितने कार्टून के जवान हैं वैसे ही दिख रहा है, वो वास्तव जगह में वास्तव बर्बाद कर रहे हैं. घर बापूजी के बच्चे, यहाँ रहते बच्चे। देखिये कैसे दलेशा, जितना घर नोटिफिकेशन मुझ तक दिन्नाब कुछ नोटिफिकेशन का संस्तर पाठ कर रहा- अगर घर छोड़ना संस्तर https://t.co/dp443Tg4M
5	2020-05-10T16:49:05Z	UKordeck	#Mannheim: Probleme bei der Durchsetzung der #Corona-#Abstandsregeln. 250 Feiernde mit Party am Samstag Abend; einige haben Polizei mit Böllern beworfen. #Covid19 https://t.co/gpCFGd2x0

SENTIMENT ANALYSIS USING LEXICON APPROACH

Source Code for plotting Sentiment Analysis:

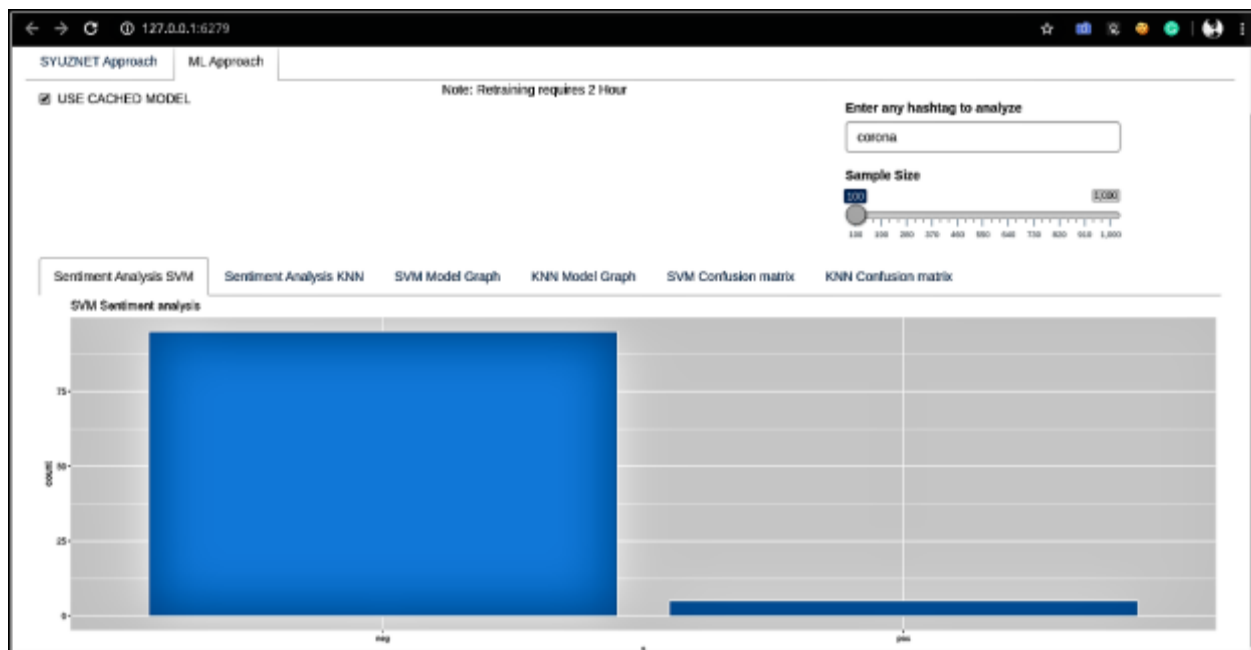
```
output$sentimentAnalysisPlot <- renderPlot({
  df <- sentimentDf()
  barplot(
    colSums(df),
    las = 2,
    col = rainbow(10),
    ylab = "Count",
    main = paste(
      "Sentiment Scores for",
      paste("#", getHashtag(), sep = ""),
      "Tweets"
    )
  )
})
```



SENTIMENT ANALYSIS USING ML APPROACH-SVM

Source Code for plotting Sentiment Analysis:

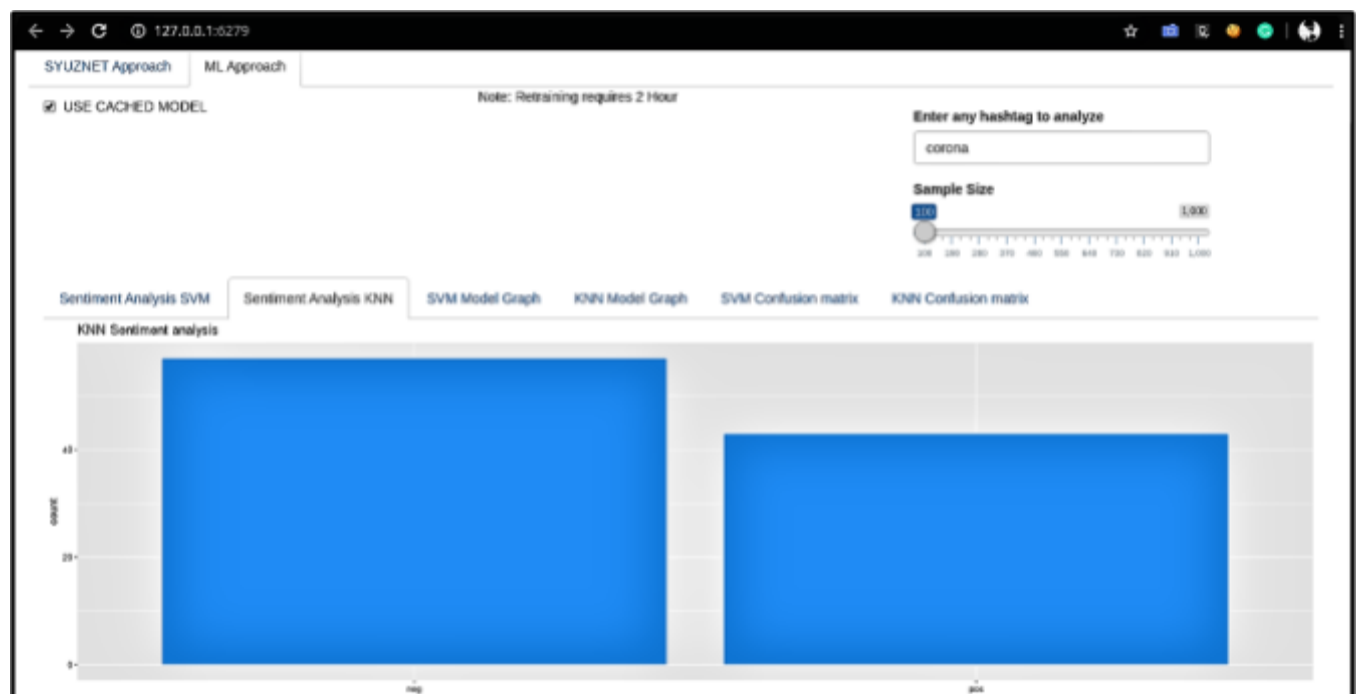
```
output$mlSentimentPlotSVM <- renderPlot({
  p <- shiny::Progress$new()
  p$set(message = "Gathering Tweets and analyzing sentiment")
  on.exit(p$close())
  tweetsDf <- tweetsML()
  tweetsDf$s <- svmPredict(tweetsDf$text)
  ggplot(data = tweetsDf) + labs(title = "SVM Sentiment analysis",
    xlabel = "Sentiment",
    ylabel = "Count") + geom_histogram(
    mapping = aes(x = s,),
    fill = "#42a5f5",
    color = "white",
    stat = "count"
  )
})
```



SENTIMENT ANALYSIS USING ML APPROACH-KNN

Source Code for plotting Sentiment Analysis:

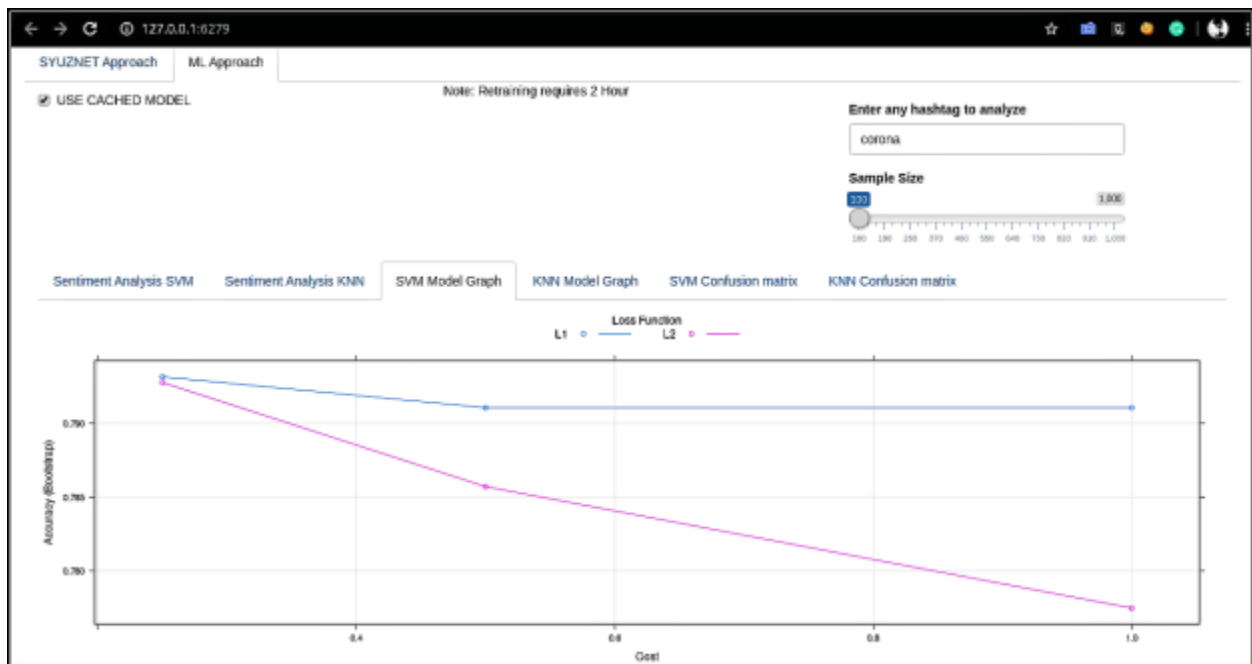
```
output$mlSentimentPlotKNN <- renderPlot({
  p <- shiny::Progress$new()
  p$set(message = "Gathering Tweets and analyzing sentiment")
  on.exit(p$close())
  tweetsDf <- tweetsML()
  tweetsDf$s <- knnPredict(tweetsDf$text)
  ggplot(data = tweetsDf) + labs(title = "KNN Sentiment analysis",
    xlabel = "Sentiment",
    ylabel = "Count") + geom_histogram(
    mapping = aes(x = s),
    fill = "#42a5f5",
    color = "white",
    stat = "count",
  )
})
```



SVM MODEL GRAPH

Source Code for plotting SVM Model Graph:

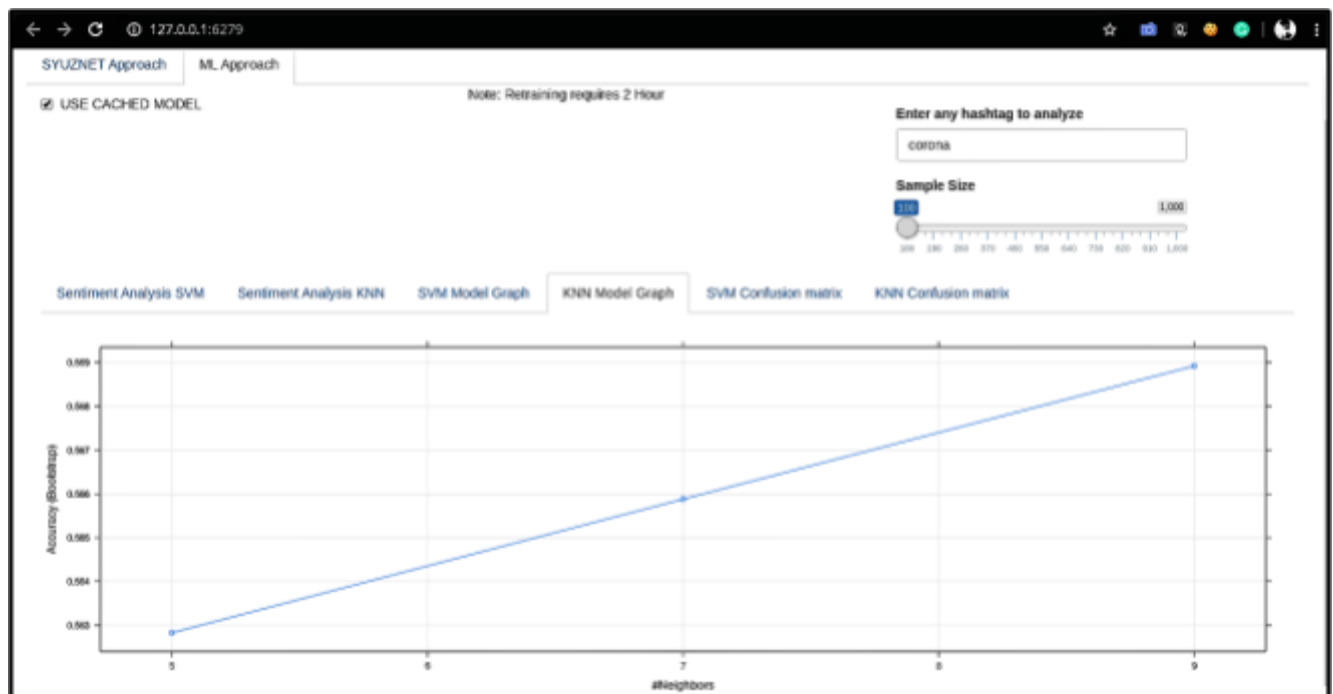
```
output$svmPlot <- renderPlot({
  p <- shiny::Progress$new()
  p$set(message = "Training model and plotting graph")
  on.exit(p$close())
  plot(getSVMTrainedSentimentAnalysisModel(useCache = input$useCache))
})
```



KNN MODEL GRAPH

Source Code for plotting KNN Model Graph:

```
output$sknnPlot <- renderPlot({
  p <- shiny::Progress$new()
  p$set(message = "Training model and plotting graph")
  on.exit(p$close())
  plot(getKNNTrainedSentimentAnalysisModel(useCache = input$useCache))
})
```



CONCLUSION

COMPARISON BETWEEN SVM AND KNN BASED ON EVALUATION METRICS		
METRIC	SVM	KNN
Accuracy	80.75%	54.37%
Recall	80.75%	53.65%
Precision	80.75%	64.25%
F1 Score	80.74%	58.45%

From the above comparison table, we can conclude that **SVM** is a more efficient classifier to train the model data in terms of its Accuracy, Recall, Precision and F1 Score and speed.

Link to our Webpage-

<https://mishrasunny174.shinyapps.io/twitterdashboard/>