

# Twitter Sentiment Streaming Analysis

Website Link: <https://agarwal123.shinyapps.io/LiveAnalysisTwitter/>

# Final Project: Twitter Sentiment Streaming Analysis

## Contents

Problem statement.....	2
About Dataset .....	2
Data Preparation & Cleansing.....	2
Dashboard 1:.....	5
Dashboard 2:.....	16
Dashboard 3:.....	18
Dashboard 4:.....	18
Android App: .....	34
Website: .....	36
References	

**Problem statement:** Perform Twitter sentiment live stream analysis and classify the sentiment of a given text further analyzing the sentiments or emotions of people towards the entity.

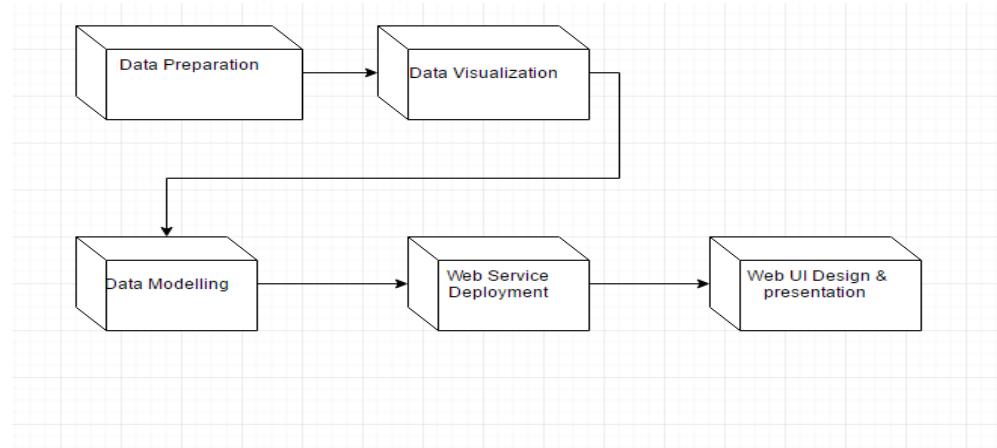
### About Dataset:

The dataset contains static and dynamic tweets, which was collected by using the Twitter Search API and keywords search.

Static Data: Twitter tweets were collected for keywords “iphone” and “ipad”

Live Data: Twitter tweets were continuously collected for dynamic keywords inputted by user using streaming API

### Basic Deployment Pipeline:



### Data Preparation & Cleansing

Tools Used: R script was used for preprocessing and cleaning the data.

Steps:

1. Find most popular Key words, and get the static data for training model.
2. Register a Twitter Account at <http://twitter.com>.

### 3. Create a Twitter Application at <http://apps.twitter.com>.

#### Create an application

Application Details

Name \*

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description \*

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website \*

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request here. To restrict your application from using callbacks, leave this field blank.

### 4. Twitter's Streaming API OAuth credentials

- On the Twitter application page, click on the Keys and Access Tokens tab.
- Consumer Key(API Key) and Consumer Secret(API Secret) can be found under Application Settings.
- Under Your Access Token section, click on Create my access token to obtain both Access Token and Access Token Secret.

Application Settings	
Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.	
Consumer Key (API Key)	d24llsrcWzA5IQnO4ITBGUBHQ
Consumer Secret (API Secret)	gkUlc91hr34BfnHhgE86zEEp2yqe87aGV3tVN4SNW0Pabez8I
Access Level	Read and write ( <a href="#">modify app permissions</a> )
Owner	Calvinle89
Owner ID	806234759437385728

Application Actions	
<a href="#">Regenerate Consumer Key and Secret</a>	<a href="#">Change App Permissions</a>
<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>	
Your Access Token	
This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.	
Access Token	806234759437385728-6YlowuhV2hCWQ4oHDKEBWkf6ILRwxy0
Access Token Secret	uTV7uEQMeAf8tc4AWVGzzHx3fgJFyNUXKrZMBsENsbI68
Access Level	Read and write
Owner	Calvinle89
Owner ID	806234759437385728

### Building Static data:

5. Using R script to get static data to build model by Key words.

```

library(twitteR)
consumerKey <- "Ev1sh2xxP5gHaKrEEp0lFYi3B"
consumerSecret <- "NmUlPoEb15zrqbv0bj80NdRwigCrRUE4hQ2pwXlMA9K1pXvk8P"
oauth_token <- "806316172996214784-tCXVHP6InzRo3vhfLck2Z5eH3dvKLTw"
oauth_token_secret <- "0c64vL7zmLFpYzLwakfExfJQTytcI4M65Jrfmd9hubsBq"

setup_twitter_oauth(consumerKey, consumerSecret, oauth_token, oauth_token_secret)

tweets1 <- searchTwitter('#iPhone', lang="en", n=500000, since='2016-12-01', until = '2016-12-12')
tweets2 <- searchTwitter('#iPad', lang="en", n=500000, since='2016-12-01', until = '2016-12-12')

tweets_df1 <- twListToDF(tweets1)
tweets_df2 <- twListToDF(tweets2)

```

6. Clean the data by using R script to normalize the original data.

- a. Delete needless symbols of “Text” (tweets).
- b. Calculate sentiments by using functions.
- c. Count all mapped positive and negative words and subtract the latter from the former

```

#clean tweets
text = gsub('[^[:graph:]]', ' ', as.character(tweets_df2$text))
text = gsub(' +', ' ', text)
text = gsub(' +'$, ' ', text)
text = gsub(' +', ' ', text)
tweets_df2$text = text

#calculate sentiment function
get_sentiment = function(txt) {
  words = strsplit(txt, '+')
  words = unlist(words)

  pos_matches = match(words, pos)
  neg_matches = match(words, neg)

  #count all the mapped positive and negative words and subtract the latter from the former
  score = sum(is.na(pos_matches)) - sum(is.na(neg_matches))
  return(score)
}

#estimate sentiment of the tweets
text = gsub('[^[:alpha:]]', ' ', text)
text = tolower(text)

tweets_df2$sentiment = sapply(text, get_sentiment)

#tweets_df1 <- select(tweets_df1, -score)
tweets_df2 <- tweets_df2 %>% mutate(score = ifelse(sentiment > 2, 4, ifelse(sentiment < -2, 0, 2)))

iphone <- select(tweets_df1, text, score)
ipad <- select(tweets_df2, text, sentiment, score)
write.csv(iphone, "iphone.csv")
write.csv(ipad, "ipad1.csv")

```

### Building Live data:

7. Live data is collected using filterStream giving a timeout = 10sec

```

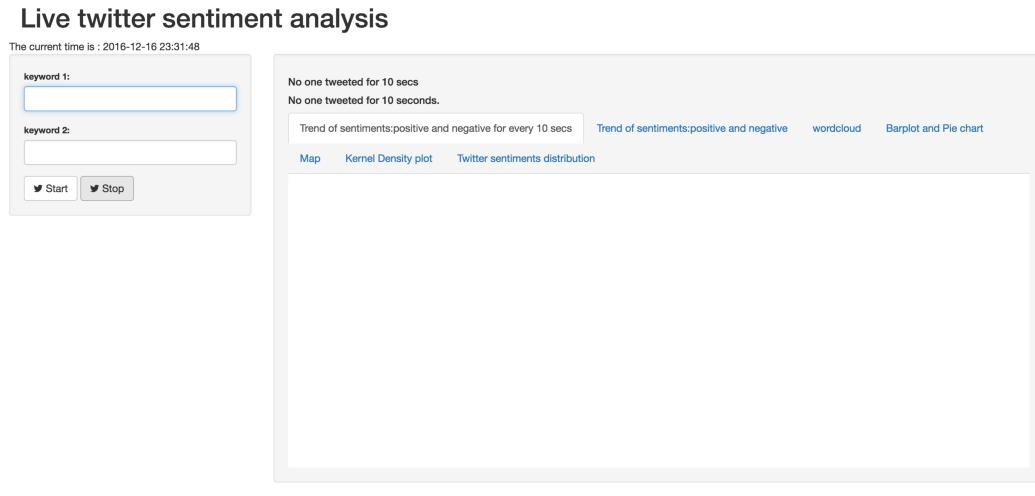
+ getAllTweets = function(searchTerm) {

  track    <- searchTerm
  print(track)
  print(searchTerm)
  #new tweets after every 10 sec
  tweets = filterStream(file.name = '', track=track,language = 'en', timeout = 10, oauth = my_oauth)
}

```

## Dashboard 1:

**Purpose:** Perform **LIVE** sentiment analysis on two different keywords and perform **LIVE** visualizations in order to determine negative and positive sentiments across various areas



**Description:** This can be rather called a product as it accepts two dynamic keywords. Eg: Trump and Hillary. It basically retrieves live tweets and one can determine sentiments and emotions of people regarding the two politicians. Tweet feeds are refreshed every 10 secs and the visualizations are also refreshed every 10 secs.

**Enter keywords here:** As soon as start button is pressed, live tweets are collected until 10 secs and visualizations are plotted.

The current time is : 2016-12-16 23:37:26

keyword 1:  
trump

keyword 2:  
hillary

Start   Stop

**Sample positive and negative tweets:** Live sample of tweets are displayed. Huge tweets data are analyzed and cleaned. Once cleaning is done, sentiment score is calculated and recognized into: “Positive”, “negative”, “neutral”

From a list of positive and negative tweets, one tweet is chosen at random and displayed on dashboard

```
trump
positive tweet: I love how since Trump lost everyone thinks the election was hacked
negative tweet: Twitter recap: Trump's reckless to deny Russia role. Obama's detached for lack of anger at Russia. HRC can never cite Rus...
hillary
positive tweet: Bill Clinton was called President of U.S.A for he won ELECTORAL VOTES and he sworn in ... according to CONSTITUTION.
negative tweet: You literally had political hacks writing pro hillary propaganda for you! You are a demonic little wic...
```

**clean tweets:**

```
#clean tweets
text = gsub('[^[:graph:]]', ' ', as.character(tweets$text))
text = gsub('^\+', ' ', text)
text = gsub(' +$', ' ', text)
text = gsub(' +', ' ', text)
text <- gsub("http://t.co/[a-z,A-Z,0-9]*{8}", "", text)
text <- gsub("https://t.co/[a-z,A-Z,0-9]*{8}", "", text)
text <- gsub("RT @*[a-z,A-Z]*:", " ", text)
text <- gsub("#*[a-z,A-Z]*", " ", text)
text <- gsub("@*[a-z,A-Z]*", " ", text)
tweets$text = text
text = gsub('[^[:alpha:]]', ' ', text)
text = tolower(text)

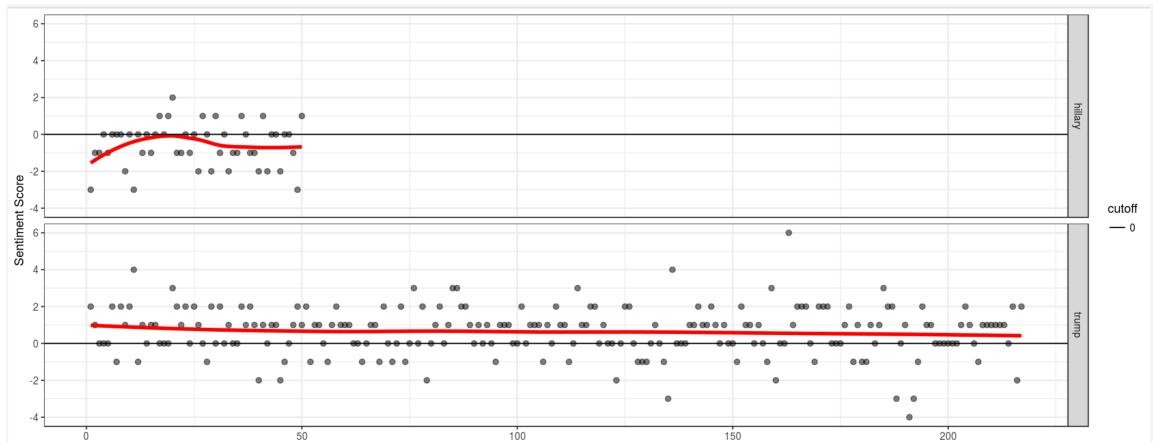
tweets$sentiment = sapply(text, calculateSentiment)
```

**Calculate sentiment score:**

```
#calculate sentiment score
calculateSentiment = function(txt) {
  words = strsplit(txt, ' +')
  words = unlist(words)
  positiveWords = match(words, pos)
  negativeWords = match(words, neg)
  score = sum(!is.na(positiveWords)) - sum(!is.na(negativeWords))
  return(score)
}
```

**Trend of sentiments - positive and negative for every 10 secs:** Every 10 secs, sentiment score for tweets is calculated and displayed as below:

**Observation:** There are more tweets for Trump as compared to Hillary. Also, Hillary has more tweets with negative sentiments. Trump has received more positive and more number of tweets. This shows that sentiments of people towards Trump is more positive than Hillary



Plot: geom\_smooth is used for ggplot. It is called as smoothed conditional means. It helps in viewing the patterns in the presence of overplotting.

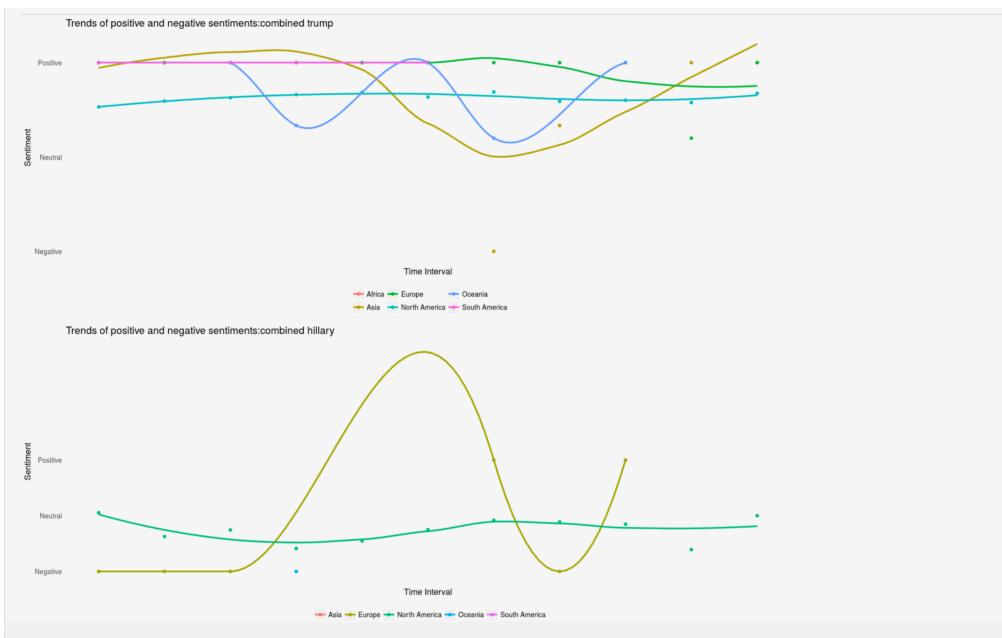
Method:loess – local polynomial regression fitting.

It fits a polynomial surface determined by one or more numerical predictors, using local fitting.

```
cutoff <- data.frame(yintercept=0, cutoff=factor(0))
boxPlot<-ggplot(keywordScore1(),aes(x=size,y=score))+ 
  facet_grid(entity ~ .)+ 
  geom_point(color = "black",size = 2, alpha = 1/2)+ 
  geom_smooth(method = "loess",se=FALSE,col='red',size=1.5, alpha = 0.7)+ 
  geom_hline(aes(yintercept=yintercept, linetype=cutoff), data=cutoff)+ 
  xlab('Tweet number')+ 
  ylab('Sentiment Score')+ 
  theme_bw()
```

**Trends of positive and negative sentiments - combined:** Every 10 secs, sentiment score for tweets are calculated. The score is summed up for each 10 secs and plot is displayed from start time till present time. Each line represents the sentiments of people from each continent. Hence, one can understand emotions of people distributed among continents.

**Observation:** Sentiments of people for trump is more towards positive in each continent, while it keeps fluctuating in case of Asia and Oceania. Sentiments of people for Hillary are mostly negative, then neutral then positive in Europe while neutral in case of North America.



Plot: `stat_smooth` is used for `ggplot`. It is called as smoothed conditional means.

Stat	Description	Default Geom
<code>stat_bin()</code>	Counts the number of observations in bins.	<code>geom_bar()</code>
<code>stat_smooth()</code>	Creates a smooth line.	<code>geom_line()</code>
<code>stat_sum()</code>	Adds values.	<code>geom_point()</code>
<code>stat_identity()</code>	No summary. Plots data as is.	<code>geom_point()</code>
<code>stat_boxplot()</code>	Summarizes data for a box-and-whisker plot.	<code>geom_boxplot()</code>

The interesting thing about `stat_smooth()` is that it makes use of local regression by default. R has several functions that can do this, but `ggplot2` uses the `loess()` function for local regression. This means that if you want to create a linear regression model you have to tell `stat_smooth()` to use a different smoother function.

```

dat_all$seq = match(dat_all$ts_r, sort(unique(dat_all$ts_r)))
dat_all = arrange(dat_all, seq)

#calculate % of positive sentiment
sentiment_prop = dat_all %>%
  group_by(continent, seq) %>%
  summarise(sentiment_prc = totalSentiment(sentiment))

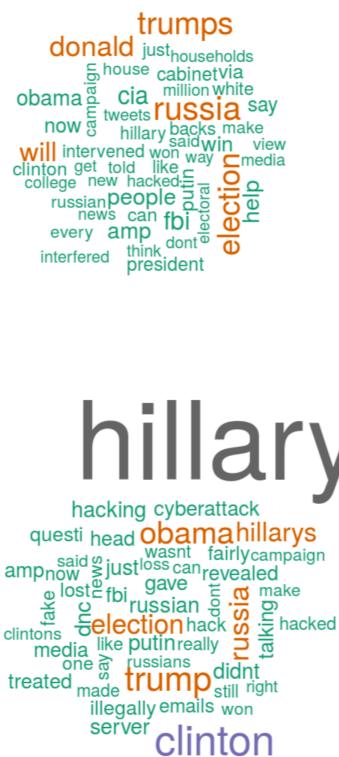
trend_plot = ggplot(sentiment_prop, aes(x = seq, y = sentiment_prc, col = continent)) +
  geom_point() + stat_smooth(method = 'loess', se = F) +
  guides(col = guide_legend(title = NULL)) +
  scale_y_continuous(breaks = seq(0, 1, .25),
                     labels = c('Negative', '', 'Neutral', '', 'Positive')) +
  theme(legend.position = 'bottom', axis.text.x = element_blank(),
        panel.grid = element_blank(), panel.border = element_blank(),
        axis.ticks = element_blank(),
        panel.background = element_blank(), plot.background = element_blank(),
        legend.background = element_blank()) +
  xlab('Time Interval') + ylab('Sentiment') +
  ggtitle(paste0(title,a))

return(trend_plot)

```

**Wordcloud:** Word cloud is a text mining method that allows us to highlight the most frequently used keywords in a paragraph of texts. It is also referred to as a text cloud or tag cloud. Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud.

**Observation:** For trump, frequently used words are ‘Russia’ and ‘election’. Other words are surrounded around. For Hillary, ‘obama’ is most frequently used.



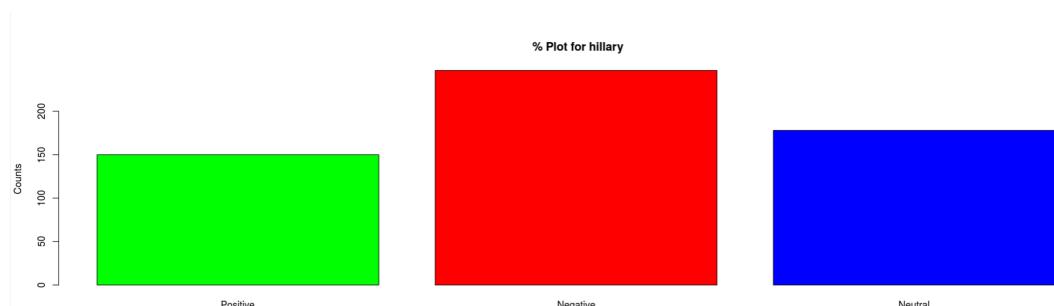
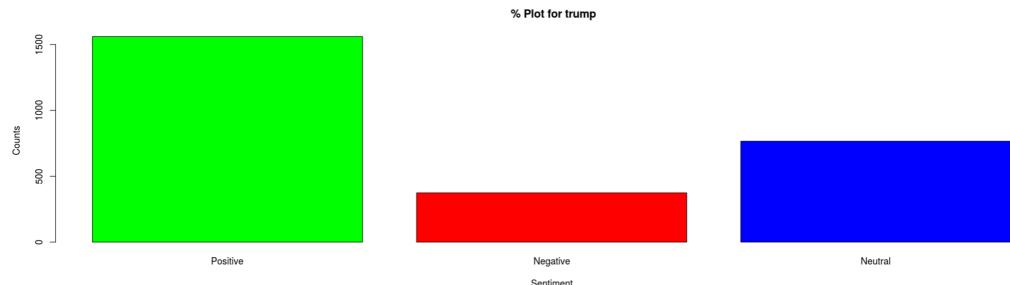
```

wordcloudentity<-function(text,a)
{
  text = gsub('http', ' ', text)
  text = gsub('https',' ',text)
  tweetCorpus<-Corpus(VectorSource(text))
  tweetTDM<-TermDocumentMatrix(tweetCorpus,control=list(removePunctuation=TRUE,
                                                       stopwords=c(stopwords('english')),
                                                       removeNumbers=TRUE,tolower=TRUE))
  tdMatrix <- as.matrix(tweetTDM)
  sortedMatrix<-sort(rowSums(tdMatrix),decreasing=TRUE)
  cloudFrame<-data.frame(word=names(sortedMatrix),freq=sortedMatrix)
  wcloudentity<-wordcloud(cloudFrame$word,cloudFrame$freq,max.words=50, colors=brewer.pal(8,"Dark2"),scale=c(8,1), random.order=TRUE)
  print(wcloudentity)
}

```

**Bar Charts and Pie charts:** A bar chart or bar graph is a chart or graph that presents grouped data with rectangular bars with lengths proportional to the values that they represent. Pie chart is a type of graph in which a circle is divided into sectors that each represents a proportion of the whole

**Observation:** For trump, positive percentage of tweets is 58%, negative: 14%, neutral: 28%. For Hillary, positive: 26%, 43%, 31%. Bar graph visualizes the count of type of tweets



## Bar graph:

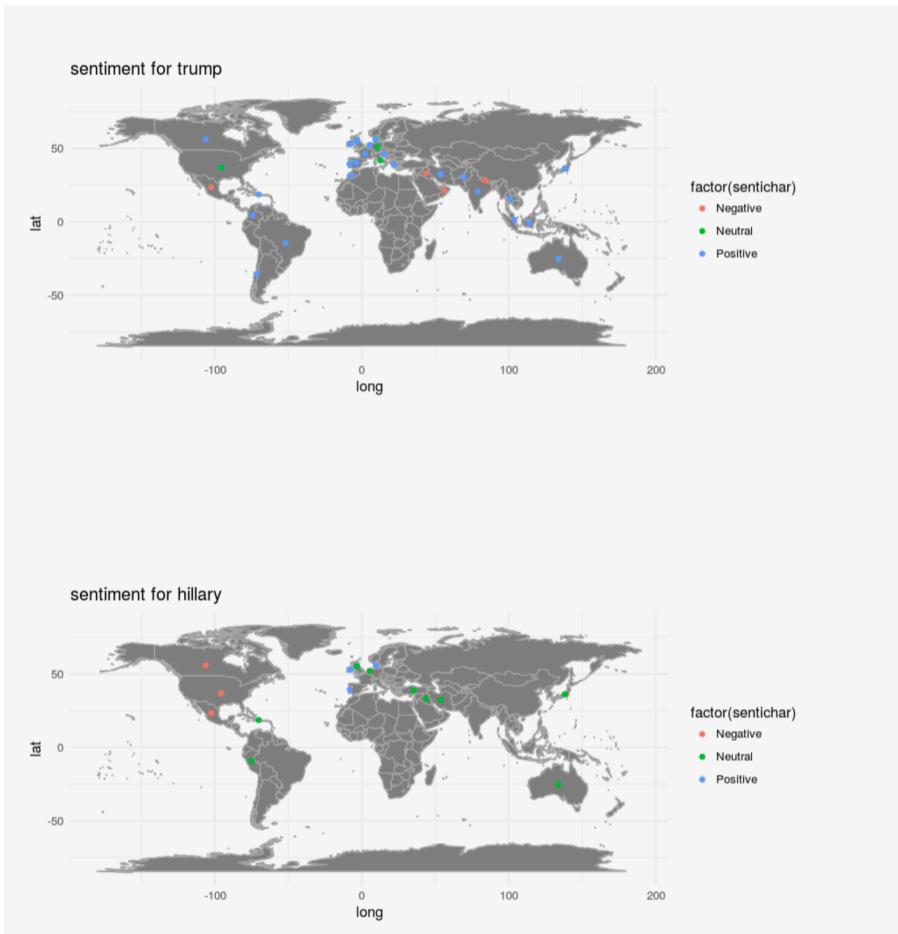
```
results1 = data.frame(tweets = c("Positive", "Negative", "Neutral"), numbers
                      = c(positivecount1,negativecount1,neutralcount1))
barplot(results1$numbers, names =
         results1$tweets, xlab = "Sentiment", ylab = "Counts",
         col = c("Green","Red","Blue"), main = paste0("% Plot for ",input$keyword1))
```

## Pie chart:

```
results2 = data.frame(tweets = c("Positive", "Negative", "Neutral"), numbers = c(positivecount2,negativecount2,neutralcount2))
pct2 <- round(results2$numbers/sum(results2$numbers)*100)
label2 <- paste(results2$tweets,pct2)
label2 <- paste(label2, "%", sep = "")
pie3D(results2$numbers,labels=label2,explode=0.1,col = c("Green","Red","Blue"), main=
       paste0("Pie Chart of percentage sentiments ",input$keyword2))
```

**Map:** Plots world map and views distribution of tweet sentiments in each continent.

**Observation:** For trump, there are more positive tweets in North America. For Hillary, there are more negative tweets in North America

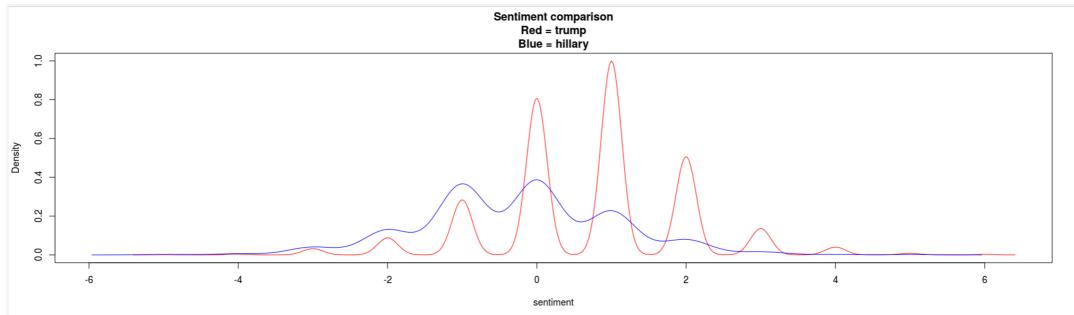


ggmap: A collection of functions to visualize spatial data and models on top of static maps from various online sources (e.g Google Maps and Stamen Maps). It includes tools common to those tasks, including functions for geolocation and routing

```
plotworldmap = function(combinedData1, title,a){  
  worldMap <- map_data("world")  
  map_world <- ggplot(worldMap)  
  map_world <- map_world + borders("world", colour="gray50", fill="gray50")  
  map_world <- map_world + geom_path(aes(x = long, y = lat, group = group), # Draw map  
                                      colour = gray(2/3), lwd = 1/3) +ggtile(paste0("sentiment for ",a))  
  map_world <- map_world + geom_point(data = combinedData1, # Add points indicating users  
                                         aes(x = lon, y = lat, colour = factor(sentichar)))  
  map_world <- map_world + coord_equal()  
  map_world <- map_world + theme_minimal()  
  print(map_world)  
}
```

**Kernel Density plot:** Technique for visualizing the underlying distribution of a continuous variable. Kernel density plot is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data-smoothing problem where inferences about the tweets are made, based on a finite data sample.

**Observation:** Density of Trump's tweet is more as Hillary's

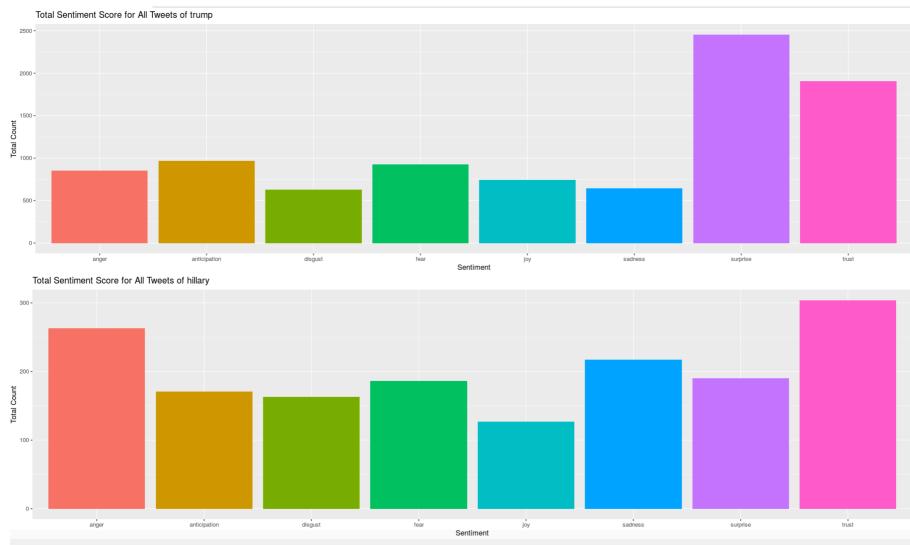


**Kernel Density Estimation:** The (S3) generic function `density` computes kernel density estimates. Its default method does so with the given kernel and bandwidth for univariate observations.

```
output$kerneldensityplot <- renderPlot({
  if(countDataEntire2>0 && countDataEntire1>0){
    d1 <- density(combinedData1$sentiment)
    d2 <- density(combinedData2$sentiment)
    plot(range(d1$x, d2$x), range(d1$y, d2$y), type = "n", xlab = "sentiment",
         ylab = "Density")
    lines(d1, col = "red")
    lines(d2, col = "blue")
    title(main= paste0("Sentiment comparison", "\n", paste0("Red = ",input$keyword1), "\n", paste0("Blue = ",input$keyword2)))
  }
})
```

**Twitter sentiments distribution:** Distribute tweet sentiments: **anger, anticipation, disgust, fear, joy, sadness, surprise, trust**

**Observation:** For trump, there is more surprise tweet sentiments than trust. But, the least ones is disgust. For Hillary, there are more trust sentiments but there are anger sentiments after that. The lease ones is joy



```
plotsentiment = function(combinedData, title, a){
  as <- combinedData
  mySentiment <- get_nrc_sentiment(as$text)
  as <- cbind(as, mySentiment)
  sentimentTotals <- data.frame(colSums(as[,c(10:17)]))
  names(sentimentTotals) <- "count"
  sentimentTotals <- cbind("sentiment" = rownames(sentimentTotals), sentimentTotals)
  rownames(sentimentTotals) <- NULL
  ggplot(data = sentimentTotals, aes(x = sentiment, y = count)) +
    geom_bar(aes(fill = sentiment), stat = "identity") +
    theme(legend.position = "none") +
    xlab("Sentiment") + ylab("Total Count") + ggtitle(paste0("Total Sentiment Score for All Tweets of ",a))
```

## Dashboard 2:

Purpose: It lets a user enter a text and output if the statement is positive, negative and neutral.

### Calculate sentiment

Enter input statement

Submit

### Calculate sentiment

Enter input statement

Submit

It is a positive statement

Model is created in azure and service is deployed. API call is made to service to retrieve the data.

#### Prerequisite:

Services can be called with any programming language and from any device that satisfies three criteria:

- Has a network connection
- Has SSL capabilities to perform HTTPS requests
- Can parse JSON (by hand or support libraries)

Once the experiment has been deployed, there are four pieces of information that we need to call either the RRS or BES service.

- a. The service **API key** or **Primary key**
- b. The service **request URI**
- c. The expected API **request headers** and **body**
- d. The expected API **response headers** and **body**

```

library("RCurl")
library("rjson")

# Accept SSL certificates issued by public Certificate Authorities
calculatePre = function(text){
  options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))
}

h = basicTextGatherer()
hdr = basicHeaderGatherer()

req = list(
  Inputs = list(
    "input1" = list(
      "ColumnNames" = list("text"),
      "Values" = list( list(text), list(text) )
    ),
    GlobalParameters = setNames(fromJSON('[]'), character(0))
  )
)

body = encodeURI(toJSON(req))
api_key = "i0cQU1sI8dq15sWxwdrtVPCWP23JhGP4lfmz7INIUK0Wctycg8fej+acSpJC19y9du/N1YFrnppm3EKYz02gg==" # Replace this with your own API key
authz_hdr = paste('Bearer', api_key, sep=' ')
h$reset()
curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/50d521edc87c4993bebe9c9e147d0c30/services/executeExperiment",
            httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
            postfields=body,
            writefunction = h$update,
            headerfunction = hdr$update,
            verbose = TRUE
)
headers = hdr$value()
httpStatus = headers["status"]
if (httpStatus >= 400)
{
  print(paste("The request failed with status code:", httpStatus, sep=" "))
  # Print the headers - they include the request ID and the timestamp, which are useful for debugging the failure
  print(headers)
}
result = h$value()
print(fromJSON(result))
result1 <- fromJSON(result)
result1 <- result1$results$output1$values[[1]][1]
return (result1)
}

```

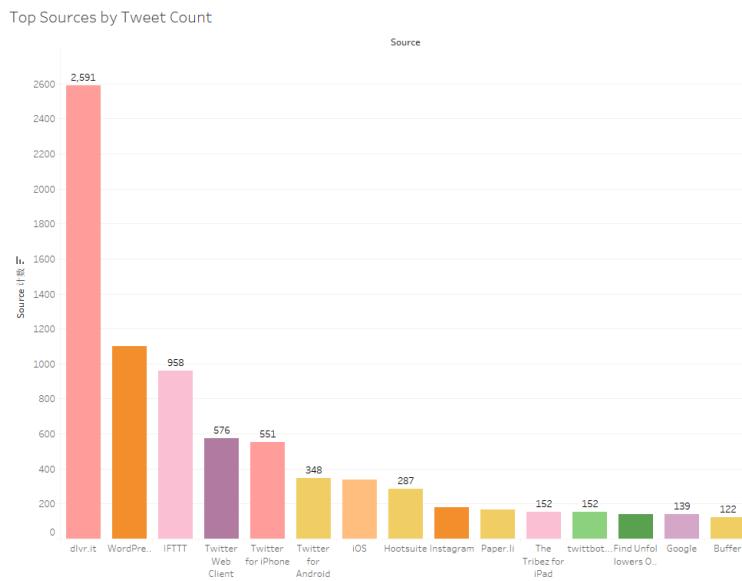
## Dashboard 3:

### Data Visualization

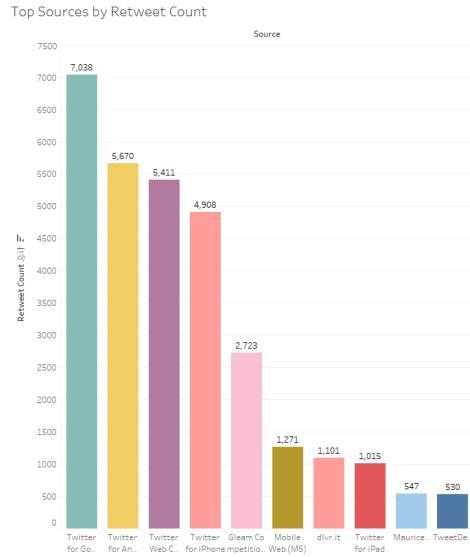
Tools Used: Tableau with Conduct Exploratory Data Analysis

Tableau Public was used to publish the visualizations on Website.

- a) We collected two csv data about 10,000 lines data with key word “iPhone” and “iPad”.
- b) Add sentiment and score label in each line of the original data source.
- c) Finding the Top 10 Sources by Tweet Count. From the bar chart, most Tweet Count were from “dlvr.it”, and then is “WordPress.com”



- a) Finding the Top 10 Sources by Retweet Count. From the bar chart, Twitter for Google TV had 7038 Retweet count, which is the highest, and then, is Twitter for Android.



- b) Top 10 Retweets from Data. The top one retweet is “FOLLOW US for a chance to win an iPhone” which had been retweet for 20,043 times. But this tweet is just an advertisement.

### Top 10 Retweets

Text	
RT @BloggersVibe: #FOLLOW US and #RT US for a chance to #win an #iphon..	20,043
RT @TheMisterFavor: Hurry #Signup for #Socialfave! #android #iphone #Bi..	3,509
RT @TechnoAndroid44: Test your luck in this week's Friday #giveaway and y..	2,723
RT @techbeardblog: How to Free Up Tons of Unused Space on iPhone or iPad ..	1,296
RT @Sofie_hans: #GIVEAWAY FINALLY! WIN #IPHONE7 #IPHONE7PLUS YOU..	1,144
RT @beerrightnow: RT if you were absolutely mesmerizing @iphone How ma..	406
RT @macgizmoguy: CHARRGE-IT! RAVPower's Dual Charger - <a href="https://t.co/0e..">https://t.co/0e..</a>	399
RT @mikeeshift: Get the new #brainwave #meditation #app for #iphone #io..	358
RT @GrantCardone: Success might be the only thing you don't want to do in ..	288
RT @Canoopsy: New video! Secret iPhone Keyboard Trick! <a href="https://t.co/8pMF..">https://t.co/8pMF..</a>	240

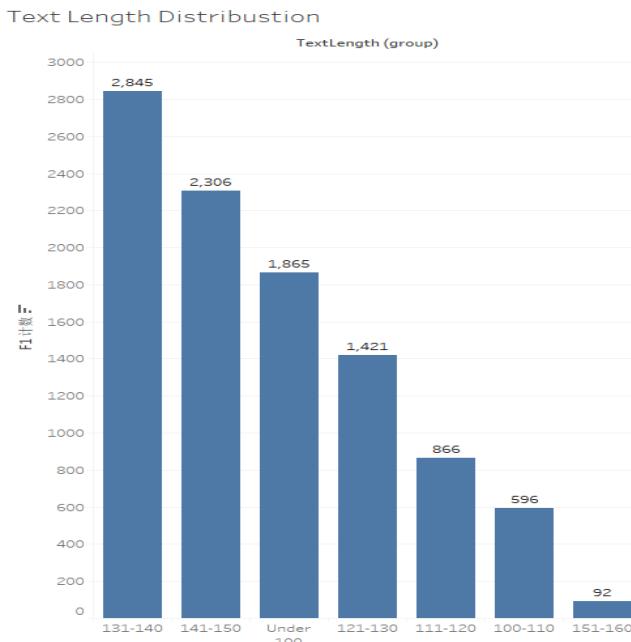
- c) The top 5 favorite count distribution shows most tweet did not have favorite.

### Top 5 FavoriteCount

#### Distribustion

Favorite Count	
0	45,163,541
1	3,671,259
2	504,644
3	226,340
4	113,490

- d) The Text Length Distribution shows, most people write 131-140 words by using iPhone, and then is 141 – 150 words.



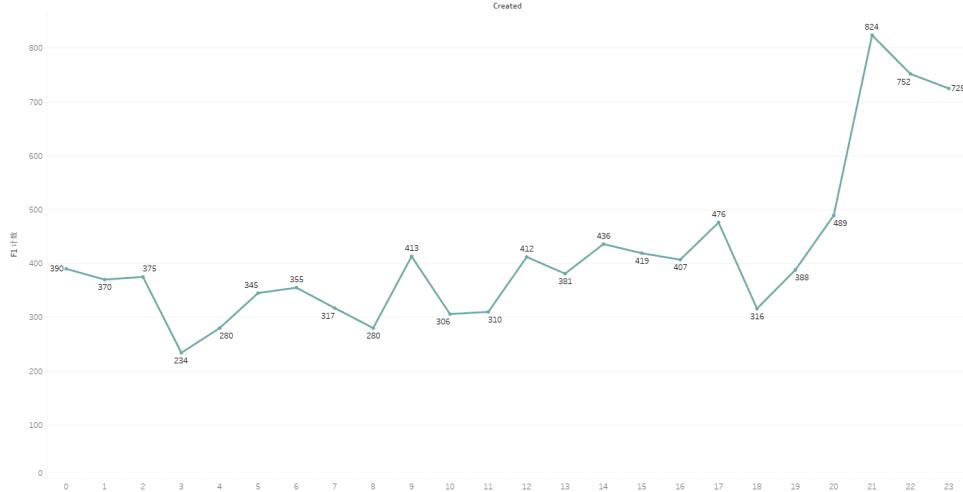
- e) Top 10 Retweet Count Distribution also shows most tweets were not be retweeted, which is similar with Favorite Count Distribution.

Top10 RetweetCount Distribustion

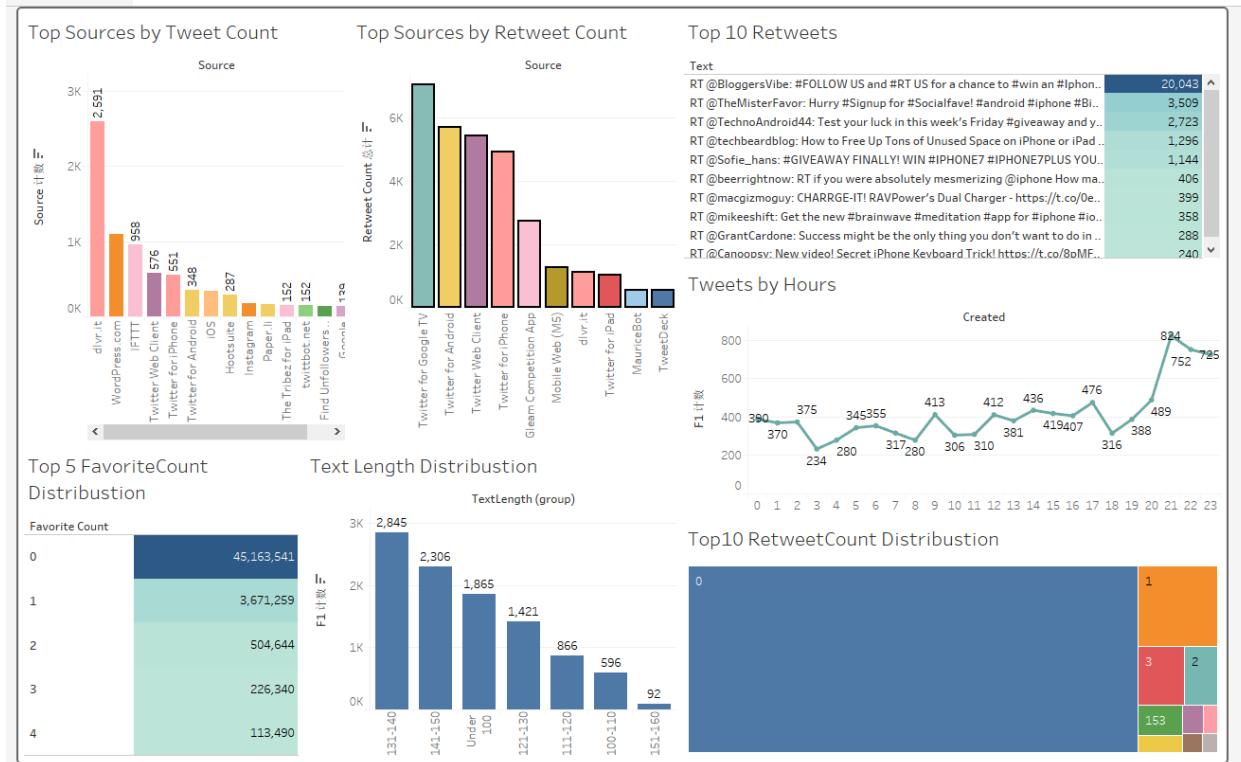


- f) Tweets by Hours shows that 9pm is the peak that people use iPhone to write tweets. The lowest time is 6pm.

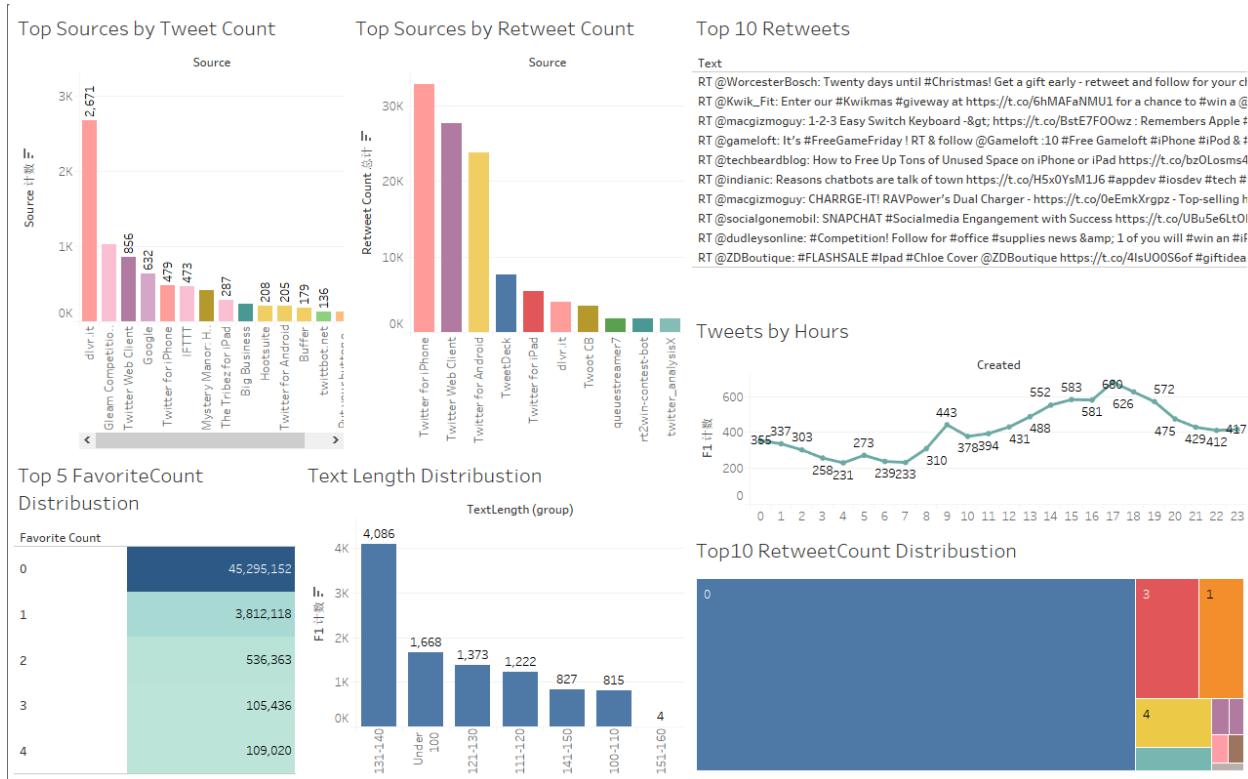
Tweets by Hours



- g) Dashboard of iPhone and iPad:  
iPhone:



## iPad:



## Data Modeling

Tools Used: Microsoft Azure ML Studio

### Building the Model

Steps:

1. Get Data by using R script. Dataset has 3 fields:
  - ✓ Sentiment: the polarity of the tweet (0=negative, 2=neutral, 4=positive)
  - ✓ Target: the query.

	score	tweet_text
view as	list	Does anyone know if you #block someone on #whatsapp or on an #iPhone does the other person see they've been blocked?
0	4	#iphone #style #giftcard #newdeals #win #santa #christmas #blackfriday SKY Devices Elite  <a href="https://t.co/asU4YLk2DO">https://t.co/asU4YLk2DO</a> <a href="https://t.co/QyZyufqaT5">https://t.co/QyZyufqaT5</a> Air Hockey Blue just released on #iphone #mobilegame #indiegames #indiedev #penguins,
0		

- ✓ Tweet\_text: the text of the tweet.

2. Text preprocessing using R script. Tweet usually requires some preprocessing before it can be analyzed. We used the following R code to remove punctuation marks, special character and digits, and then performed case normalization.

▲ Execute R Script

```
R Script
1 # Map 1-based optional input ports to variables
2 dataset <- maml.mapInputPort(1) # class: data.frame
3
4 # Separate the label and tweet text
5 sentiment_label <- dataset[[1]]
6 tweet_text      <- dataset[[2]]
7
8 # Replace punctuation, special characters and digits with space
9 tweet_text <- gsub("[^a-z]", " ", tweet_text, ignore.case = TRUE)
10
11 # Convert to lowercase
12 tweet_text <- sapply(tweet_text, tolower)
13
14 data.set <- as.data.frame(cbind(sentiment_label,tweet_text),
15   stringsAsFactors=FALSE)
16
17 # Select data.frame to be sent to the output Dataset port
18 maml.mapOutputPort("data.set")
```

3. Edit Metadata. We used edit metadata to marked the text column as non-categorical column, and marked the text column as a non-feature.

[Edit Metadata](#)

Column  
**Selected columns:**  
 Column names: tweet\_text

Launch column selector

Data type  
 String

Categorical  
 Make non-categorical

Fields  
 Clear feature

4. Feature Hashing, which can be used to represent variable-length text documents as equal-length numeric feature vectors. We set the number of hashing bits to 17 and the number of N-grams to 2. With these settings, the hash table can hold  $2^{17}$  entries in which each hashing feature will represent one or more unigram or bigram features.

[Feature Hashing](#)

Target column(s)  
**Selected columns:**  
 Column names: tweet\_text

Launch column selector

Hashing bitsize  
 17

N-grams  
 2

5. Split the data into train and test. We use 80% data for training and 20% for testing.

[Split Data](#)

Splitting mode  
 Split Rows

Fraction of rows in the first output dataset  
 0.8

Randomized split

Random seed  
 123

Stratified split  
 True

Stratification key column  
**Selected columns:**  
 Column names: sentiment\_label

Launch column selector

## 6. Train the model based on the algorithm: Two-Class Support Vector Machine Model

### ▲ Two-Class Support Vector Machine

Create trainer mode

Single Parameter

Number of iterations

10

Lambda

0.001

Normalize features

Project to the unit-sphere

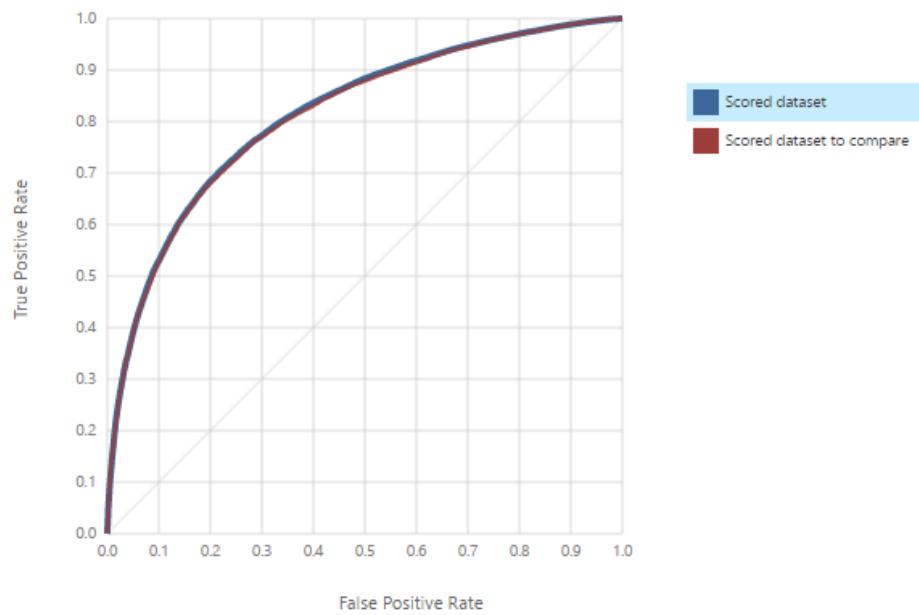
Random number seed

Allow unknown categorical levels

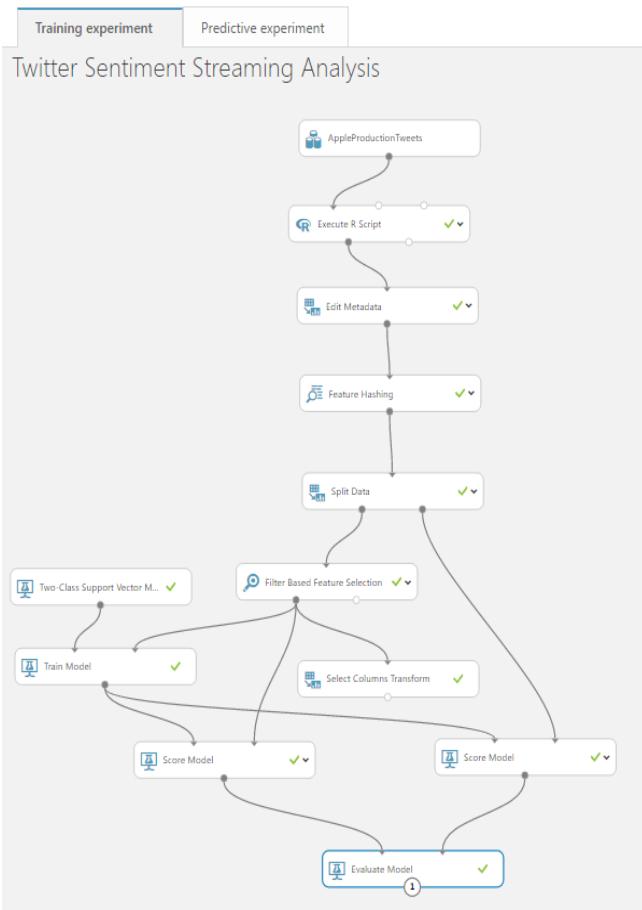
## 7. Evaluate the model

Twitter Sentiment Streaming Analysis > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
49505	14495	0.736	0.720	0.5	0.814
False Positive	True Negative	Recall	F1 Score		
19265	44735	0.774	0.746		
Positive Label	Negative Label				
4	0				



## 8. Add Web Service Input and Output.

## 9. Execute R Script Module:

### Execute R Script

R Script

```

1 #get the scores returned from the model
2 dataset1 <- maml.mapInputPort(1) # class: data.frame
3 #set thresholds for classification
4 threshold1 <- 0.60
5 threshold2 <- 0.45
6 positives <- which(dataset1["Scored Probabilities"] > threshold1)
7 negatives <- which(dataset1["Scored Probabilities"] < threshold2)
8 neutrals <- which(dataset1["Scored Probabilities"] <= threshold1 &
9   dataset1["Scored Probabilities"] >= threshold2)
10 new.labels <- matrix(nrow=length(dataset1["Scored Probabilities"]),
11                       ncol=1)
12 new.labels[positives] <- "positive"
13 new.labels[negatives] <- "negative"
14 new.labels[neutrals] <- "neutral"
15
16 data.set <- data.frame(assigned=new.labels,
17   confidence=dataset1["Scored Probabilities"])
18 colnames(data.set) <- c('Sentiment', 'Score')
19 # Select data.frame to be sent to the output Dataset port
20 maml.mapOutputPort("data.set");

```

## **Deploying the Web Service:**

Steps:

1. The Web service is deployed using REST API in the by node.js in HEROKU server, and in the frontend the data is consumed using JQuery AJAX function. The Web Pages are redirected using Flask Framework which is a node.js framework used for request dispatching in the web applications
2. The technologies used for the FrontEnd are: Bootstrap, HTML5, CSS, Javascript, JQuery, Font-Awesome.

## Dashboard 4:

## Integration of Front end and REST API

Technologies used: Node JS Server, Javascript

```
var Twit = require('twit');
var express = require('express');
var cp = require("child_process");
var http = require('http')
var server = http.createServer(app)
var io = require('socket.io').listen(server);
var app = express();
// 
//var http = require("http");
var https = require("https");
var querystring = require("querystring");
var fs = require('fs');

app.listen(3000);
server.listen(4000);
app.use(express.static(__dirname + '/'));
//app.get('/', function(req,res){res.sendFile('/chart.html')});
function getipaddr(data){
    //print a header
    var host = 'ussouthcentral.services.azureml.net';
    var path = '/workspaces/bd7ad435bc2245faa8796d385b1c5e5/services/e72bbfc04d7d44c69684ec7fe0cc9140/execute?api-version=2.0&details=true';
    var method = 'POST';
    var api_key = 'dvAp//1yFZksMfEW3FhRvbelxbcvQgF7Vj/SXKyf6k8gHpbEK/dv53SY0DMF+hcXmprXla4KKVN6ICAgRb9l/W==';
    var headers = {'Content-Type':'application/json', 'Authorization':'Bearer ' + api_key};

    var options = {
        host: host,
        port: 443,
        path: path,
        method: method,
        headers: headers
    };

    //console.log('data: ' + data);
    console.log('method: ' + method);
    console.log('api_key: ' + api_key);
    console.log('headers: ' + headers);
    console.log('options: ' + options);

    //POST request model using options of header
    var reqPost = https.request(options, function (res) {
        console.log('==>reqPost()==');
        console.log('statusCode: ', res.statusCode);
        console.log('headers: ', res.headers);

        //response
        res.on('data', function(d) {
            io.sockets.emit('stream', d.toString());
            process.stdout.write(d);
        });
    });

    //request practice
    var dataString = JSON.stringify(data);
    console.log(dataString);
    reqPost.write(dataString);
    reqPost.end();
    reqPost.on('error', function(e){
        console.error(e);
    });
});

}

});
```

- Using REST API in the Node js Server
  - In the Frontend calling the API using JQuery.
  - The Front-End code snippet - Blank.html

- Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>

    <!-- start: Meta -->
    <meta charset="utf-8">
    <title>Home</title>
    <meta name="description" content="Bootstrap Metro Dashboard">
    <meta name="author" content="">
    <meta name="keyword" content="">
    <!-- end: Meta -->

    <!-- start: Mobile Specific -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- end: Mobile Specific -->

    <!-- start: CSS -->
    <link id="bootstrap-style" href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/bootstrap-responsive.min.css" rel="stylesheet">
    <link id="base-style" href="css/style.css" rel="stylesheet">
    <link id="base-style-responsive" href="css/style-responsive.css" rel="stylesheet">
    <link href="http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,700italic,800italic,400,300,600,700,800&subset=latin,cyrillic-ext,latin-ext" rel="stylesheet" type="text/css">
    <link href="css/barchart.css" rel="stylesheet">
    <!-- end: CSS -->

    <!-- start: Favicon -->
    <!-- end: Favicon -->
    <script src="js/excanvas.js"></script>
    <script src="js/retina.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
    <script src="js/iphonebar.js"></script>
    <script src="https://cdn.socket.io/socket.io-1.4.5.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.7/angular.min.js"></script>
    <script src=".node_modules/chart.js/dist/chart.js"></script>
    <script src="https://cdn.socket.io/socket.io-1.4.5.js"></script>

</head>

<body>
    <!-- start: Header -->
    <div class="navbar">
        <div class="navbar-inner">
            <div class="container-fluid">
                <a class="btn btn-navbar" data-toggle="collapse" data-target=".top-nav.nav-collapse,.sidebar.nav.nav-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </a>
                <a class="brand" href="introduction.html"><span>Twitter Stream Analysis</span></a>
            <!-- start: Header Menu -->
            <div class="nav-no-collapse header-nav">
                <ul>
                    </ul>
                </div>
            <!-- end: Header Menu -->
        </div>
    </div>
</body>
```

```
<div class="container-fluid-full">
<div class="row-fluid">

    <!-- start: Main Menu -->
    <div id="sidebar-left" class="span2">
        <div class="nav-collapse sidebar-nav">
            <ul class="nav nav-tabs nav-stacked main-menu">
                <li><a href="Introduction.html"><i class="icon-font"></i><span class="hidden-tablet"> Introduction</span></a></li>
                <li><a href="Dashboard.html"><i class="icon-list-alt"></i><span class="hidden-tablet"> Dashboard</span></a></li>
                <li><a href="AppleProduction.html"><i class="icon-bar-chart"></i><span class="hidden-tablet"> AppleProduction</span></a></li>
                <li><a href="gallery.html"><i class="icon-picture"></i><span class="hidden-tablet"> Gallery</span></a></li>
                <li><a href="login.html"><i class="icon-lock"></i><span class="hidden-tablet"> Login Page</span></a></li>
            </ul>
        </div>
    </div>
    <!-- end: Main Menu -->

    <!-- start: Content -->
    <div id="content" class="span10">

        <ul class="breadcrumb">
            <li>
                <i class="icon-home"></i>
                <a href="index.html">Home</a>
                <i class="icon-angle-right"></i>
            </li>
            <li><a href="#">AppleProduction</a></li>
        </ul>
    </div>
</div>
```

## Dynamic Bar Chart:

```
<div id="content" class="span10">

    <ul class="breadcrumb">
        <li>
            <i class="icon-home"></i>
            <a href="index.html">Home</a>
            <i class="icon-angle-right"></i>
        </li>
        <li><a href="#">AppleProduction</a></li>
    </ul>

    <canvas id = "myChart"></canvas>
    <script>
        // var data = [0,0,0];
        var ctx = document.getElementById("myChart");
        function updateChart(data){
            var myChart = new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: ["iPhonePositive", "iPhoneNegative", "iPhoneNeutral","iPadPositive", "iPadNegative", "iPadNeutral"],
                    datasets: [{{
                        label: "# of Votes",
                        data: data,
                        backgroundColor: [
                            'rgba(255, 99, 132, 0.2)',
                            'rgba(54, 162, 235, 0.2)',
                            'rgba(75, 192, 192, 0.2)',
                            'rgba(255, 99, 132, 0.2)',
                            'rgba(54, 162, 235, 0.2)',
                            'rgba(75, 192, 192, 0.2),
                            ],
                        borderColor: [
                            'rgba(255,99,132,1)',
                            'rgba(54, 162, 235, 1)',
                            'rgba(75, 192, 192, 1)',
                            'rgba(255,99,132,1)',
                            'rgba(54, 162, 235, 1)',
                            'rgba(75, 192, 192, 1),
                            ],
                        borderWidth: 1
                    }}]
                },
                options: {
                    scales: {
                        yAxes: [{{
                            ticks: {
                                beginAtZero:true
                            }
                        }}]
                    }
                });
        }
    </script>

```

```

<script>
    var socket = io.connect('http://127.0.0.1:4000');
    var iphonePos = 0;
    var iphoneNeg = 0;
    var iphoneNeutral = 0;
    var ipadPos = 0;
    var ipadNeg = 0;
    var ipadNeutral = 0;
    var none = 0;
    socket.on('ipadStream', function(tweet){
        console.log(tweet)
        // tweet = testData();
        $('#ipadul').append('<li>' + tweet + '</li>');
        if(tweet.includes("negative")){
            ipadNeg++;
            console.log("ipadNeg: " + ipadNeg);
            $('#ipadNeg').html(ipadNeg);
            //data=[ipadPos,ipadNeg,ipadNeutral];
            //$('#ipadul').append('<li>' + "Negative" + '</li>');
        } else if(tweet.includes("positive")) {
            ipadPos++;
            console.log("ipadPos: " + ipadPos);
            $('#ipadPos').html(ipadPos);
            //$('#ipadul').append('<li>' + "Positive" + '</li>');
        } else if(tweet.includes("neutral")) {
            ipadNeutral++;
            console.log("ipadNeutral: " + ipadNeutral);
            $('#ipadNeutral').html(ipadNeutral);
            //$('#ipadul').append('<li>' + "Neutral" + '</li>');
        } else {
            // none++;
            // $('#none').html(none);
        }
        data=[iphonePos,iphoneNeg,iphoneNeutral,ipadPos,ipadNeg,ipadNeutral];
        updatechart(data);
    });
    socket.on('iphonestream', function(tweet){
        $('#iphoneul').append('<li>' + tweet + '</li>');
        if(tweet.includes("negative")){
            iphoneNeg++;
            console.log("iphoneNeg: " + iphoneNeg);
            $('#iphoneNeg').html(iphoneNeg);
            //$('#ipadul').append('<li>' + "Negative" + '</li>');
        } else if(tweet.includes("positive")) {
            iphonePos++;
            console.log("iphonePos: " + iphonePos);
            $('#iphonePos').html(iphonePos);
            //$('#ipadul').append('<li>' + "Positive" + '</li>');
        } else if(tweet.includes("neutral")) {
            iphoneNeutral++;
            console.log("iphoneNeutral: " + iphoneNeutral);
            $('#iphoneNeutral').html(iphoneNeutral);
            //$('#ipadul').append('<li>' + "Neutral" + '</li>');
        } else {
            // none++;
            // $('#none').html(none);
        }
        data=[iphonePos,iphoneNeg,iphoneNeutral,ipadPos,ipadNeg,ipadNeutral];
        updateChart(data);
    });

```

```

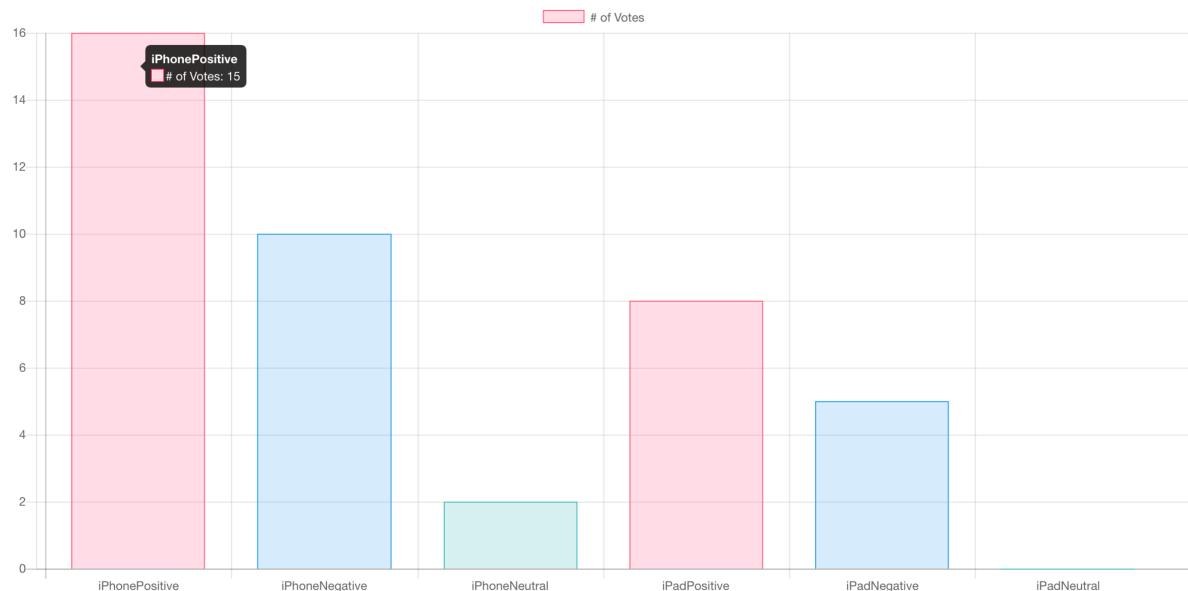
<div class="row-fluid">
    <div>
        <canvas style="margin-bottom: -400px;" id = "myChart" width="400" height="400"></canvas>
        <table width="100%" id="sentimentNum" width="60%" border="1" cellspacing="20%" cellpadding="30">
            <tr>
                <td>iPhone Positive:</td>
                <td>iPhone Negative:</td>
                <td>iPhone Neutral:</td>
                <td>iPad Positive:</td>
                <td>iPad Negative:</td>
                <td>iPad Neutral:</td>
                <!-- <td>None above:</td> -->
            </tr>
            <tr>
                <td id = "iphonePos">0</td>
                <td id = "iphoneNeg">0</td>
                <td id = "iphoneNeutral">0</td>
                <td id = "ipadPos">0</td>
                <td id = "ipadNeg">0</td>
                <td id = "ipadNeutral">0</td>
                <!-- <td id = "none">0</td> -->
            </tr>
        </table>
    </div>

    <br><hr>
    <!-- </div> -->

</div>
<!--iPhone Text-->
<div>
    <div id="iphoneText" class="sparkLineStats span4 widget green" onTablet="span5" onDesktop="span4" style="height:300px; width:480px; margin-left:0; overflow: auto">
        <ul class="dashboard-list metro" id = "iphoneul">
            </ul>
    </div>
</div>

```

When you click the “iPhone & iPad” will connect to the AppleProduction:  
 You will see the dynamic Barchart will show the analysis output from AML Stream Analysis Model to show, how many iPhone and iPad user like or don't like to use thest two product by analysis their tweets.



The table will colcuate how many people like or don't like iphone and ipad.  
The grean filed shows their streaming tweets.

iPhone Positive:	iPhone Negative:	iPhone Neutral:	iPad Positive:	iPad Negative:	iPad Neutral:
32	43	11	20	13	0

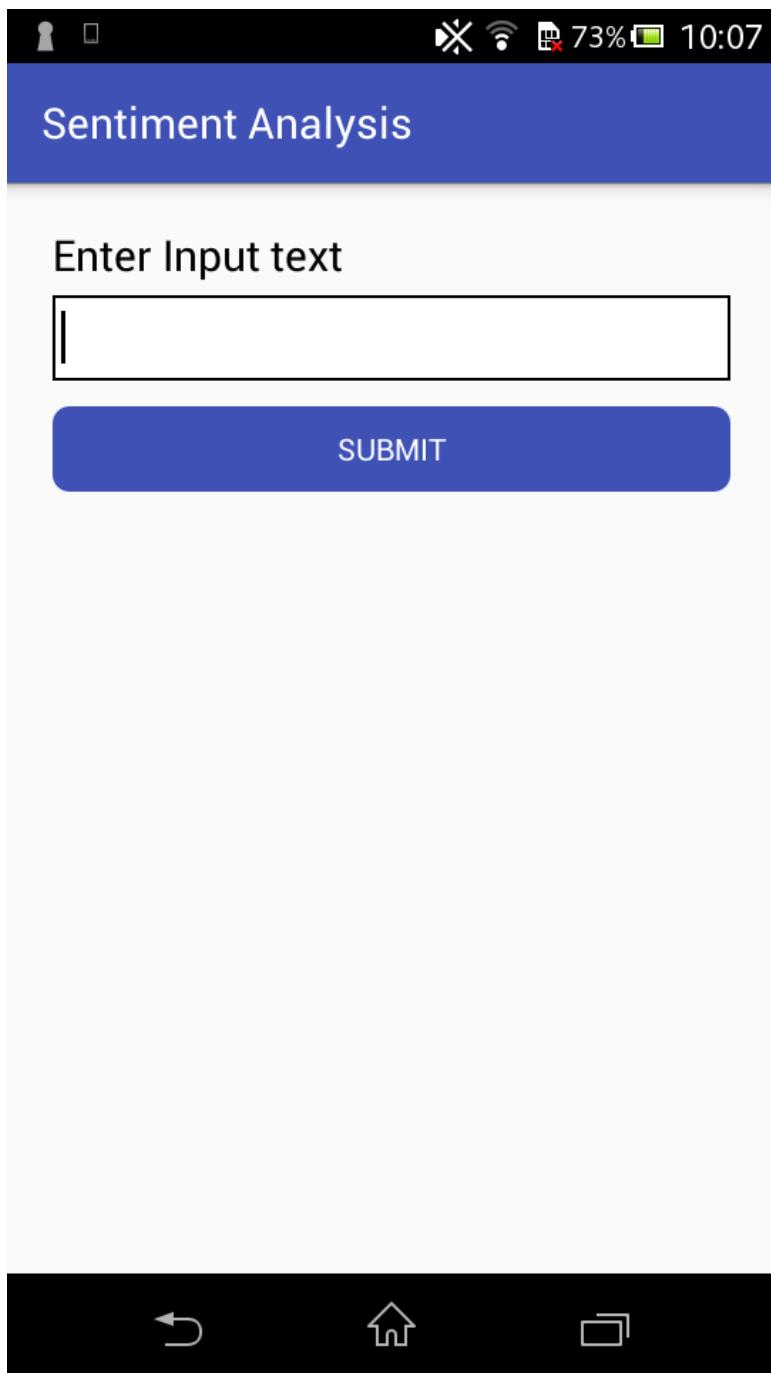
RT brownjenjen: Latest: San Bernardino iPhone battle rages iPhone iPhone <a href="https://t.co/fD5VFVitDR">https://t.co/fD5VFVitDR</a>	<a href="https://t.co/LnCQvqu9A">https://t.co/LnCQvqu9A</a> via KingofAvalonKOA
RT brownjenjen: Latest: San Bernardino iPhone battle rages iPhone iPhone <a href="https://t.co/fD5VFVitDR">https://t.co/fD5VFVitDR</a>	Excalibur reforged Discover how to win a free iPad Kingarthur legendofthesword <a href="https://t.co/3AQwM3Q1WJ">https://t.co/3AQwM3Q1WJ</a> via KingofAvalonKOA
RT brownjenjen: Latest: San Bernardino iPhone battle rages iPhone iPhone <a href="https://t.co/fD5VFVitDR">https://t.co/fD5VFVitDR</a>	I've collected 104700 gold coins <a href="https://t.co/V08bpAh1aA">https://t.co/V08bpAh1aA</a> ipad ipadgames gameinsight

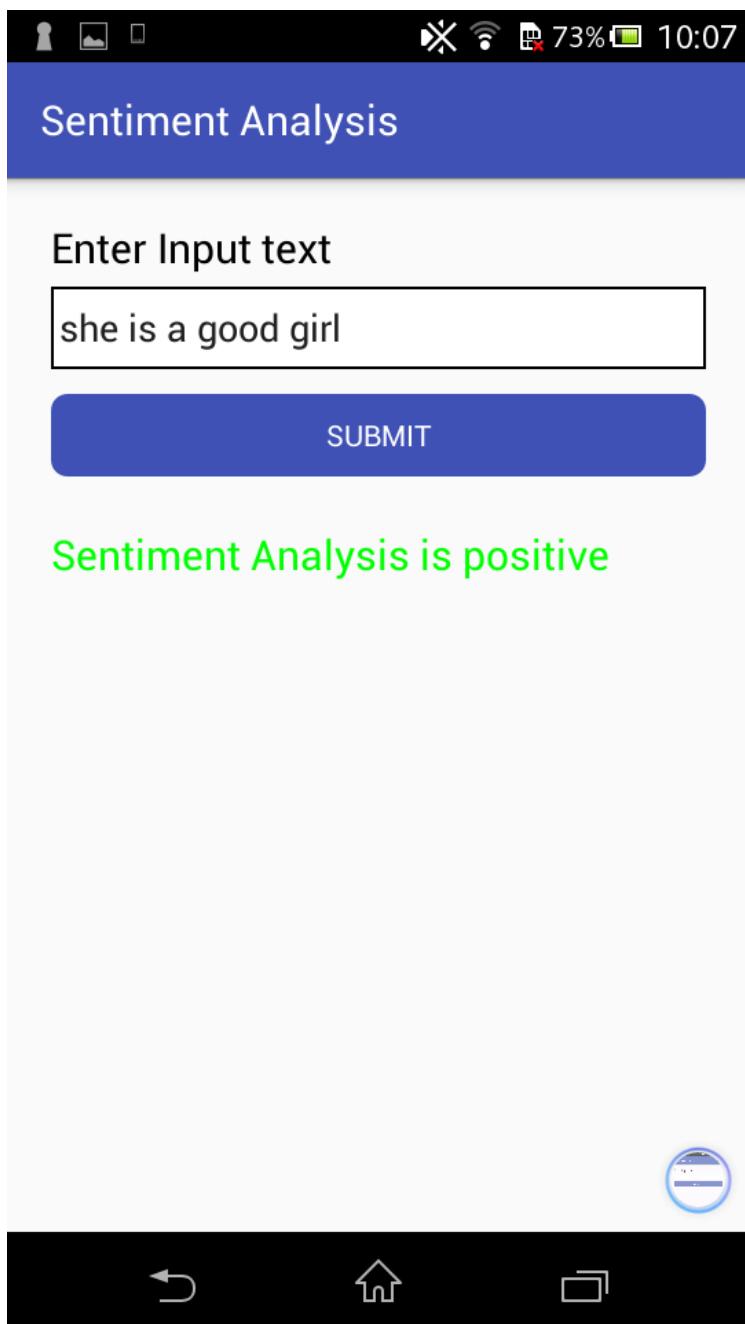
## **Android App:**

We have created an android app that accepts a text and outputs the sentiment of it

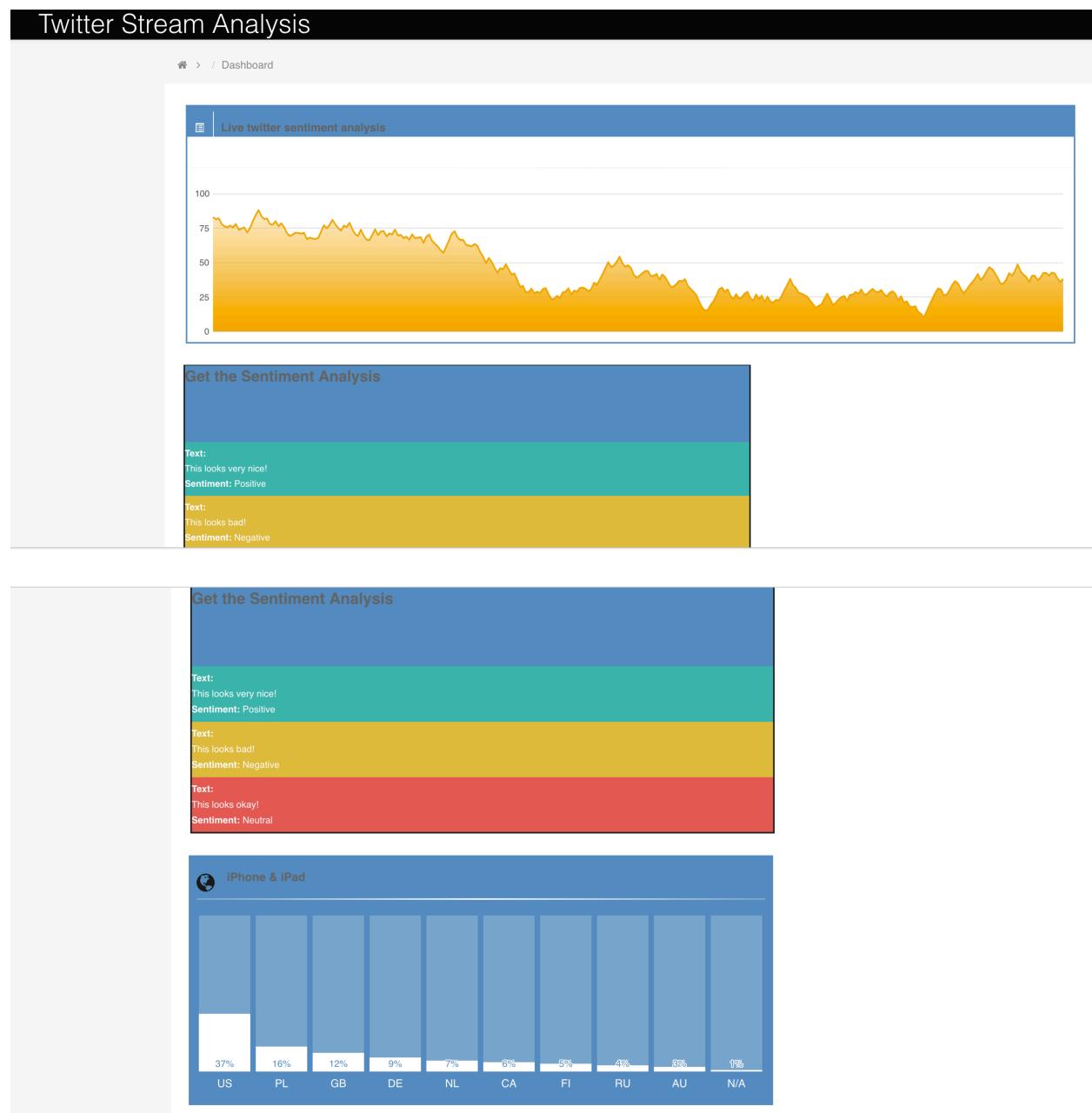
Tool used: Android SDK tool

Language: Java





## Website:



**iPhone & iPad**

Region	Percentage
US	37%
PL	16%
GB	12%
DE	9%
NL	7%
CA	6%
FI	5%
RU	4%
AU	3%
N/A	1%

**Tableau**

```

1 package com.example.priti.sentimentanalysis;
2
3 import android.app.AlertDialog;
4 import android.app.ProgressDialog;
5 import android.content.DialogInterface;
6 import android.graphics.Color;
7 import android.os.AsyncTask;
8 import android.support.v7.app.AppCompatActivity;
9 import android.os.Bundle;
10 import android.view.View;
11 import android.widget.EditText;
12 import android.widget.TextView;
13
14 import org.json.JSONArray;
15 import org.json.JSONException;
16 import org.json.JSONObject;
17
18 import java.io.IOException;
19
20 import okhttp3.MediaType;
21 import okhttp3.OkHttpClient;
22 import okhttp3.Request;
23 import okhttp3.RequestBody;
24 import okhttp3.Response;
25
26 public class MainActivity extends AppCompatActivity {
27
28     String textInput;
29     EditText TextInput;
30     TextView OutputText;
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_main);
35
36         OutputText=(TextView)findViewById(R.id.outputText);
37         TextInput=(EditText)findViewById(R.id.textInput);
38     }
39
40     public void Submit(View view) {
41         textInput = TextInput.getText().toString();
42
43         if (textInput.equals("")) {
44             AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(MainActivity.this);
45             alertDialogBuilder.setTitle("Buzz....");
46             alertDialogBuilder
47                 .setMessage("Please Enter")
48                 .setCancelable(false)
49

```

```

40     public void submit(View view) {
41         TextInput = TextInput.getText().toString();
42
43
44         if (textInput.equals("")) {
45             AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(MainActivity.this);
46             alertDialogBuilder.setTitle("Buzz....");
47             alertDialogBuilder
48                 .setMessage("Please Enter")
49                 .setCancelable(false)
50                 .setNegativeButton("Ok", new DialogInterface.OnClickListener() {
51                     public void onClick(DialogInterface dialog, int id) {
52                         dialog.cancel();
53                     }
54                 });
55             AlertDialog alertDialog = alertDialogBuilder.create();
56             alertDialog.show();
57         } else {
58             String a = "{\n" +
59             "    \"Inputs\": {\n" +
60             "        \"input1\": {\n" +
61             "            \"ColumnNames\": [\n" +
62             "                \"text\"\n" +
63             "            ],\n" +
64             "            \"Values\": [[\""+textInput+"\"]], [\\""+textInput+"\"]]\n" +
65             "        }\n" +
66             "    },\n" +
67             "    \"GlobalParameters\": {}"
68             "}";
69
70             new ServiceFeeCall().execute(a);
71         }
72     }
73
74     public class ServiceFeeCall extends AsyncTask<String, Void, String> {
75         String result;
76         ProgressDialog pd;
77         String outputvalue;
78         @Override
79         protected void onPreExecute() {
80             super.onPreExecute();
81             pd = new ProgressDialog(MainActivity.this);
82             pd.setMessage("Searching for Result. Please wait...");
83             pd.setIndeterminate(false);
84             pd.setCancelable(false);
85             pd.show();
86         }
87         @Override
88         protected String doInBackground(String... params) {
89
90             pd.setMessage("Searching for Result. Please wait...");
91             pd.setIndeterminate(false);
92             pd.setCancelable(false);
93             pd.show();
94         }
95         @Override
96         protected String doInBackground(String... params) {
97             try {
98                 String Url = params[0];
99
100                OkHttpClient client = new OkHttpClient();
101                MediaType mediaType = MediaType.parse("application/json");
102                RequestBody body = RequestBody.create(mediaType, Url);
103                Request request = new Request.Builder()
104                    .url("https://ussouthcentral.services.azureml.net/workspaces/50a521edc87c4993bebe9c9e147d0c30/services/4ef63f97a4e04f3fa9c0256e98d3e529/execute?api-version=2016-05-01")
105                    .post(body)
106                    .addHeader("Authorization", "Bearer i0c0UCIsUBdq15swXwdRtVPCWP23JhGP4lfmz7INIUK0Wctycg8fej+acSpJC19y9du/NlYFrppm3EKYz02gg==")
107                    .addHeader("Content-Type", "application/json")
108                    .addHeader("Cache-Control", "no-cache")
109                    .build();
110
111                Response response = client.newCall(request).execute();
112                result = response.body().string();
113
114                return result;
115            } catch (IOException e) {
116                return null;
117            }
118        }
119        @Override
120        protected void onPostExecute(String s) {
121            super.onPostExecute(s);
122
123            try {
124                JSONObject json = new JSONObject(result);
125                JSONArray json1 = json.getJSONObject("Results").getJSONObject("output1").getJSONObject("value").getJSONArray("Values");
126                for (int i = 0; i < json1.length(); i++) {
127                    JSONArray json2 = json1.getJSONObject(i).getJSONArray("Value");
128                    for (int j = 0; j < json2.length() - 1; j++) {
129                        outputvalue = json2.getString(j);
130                    }
131                }
132            }
133        }

```

```
        .post(body)
        .addHeader("authorization", "Bearer i0cQUCIsUBdq15sWXwdRtVPCWP2JhGP4lfmz7INIUK0Wctycg8fej+acSpJC19y9du/NlYFrnppm3EKYz0")
        .addHeader("content-type", "application/json")
        .addHeader("cache-control", "no-cache")
        .build();

    Response response = client.newCall(request).execute();
    result = response.body().string();

    return result;
} catch (IOException e) {
    return null;
}
}

@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);

    try {
        JSONObject json = new JSONObject(result);
        JSONArray json1 = json.getJSONObject("Results").getJSONObject("output1").getJSONObject("value").getJSONArray("Values");
        for (int i = 0; i < json1.length(); i++) {
            JSONArray json2 = json1.getJSONArray(i);
            for (int j = 0; j < json2.length() - 1; j++) {
                outputvalue = json2.getString(j);
            }
        }
        OutputText.setText("Sentiment Analysis is " + outputvalue);
        if (outputvalue.equals("positive")) {
            OutputText.setTextColor(Color.GREEN);
        } else if (outputvalue.equals("negative")) {
            OutputText.setTextColor(Color.RED);
        } else {
            OutputText.setTextColor(Color.BLUE);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    pd.dismiss();
}
}
```

## References:

- [https://plot.ly/ggplot2/stat\\_smooth/](https://plot.ly/ggplot2/stat_smooth/)
- [https://plot.ly/ggplot2/stat\\_smooth/#adding-trend-lines-in-r-with-statsmooth](https://plot.ly/ggplot2/stat_smooth/#adding-trend-lines-in-r-with-statsmooth)
- <https://susanejohnston.wordpress.com/2012/08/09/a-quick-and-easy-function-to-plot-lm-results-in-r/>
- <http://stackoverflow.com/questions/7005483/geom-smooth-what-are-the-methods-available>
- <http://stackoverflow.com/questions/32908222/conditional-colouring-of-a-geom-smooth?rq=1>
- <https://dev.twitter.com/>
- <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-twitter-sentiment-analysis-trends>
- <http://tweetsentiment.azurewebsites.net/>
- <https://www.r-bloggers.com/search/twitter/>
- <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/>
- <http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>
- <http://www.sananalytics.com/lab/twitter-sentiment/>
- <http://tweetsentiment.azurewebsites.net/>