Airline Sentiment Analysis

❖ Motivation

As we see the need increases for airline companies to revitalize their Customer Relation Management (CRM) strategies, as well as our wishes for the airline industry to be more customer friendly, the team conducted an Airline Sentiment Analysis based on tweets from real passengers. The goal is to build a model that can correctly predict the passenger's sentiment (negative, positive/neutral) out of a tweet. By successfully predicting the negative sentiment, the model helps companies save time filtering the information so that they can focus on the most useful information to find out what needs to be improved.

❖ Data

1. About dataset

The file 'airline' includes daily data, covering from the period from 2/17/15 through 2/24/15, with 14640 observations on 20 features. In our report, we will include three of them:

- Customers' sentiment in each tweet -- categorical variable (airline_sentiment)
- Airline companies – categorical variable (airline)
- Tweet's text – text variable (text)

Besides these three features, we will add one more variable into 'airline' dataset as data modification.

- Tweet's created day, representing in Monday, … etc. – categorical variable (day)

2. Descriptive statistics

First, we will conduct basic data visualization to evaluate the nature of our data. Then we will use text mining method to conduct Natural Language Processing on our text data.

i. Customers' Sentiment Description (See Figure 1)

- Among 14640 observed tweets, 63% of them are identified as negative sentiment, means most of tweets are negative.
- Each airline company has a different percentage of negative sentiment, with American Airline to be the highest, meaning that among all the tweets posted by customers who take American Airline's flights, 71.04% of them are identified to be negative sentiment.
- Within a week, every day except Tuesday has a percentage of Negative sentiment greater than 50%, meaning that except Tuesday, more than half of tweets posted per day by customers would be negative sentiment.

ii. Reason on Negative Sentiment (See Figure 2)

In this step, we want to visualize what would be the distribution of reason of negative sentiment. There are eight kind of negative reasons: Bad Flight; Can't Tell; Cancelled Flight; Customer Service; Damaged Luggage; Flight Attendant Complaint; Flight Booking Problems; Late Flight; Longlines; Lost Luggage; and NA. From this graph, we

would see that the most commonly negative reason would be Customer Service, with the second commonly to be Late Flight.

iii. Tweets Word Count (See Figure 3)

None of the tweets contain greater than 40 words, meaning that none of the customers like to leave long reviews after flights.

3. Data processing

We clean the original data and make preparation for NLP model. The basic cleaning is as following steps:

- Create a new variable called "Negative" and convert the 'sentiment' value to 1 or 0 based on the rule that negative sentiment is equal to 1, 0 otherwise.
- Format adjustment: Change all the text into the lower case and remove all extra space.
- Remove stopwords
- Get rid of @virginamerica,@usairways,@united,@southwestair,@jetblue, @americanair
- Remove remaining punctuation
- Remove all http links
- Stem our document which means chopping off the ends of words that aren't maybe as necessary as the rest

  In this step, we've finished our basic cleaning. We should mention that we don't remove numbers in case it states something like X hours late. In the following step, we try to calculate frequencies of words across each observation.
- Create a word count matrix (rows are each observation, columns are words).
- Account for sparsity and remove infrequent terms. I keep terms that appear in 1% or more of the observations
- Create data frame from the document-term matrix

  In the end, we leave 155 words and make the world cloud. (See Figure 4 the word cloud of tweets)

  The difference between negative tweets and non-negative tweets

  We divide the original dataset into the negative and non-negative part and separately process the subsets using the above method. And we analyze the relationship of the two subsets. (See Figure 5 and 6)

  We found that "thank" is the most common word in the non-negative tweets. However, the words "get", "hour", "help", "time" and "bag" appear commonly in the negative tweets. These words in the negative tweets could indirectly reflect the aspects that airlines need to improve.

❖ Model

We approach building models from two different ways: one is taking NLP data to build supervised learning models, such as Logistic Regression, CART, Random Forest and Boosting to predict customer's sentiment. The other way is by considering customer reviews' week date

and Airline to make prediction through logistic regression. Finally, we apply blending methodology to combine multiple high-performance models to find the best result.

Before conducting models, we split observations into three datasets: training set with 80% of the observations, validation set with 15% of the observations, and testing set with 5% of the observations.

1. NLP Models (Approach 1)

(1) Baseline Model

In the Baseline model, we find the accuracy of predicting negative in the Test set to be 0.6434426. It will be used to compare with other advanced models to assess the quality of those models.

(2) Cross-validated CART Model for sentiment prediction

We use 10 folds-cross validation to select cp parameter. We set cp=seq(0, 0.4, 0.002). According to the decision tree, we find that "cancel", "delay", "will", "can", and "help" are important words and "problem", "gate" and "never" are sentimental words.

(3) Cross validated Random Forest Model and analysis of Important word

We use 5 folds-cross validation to select mtry. Set mtry=1:120. In result, best mtry = 5. According to importance, we find that the words "hour", "great", "thank", "call", "hold", "bag", "custom", "delay", "cancel", and "wait" have high MeanDecreseGini, which proves their importance in the model and prediction.

(4) Logistic Regression for prediction of customer sentiment

According to the output, 105 out of 157 variables are significant at the 95% level.

(5) Boosting

We use 5 folds-cross validation to select the parameter n.trees (total number of boosting iterations) and interaction depth (maximum number of variable interactions in each tree). The final values used for the model were $\mathrm{n.trees} = 3950$, $\mathrm{interaction.depth} = 6$, $\mathrm{shrinkage} = 0.01$ and $\mathrm{n.minobsinnode} = 10$.

Comparison of Out of Sample Predictive Quality

Model Type	Accuracy
Baseline	0.6434426
Logistic Regression	0.784153
CART	0.7472678
Random Forest	0.8046448
Boosting	0.8087432

2. Logistic Regression Model (Approach 2)

Considering airline and tweet week date as two features to predict negative sentiment. Set threshold=0.5. By running logistic regression model, we obtained $\mathrm{Accuracy} = (123 + 391)/(123 + 391 + 80 + 138) = 0.7021858$.

In conclusion, from the airline's perspective: Delta, Southwest, and Virgin America have significant p-values with negative coefficient, meaning that passengers are more satisfied with

these three airlines. US airways has significant p-value with positive coefficient, meaning that it is less satisfied by passengers. From the day's perspective, tweets on Saturday and Sunday have significant p-values with positive coefficient, meaning that passengers tend to give negative reviews on weekends.

3. Blending Model: Approach 1 + Approach 2

Based on first approach, Random Forest and Boosting have relatively high performance in accuracy. We will blend the two models with second approach, logistic regression. Applying logistic regression model to build blending model. Set threshold=0.5. Validating the blending model in test dataset, we receive accuracy=0.8101093.

We also try individual blending model: RF blend with Logistic or Boosting blend with Logistic model. It turns out that RF with Logistic gives higher accuracy **0.8114754.**

⇨ **Final Model: Blending model – RF+Logistic Regression**

4. Model Validation

Use the bootstrap to assess the performance of final model – Blending model, in a way that properly reports on the variability of the relevant performance metrics (accuracy, TPR, and FPR). After running 10000 bootstrap replicates, we receive a better idea about Blending Model's performance based on accuracy, TPR and FPR. Each metric's variability (max-min) is between 0.06~0.11 by tracking each metric's confidence interval. This means that the model performs reasonably and relatively stable in the test set.

Bootstrap Statistics				
	Original	Bias	Std. error	Confidence Interval
Accuracy	0.8114754	-8.169399e-05	0.01465279	(0.7828, 0.8415)
TPR	0.8832272	-9.621899e-05	0.01491487	(0.8551, 0.9136)
FPR	0.3180077	2.530078e-05	0.02887533	(0.2610, 0.3731)

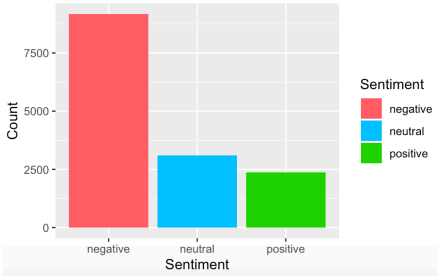
❖ Impact

The team was able to build a significant model which has both accuracy and TPR over 80%. Airline companies can apply this model to tweets and evaluate their customer feedback. Specifically, our work will enable the airline companies to save time and target the negative tweets more efficiently. Moreover, the NLP models show that there are words that do not appear as much but have strong sentiment indication and therefore, the airline companies CRM should pay extra attention to those words. From passengers' view, they can use the logistic models to compare between airlines and find the ones that have fewer negative comments. In the future, it might help to add another feature to the model related to "frequent flyer program" so that it can tell passengers whether it is worthwhile to reach higher status for the specific airline based other passengers' reviews. If an elite member of a program posts many negative tweets, it would indicate that the airline's program was not so worthy. On the other hand, once companies start using the model, it will be trained better with more data and ideally the model can be used by other industries to improve their customer service as well.

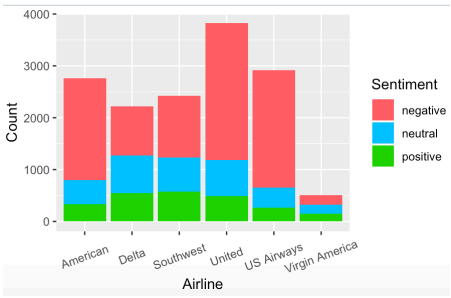
Appendix

- Data Description:

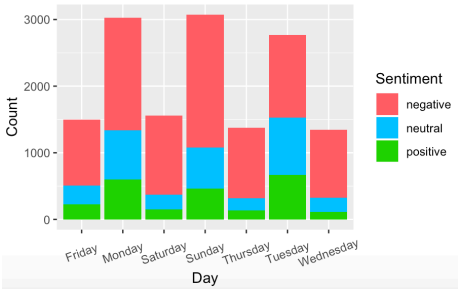
i. Customers' Sentiment Description



Histogram 1: Global Customers' Sentiment



Histogram 2: Sentiment towards different Airline Companies



Histogram 3: Sentiment on Specific Weekday

Figure 1 Customers' Sentiment Description

ii. Reasons on Negative Sentiment:

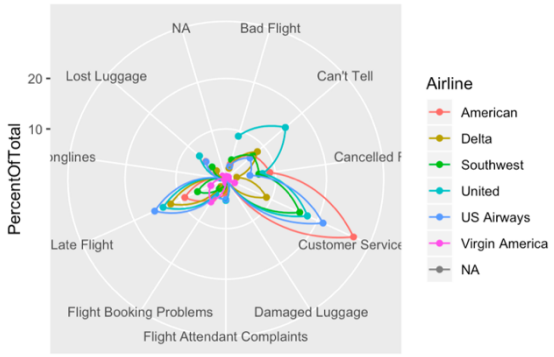


Figure 2 Reasons on Negative Sentiment

iii. Tweets Word Count

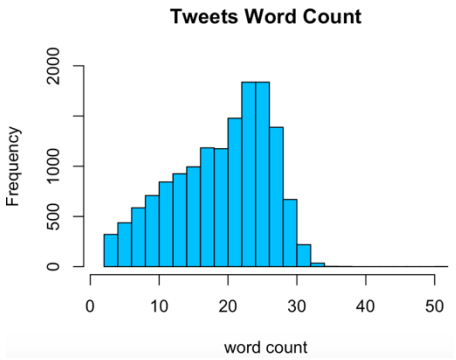


Figure 3 Tweets word count

- Data Processing:

i. The word cloud of tweets

Finally, we leave 155 words. The size of each word represents the frequencies in the tweets.

Figure 4 the word cloud of tweets

ii. negative tweets

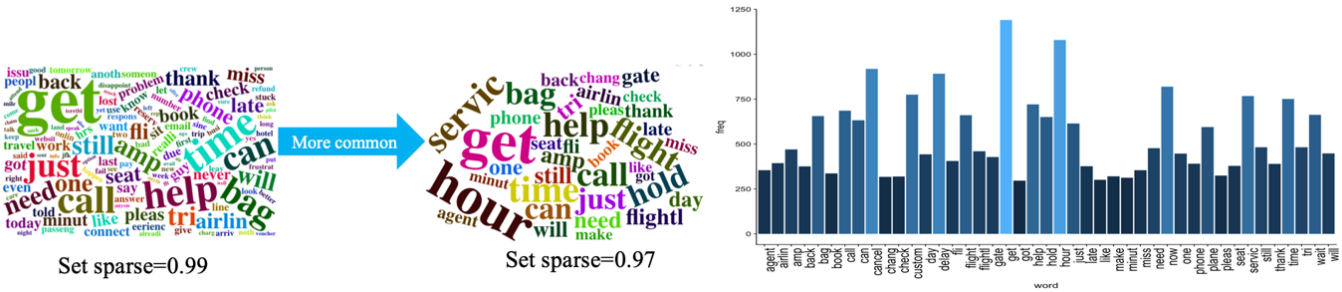


Figure 5 the words of negative tweets

iii. non-negative tweets

Figure 6 the words of non-negative tweets

Code

```
---
  title: "242 Final Project"
  author: "Joanne Jiang, Peter Pan, Huidi Wang, Yuyang Zhao, Yiyaqi Zuo"
  date: "12/9/2019"
  output: pdf_document
---

## INTRODUCTION
```{r  LOAD LIBRARIES AND DATA}
library(tm)
library(SnowballC)
library(wordcloud)
library(MASS)
library(caTools)
library(dplyr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)

library(tm.plugin.webmining)
library(boot)
library(ggplot2)
library(readr)
library(RColorBrewer)
library(biclust)
library(cluster)
library(igraph)
library(fpc)
library(gridExtra)
library(cowplot)
library(reshape2)
library(scales)
library(ngram)

airline=read.csv('Airline-Sentiment-2-w-AA.csv',stringsAsFactors=FALSE)
```

1. UNDERSTANDING DATASET
```{r  1. UNDERSTANDING DATASET}
# 1. Global customers' sentiment histogram
overallSentiment = as.data.frame(table(airline$airline_sentiment))
colnames(overallSentiment) = c("Sentiment","Count")
histPlot1 = ggplot(overallSentiment) + aes(x=Sentiment, y=Count, fill=Sentiment) + scale_fill_manual(values=c("indianred1","deepskyblue","chartreuse3"))
histPlot1 = histPlot1 + geom_bar(stat="identity")
histPlot1
table(airline$airline_sentiment)

# 2. Customers' sentiment towards different Airlines histogram
airlineSentiment = as.data.frame(table(airline$airline,airline$airline_sentiment))
colnames(airlineSentiment) = c("Airline","Sentiment","Count")
colours = c("firebrick1","deepskyblue","chartreuse3")

histPlot2 = ggplot(airlineSentiment) + aes(x=Airline,y=Count,fill=Sentiment) + scale_fill_manual(values=c("indianred1","deepskyblue","chartreuse3"))
histPlot2 = histPlot2 + geom_bar(stat="identity") +theme(axis.text.x = element_text(angle = 20, hjust = 0.5, vjust = 0.5))
histPlot2
table(airline$airline,airline$airline_sentiment)

# 3. Customers' sentiment on specific weekdays histogram
daySentiment = as.data.frame(table(airline$day,airline$airline_sentiment))
colnames(daySentiment) = c("Day","Sentiment","Count")
colours = c("firebrick1","deepskyblue","chartreuse3")

histPlot3 = ggplot(daySentiment) + aes(x=Day,y=Count,fill=Sentiment) + scale_fill_manual(values=c("indianred1","deepskyblue","
```
```

```
chartreuse3"))
histPlot3 = histPlot3 + geom_bar(stat="identity") +theme(axis.text.x = element_text(angle = 20, hjust = 0.5, vjust = 0.5))
histPlot3
table(airline$day,airline$airline_sentiment)

# 4. Negative reason
# Initial Global analysis
#str(airline)
table(airline$negativereason, airline$airline)

globalSentReasons = as.data.frame(table(airline$negativereason, airline$airline))
colnames(globalSentReasons) = c("Reason","Airline", "Freq")
#globalSentReasons

ggplot(globalSentReasons) + aes(y = Freq, x = Reason, group = Airline, colour = Airline) + coord_polar() + geom_point() + geom_path()

## TYRING TO CALCULATE PERCENTAGES
aggregate(Freq ~ Airline, globalSentReasons, sum)
globalSentReasons$TotalTwAirline = 0
globalSentReasons[1:11,4] = 2759
globalSentReasons[12:22,4] = 2222
globalSentReasons[23:33,4] = 2420
globalSentReasons[34:44,4] = 3822
globalSentReasons[45:55,4] = 2913
globalSentReasons[56:66,4] = 503
globalSentReasons$PercentOfTotal = (globalSentReasons[,3]/globalSentReasons[,4])*100

ggplot(globalSentReasons) + aes(y = PercentOfTotal, x = Reason, group = Airline, colour = Airline) + coord_polar() + geom_point() + geom_path() + labs(x = NULL)
#ggplot(globalSentReasons) + aes(y = PercentOfTotal, x = Reason, group = Airline, colour = Airline) + geom_hist() + geom_path() + labs(x = NULL)

# 5. Word count
airline$word_count <- sapply(airline$text, function(x) length(unlist(strsplit(as.character(x),"\\W+"))))
word_count <- airline$word_count
hist(word_count,
     main='Tweets Word Count',
     xlab='word count',
     xlim=c(1,50),
     ylim=c(0,2000),
     breaks=20,
     col = 'deepskyblue',
     freq=TRUE)
```

2.Data PROCESSING

DATA PRE-PROCESSING PART II: DATA CLEANING + NLP

```{r }
#Add tweet weekdate as a new feature
airline$tweet_date=weekdays(as.Date(airline$tweet_created))
#factorize negative sentiment as 1, other sentiment as 0
airline$Negative = as.factor(as.numeric(airline$airline_sentiment=="negative"))
```

```{r NLP Processing}
corpus = Corpus(VectorSource(airlinet$text))
strwrap(corpus[[8]])
corpus <- tm_map(corpus, stripWhitespace)
strwrap(corpus[[8]])
corpus <- tm_map(corpus, content_transformer(tolower))
strwrap(corpus[[8]])
# Remove stopwords
corpus = tm_map(corpus, removeWords, stopwords("english"))
strwrap(corpus[[8]])
# Want to get rid of @virginamerica, @usairways, @united, @@southwestair, @jetblue, @americanair
corpus <- tm_map(corpus, function(x) gsub('@', 'KeepAtPunct', x))
strwrap(corpus[[8]])
corpus = tm_map(corpus, removeWords, c("KeepAtPunctvirginamerica","KeepAtPunctusairways","KeepAtPunctunited","KeepAtPunctsouthwestair","KeepAtPunctjetblue","KeepAtPunctamericanair"))
strwrap(corpus[[8]])
corpus <- tm_map(corpus, function(x)
```

```
gsub('KeepAtPunct', '@', x))
# Remove remaining punctuation
corpus <- tm_map(corpus, removePunctuation)
strwrap(corpus[[8]])
# Remove "x."
corpus <- tm_map(corpus, function(x) gsub('x.', '', x))
# Remove all http links
corpus <- tm_map(corpus, function(x) gsub('http[[:alnum:]]*', '', x))
strwrap(corpus[[8]])
Clean <- function(HTML){ return(gsub("\ / < . *?> <p> \t \n /f /n", " ", HTML))}
corpus = tm_map(corpus, Clean)
# Remove "flight"
corpus <- tm_map(corpus, removeWords,c('flight'))
# Stem the document
corpus = tm_map(corpus, stemDocument)
# Create matrix
corpus= DocumentTermMatrix(corpus)
sparse = removeSparseTerms(corpus, 0.99)
dtm = as.data.frame(as.matrix(sparse))
#DESCRIPTIVE ANALYTICS

freq = colSums(as.matrix(dtm))
ord = order(freq)

wf = data.frame(word=names(freq), freq=freq)
wf = wf[order(wf$freq,decreasing = T),]
a=wf
p = ggplot(subset(a, freq>=100), aes(word, freq, fill = freq))
p = p + geom_bar(stat="identity", show.legend = TRUE)
p = p + theme(axis.text.x=element_text(angle=90, hjust=1, size = rel(1.5)))
##CONVERTING TO DATA FRAME
tweets = as.data.frame(as.matrix(dtm))
colnames(tweets) = make.names(colnames(tweets))
tweets$airline_sentiment = airline$airline_sentiment
```

```{r wordcloud}
install.packages("wordcloud")
library(wordcloud)
install.packages("RColorBrewer")
library(RColorBrewer)
install.packages("wordcloud2")
library(wordcloud2)
wordcloud2(data=wf, size=0.7, color='random-dark',shape = 'pentagon')
```

3. BUILDING MODELS-Approach1: NLP Models

```{r 3. BUILDING MODELS}
tableAccuracy <- function(test, pred) {
  t = table(test, pred)
  a = sum(diag(t))/length(test)
  return(a)
}
# Load the data set
set.seed(123)
tweets$Negative = as.factor(as.numeric(tweets$airline_sentiment=="negative"))
tweets$airline_sentiment <- NULL
#spl = sample.split(tweets0$Negative, SplitRatio = 0.7)
train.ids <- sample(nrow(tweets), 0.95*nrow(tweets))
TweetsTrain <- tweets[train.ids,]
TweetsTest <- tweets[-train.ids,]

# split training into real training and validation set
val1.ids <- sample(nrow(TweetsTrain), (10/95)*nrow(TweetsTrain))
val1 <- TweetsTrain[val1.ids,]
TweetsTrain <- TweetsTrain[-val1.ids,]

#TweetsTrain = tweets0 %>% filter(spl == TRUE)
#TweetsTest = tweets0 %>% filter(spl == FALSE)

#Base line Model
#use to find the accuracy of baseline model
table(TweetsTest$Negative)
(471)/(471+261)

# Cross-validated CART model
set.seed(3421)
train.cart = train(Negative ~ .,
                   data = TweetsTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp=seq(0, 0.2, 0.002)),
                   trControl = trainControl(method="cv", number=10))
```

```
train.cart
train.cart$results

ggplot(train.cart$results, aes(x = cp, y = Accuracy)) +
  geom_point(size = 2) +
  geom_line() +
  ylab("CV Accuracy") +
  theme_bw() +
  theme(axis.title=element_text(size=18),
axis.text=element_text(size=18))

mod.cart = train.cart$finalModel
prp(mod.cart)

predict.cart = predict(mod.cart, newdata = TweetsTest,
type = "class") # why no model.matrix?
table(TweetsTest$Negative, predict.cart)
tableAccuracy(TweetsTest$Negative, predict.cart)

# Cross validated RF
# WARNING: this took me approx. 24 hour to run
set.seed(311)
train.rf = train(Negative ~ .,
                 data = TweetsTrain,
                 method = "rf",
                 tuneGrid = data.frame(mtry = 1:120),
                 trControl = trainControl(method =
"cv", number = 5, verboseIter = TRUE))
train.rf
train.rf$results

ggplot(train.rf$results, aes(x = mtry, y = Accuracy)) +
geom_point(size = 2) + geom_line() +
  ylab("CV Accuracy") + theme_bw() +
  theme(axis.title=element_text(size=18),
axis.text=element_text(size=18))

mod.rf = train.rf$finalModel
importance(mod.rf)
predict.rf = predict(mod.rf, newdata = TweetsTest)
table(TweetsTest$Negative, predict.rf)
tableAccuracy(TweetsTest$Negative, predict.rf)

# Logistic Regression
TweetLog = glm(Negative ~ ., data = TweetsTrain, family
= "binomial")
PredictLog = predict(TweetLog, newdata = TweetsTest,
type = "response")
# You may see a warning message - suspicious, but we
will just ignore this

summary(TweetLog)
table(TweetsTest$Negative, PredictLog > 0.5)
tableAccuracy(TweetsTest$Negative, PredictLog > 0.5)

# Boosting
tGrid = expand.grid(n.trees = (1:100)*50,
interaction.depth = c(1,2,4,6,8,10,12,14,16),
                    shrinkage = 0.01,
n.minobsinnode = 10)
set.seed(232)
# WARNING: this took me approx. 8 hour to run
train.boost <- train(Negative ~ .,
                     data = TweetsTrain,
                     method = "gbm",
                     tuneGrid = tGrid,
                     trControl =
trainControl(method="cv", number=5, verboseIter =
TRUE),
                     metric = "Accuracy",
                     distribution = "bernoulli")
train.boost
train.boost$results

ggplot(train.boost$results, aes(x = n.trees, y = Accuracy,
colour = as.factor(interaction.depth))) + geom_line() +
  ylab("CV Accuracy") + theme_bw() +
theme(axis.title=element_text(size=18),
axis.text=element_text(size=18)) +
  scale_color_discrete(name = "interaction.depth")

mod.boost = train.boost$finalModel
TweetsTest.mm = as.data.frame(model.matrix(Negative
~ . +0, data = TweetsTest))
predict.boost = predict(mod.boost, newdata =
TweetsTest.mm, n.trees = 4100, type = "response")
table(TweetsTest$Negative, predict.boost < 0.5) # for
some reason the probabilities are flipped in gbm
tableAccuracy(TweetsTest$Negative, predict.boost < 0.5)

```

3. BUILDING MODELS- Approach 2: Logistic

Regression

```{r}
library(gbm)

### Model2- Logistic Regression

model2_df<-airline[,c("Negative", "airline","tweet_date")]
df_Train <- model2_df[train.ids,]
df_Test <- model2_df[-train.ids,]
df_val1 <- df_Train[val1.ids,]
df_Train <- df_Train[-val1.ids,]


mod2Log = glm(Negative ~ ., data = df_Train, family = "binomial")
Predict_mod2Log = predict(mod2Log, newdata = df_Test, type = "response")
summary(Predict_mod2Log)
table(df_Test$Negative, Predict_mod2Log > 0.5)
tableAccuracy(df_Test$Negative, Predict_mod2Log > 0.5)
summary(mod2Log)
```

3. BUILDING MODELS- Blending Model: Approach1+Approach2

```{r}
###Blending: RF+BOOSTING+LOGISTIC

#Blending Prediction
set.seed(1000)
val.predict.rf=predict(mod.rf, newdata = val1)
val.predict.tweetlog=predict(mod.log, newdata = val1, type = "response")
val.predict.cart=predict(mod.cart, newdata = val1, type = "class")
TweetsVal.mm = as.data.frame(model.matrix(Negative ~ . +0, data = val1))
val.predict.boost=predict(mod.boost, newdata = TweetsVal.mm, n.trees = 3950, type = "response")
val.predict.log=predict(mod2Log, newdata = df_val1, type = "response")

# Build validation set data frame
val.blending_df = data.frame(Negative = (val1$Negative),
                             log_preds = val.predict.log,
                             rf_preds = (val.predict.rf)

#tweetlog_preds=val.predict.tweetlog,

#cart_preds=(val.predict.cart),

#boost_preds=val.predict.boost
                             )

# Train blended model
blend.mod =glm(Negative ~ . -1, data = val.blending_df, family = "binomial")
summary(blend.mod)
# Get predictions on test set
set.seed(1000)
test.predict.rf=predict(mod.rf, newdata = TweetsTest)
test.predict.tweetlog=predict(mod.log, newdata = TweetsTest, type = "response")
test.predict.cart=predict(mod.cart, newdata = TweetsTest, type = "class")
test.predict.boost=predict(mod.boost, newdata = TweetsTest.mm, n.trees = 3950, type = "response")
test.predict.log=predict(mod2Log, newdata = df_Test, type = "response")


test.blending_df = data.frame(Negative = (df_Test$Negative),
                              log_preds = test.predict.log,
                              rf_preds = (test.predict.rf)

#tweetlog_preds=test.predict.tweetlog,

#cart_preds=(test.predict.cart),

#boost_preds=test.predict.boost
)

test.preds.blend <- predict(blend.mod, newdata = test.blending_df,type = "response")
```

```
table(df_Test$Negative, test.preds.blend > 0.5)
tableAccuracy(df_Test$Negative, test.preds.blend > 0.5)
```

4. MODEL VALIDATION

```{r  4. MODEL VALIDATION}

### Data Validation: Bootstrap
#Best Final Model: Blending

tableAccuracy <- function(label, pred) {
  t = table(label, pred)
  a = sum(diag(t))/length(label)
  return(a)
}

tableTPR <- function(label, pred) {
  t = table(label, pred)
  return(t[2,2]/(t[2,1] + t[2,2]))
}

tableFPR <- function(label, pred) {
  t = table(label, pred)
  return(t[1,2]/(t[1,1] + t[1,2]))
}

boot_accuracy <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  return(tableAccuracy(labels, predictions))
}

boot_tpr <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  return(tableTPR(labels, predictions))
}

boot_fpr <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  return(tableFPR(labels, predictions))
}

boot_all_metrics <- function(data, index) {
  acc = boot_accuracy(data, index)
  tpr = boot_tpr(data, index)
  fpr = boot_fpr(data, index)
  return(c(acc, tpr, fpr))
}
# Finalmodel
library(boot)
big_B=10000
blending_df = data.frame(labels = df_Test$Negative, predictions = test.preds.blend > 0.5)
set.seed(432)
Boost_boot = boot(blending_df, boot_all_metrics, R = big_B)
Boost_boot
boot.ci(Boost_boot, index = 1, type = "basic")  #CI for Accuracy of Boosting
boot.ci(Boost_boot, index = 2, type = "basic")  #CI for tpr of Boosting
boot.ci(Boost_boot, index = 3, type = "basic")  #CI for fpr of Boosting
```