USE CASE STUDY REPORT

Background and Introduction

Background

Feedback is very important for any Company's improvement and growth process. Only by analysing feedback from customers, can a company identify areas for potential improvement, and thereby take measures to achieve the same.

Mainstream Social Media plays a significant role in maintaining a good public standing for any company. As such, it is important to understand, analyse and reflect upon the comments, suggestions and opinions people may have about a company.

Objective

We aim to analyse tweets about Airlines, understanding the reasons for their feedback and also ultimately build a classifier based on the text data to separate the tweets into negative and positive feedback.

Problem Statement

Perform Sentiment Analysis and Classify Tweets about 6 major Airlines as Negative, Positive.

Goal of the Study

Through the process of building the classifier, we want to learn about how different models work, and more importantly, which models are suitable to the problem at hand and the reasons for their behaviours.

We want to get an overall view as to the steps involved in building a Machine Learning model, and learn more about the Data Mining pipeline in general.

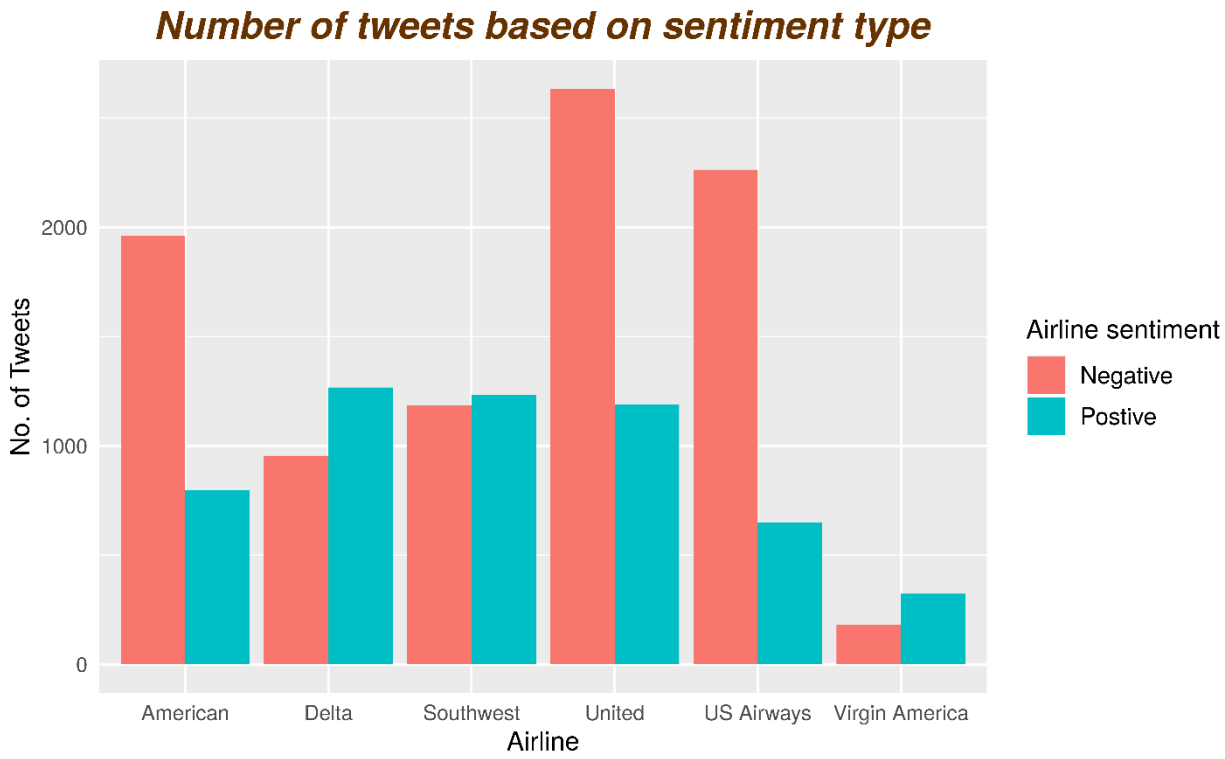
Possible Solution

Since we are dealing with unstructured data it is important to convert it to the proper form to make it ready for modelling. Our possible solutions include, after properly pre – processing the data, building different Supervised Machine Learning Models, and eventually decide on the best model for the problem at hand.

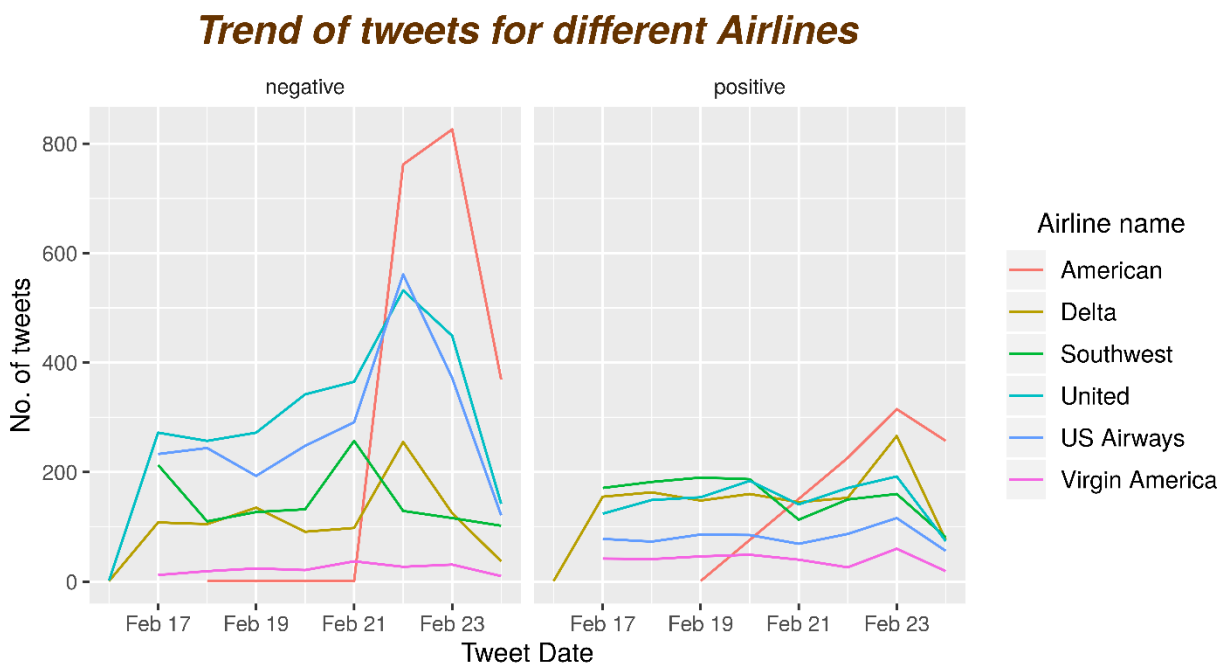
Data Exploration and Visualization

We obtained the data from Kaggle which consisted of Tweets – unstructured text data, along with several other important attributes including userid, date, location, etc.

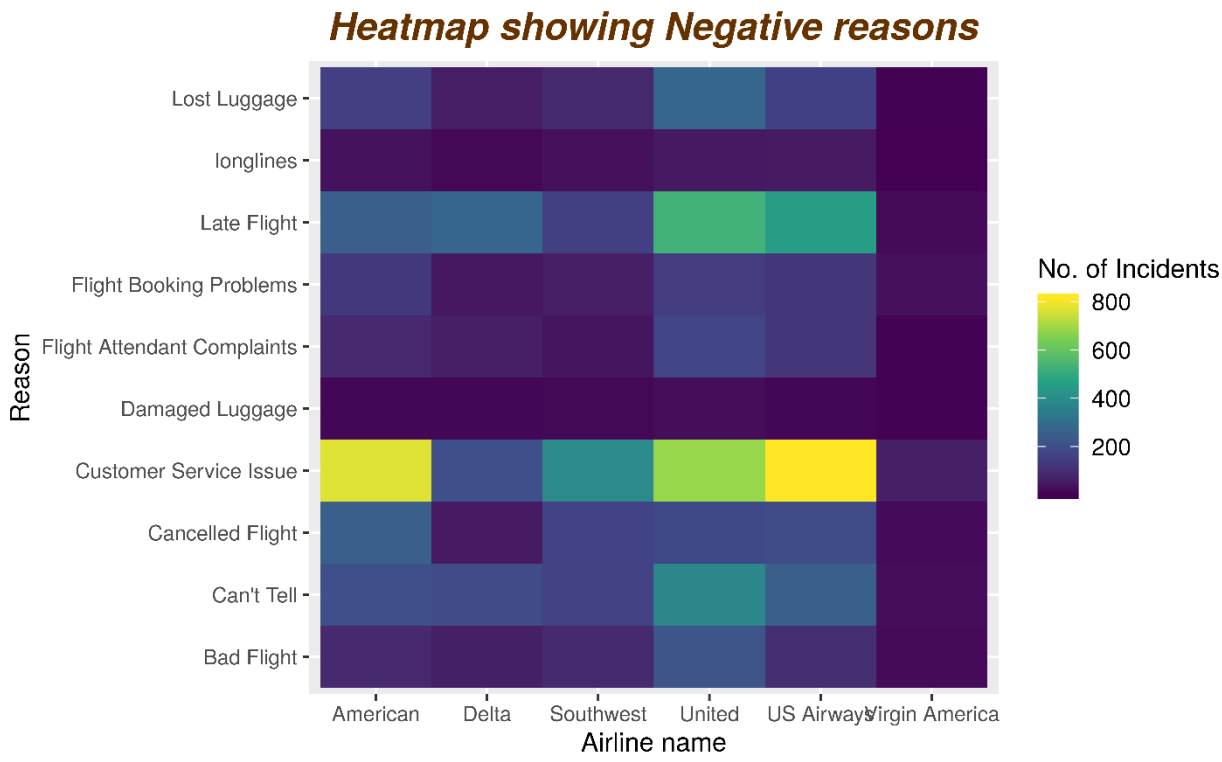
The following exploration helped us understand our data, and draw some useful insights, as highlighted below.



The bar plot shows the number of Positive and Negative tweets of all the airlines, as we can see American, United and US Airways have more Negative tweets compared to Positive Tweets, While Delta, Southwest and Virgin America has more Positive tweets then Negative Tweets. On looking closer we can observe US Airways seems to have the highest value of Negative to positive Tweets and Virgin America has the least negative score.

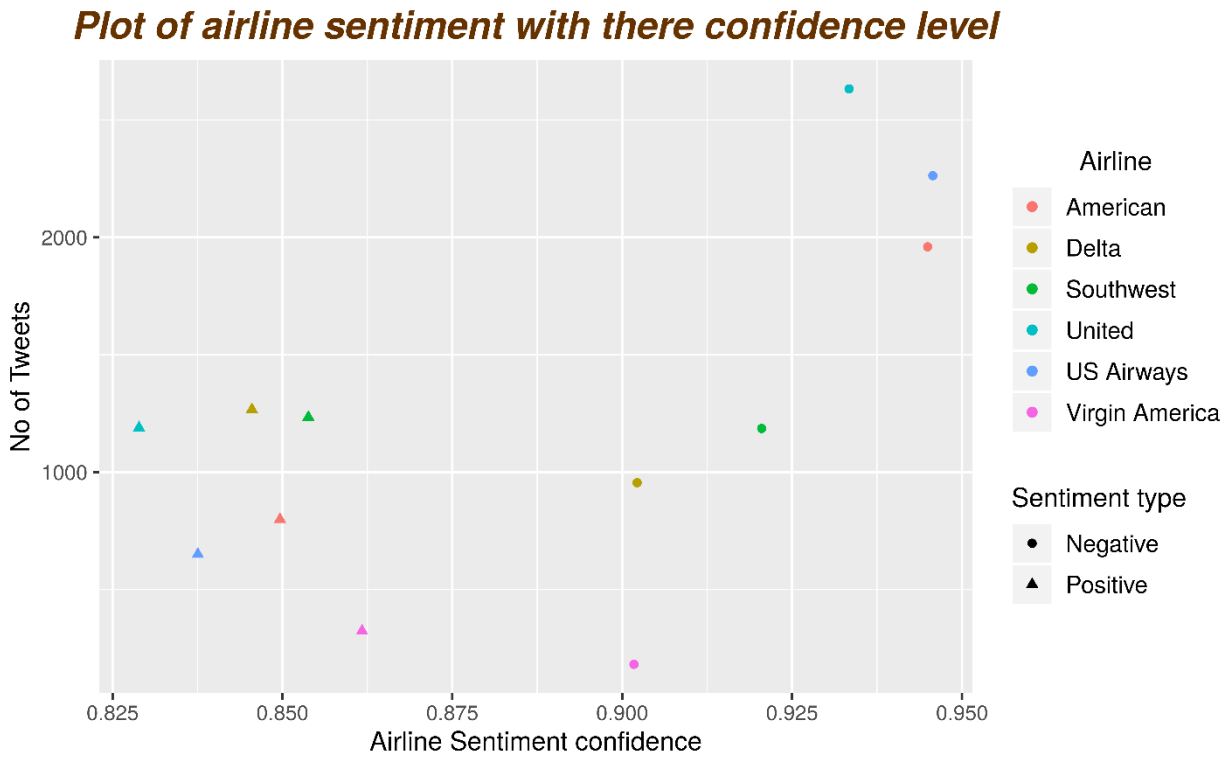


From the trend lines we can make the following observations, Virgin America doesn't show much changes in either category. Delta Airlines has an almost identical variation. United and US Airways shows more fluctuation towards negative sentiment. We can clearly see a spike in the number of negative tweets for American Airlines.



The Heatmap shows the negative reasons for different airlines. We get some interesting information - it shows why passenger sentiment is negative towards an airline. Most common reason for having the negative sentiment is due to poor customer service. Many of the

passengers have expressed they have an issue with the customer service of American, United and US Airways. The second reason seems to be because of Late flight, followed by other reasons such as Lost Luggage, Bad flight, etc.



The above plot shows the Average Airline sentiment confidence Probability of different airlines for negative and positive Tweets. American, United and US Airways have the highest confidence levels of being negative, while their Positive counterparts have low positive confidence.

The data analysis above helps us understand the data, and the important reasons and characteristics for feedback, as well as giving us an overall picture of how each airline fares.

Since this is a case study primarily to learn, we wanted to take the opportunity to try to classify unstructured data, which is one the more challenging tasks in Machine Learning.

We removed any relation to airlines and tried to build a pure “text” classifier which will not be influenced by the airline they were tweeting about.

Data Preparation and Pre-Processing

Cleaning the data

Since our data is Text data, which is unstructured, the features of the data are unclear.

And for any Machine Learning model to be trained, we need features, or characteristics for the data in question. Using these features, which we would eventually derive through pre-processing, we would be able to train the model for our purposes.

It should be kept in mind that for any incoming data, which needs to be classified, the same pre-processing steps need to be applied, and hence, we might need to incorporate these pre-processing steps in our data pipeline.

The steps involved in cleaning are as follows:

- **Converting to Corpus:** In this step, we convert the comments into a Corpus, where each document represents each comment.
- **Changing to lowercase:** Since there is no real difference between the words “Toilet” and “toilet”, we convert everything to lowercase to make sure the machine understands that they are the same word.
- **Removal of punctuation:** To a machine, punctuation just adds more characters that need to be considered, while not contributing anything to the meaning of the text itself. Hence, punctuation is also removed.
- **Removal of stopwords:** There are a lot of unnecessary words such as “the”, “he”, etc. in the English language that do not need to be processed to understand the gist of the comment. Hence, these are also removed to further trim the data, which would aid in faster training of the models.
- **Stemming:** Stemming involves converting a word to its base form to avoid considering different terms of the same word as completely different words, which might result in us not being able to capture the information properly. Hence, stemming is preformed to avoid this and bring all the words to their base form, understood by a machine.

The above steps make up what we call the “cleaning” of text data. These steps are meant to prepare the data to extract its features. In some cases, whitespaces are also stripped, but we feel that the separation of words would help us pin down more easily the reasons for a particular sentiment.

Converting to Word Vectors or Bag of Words

For text data, after the cleaning is done, we need to convert them into a form which would help us derive meaningful characteristics or features from it. This is achieved through word vectors.

Each vector consists of words separated into columns, with each column consisting of the frequency of the word in that particular comment. Each row in the vector would comprise of each of the original comments(document).

As one might be able to imagine, this results in values of the matrix consisting of a lot of zeroes, called a sparse matrix. Since they don't really contribute much to the features, it is better to remove such columns from the matrices.

Conversion into a Data Frame

The final step in the pre-processing is to convert the above matrix into a valid Data Frame, which can then be used to train our traditional models. Here, the feature we extracted from unstructured text data is the frequency of words in each comment, which is then used to train our models.

Modification of Factor levels

Since for any airline, it is more important to analyse the negative feedback, than it is to analyse positive and neutral feedback, we converted the positive and neutral sentiment to factor level 0 and the negative sentiment into factor level 1.

Preparation of Training and Test sets and Benchmark

We then split the data into 60% training data and 40% test data. The benchmark accuracy for the test data is about 62%, which involves classifying everything as the majority class, in this case, negative sentiment.

Data Mining Techniques and Implementation

System Specifications

16GB RAM i7 9th gen CPU

Problem Statement

The problem we looked to solve was to see how well we could classify the tweets of different users into negative sentiment or positive/neutral sentiment using a dataset from Kaggle.

We implemented the following techniques in an attempt to solve this problem, and to learn in the process the advantages and drawbacks of each method.

- Logistic Regression
- KNN Classification
- Neural Nets
- Support Vector Machine
- Linear Discriminant Analysis
- Decision Tree
- Random Forest

Logistic Regression

Since it is a binary classification, logistic regression is one of the first models we wanted to test. Also, it is one of the most basic models which can be employed for this purpose.

```
          Reference
Prediction    1    0
         1  583 1667
         0 3039  567

               Accuracy : 0.1964
                 95% CI : (0.1863, 0.2068)
    No Information Rate : 0.6185
    P-Value [Acc > NIR] : 1

                  Kappa : -0.5236

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.16096
            Specificity : 0.25380
         Pos Pred Value : 0.25911
         Neg Pred Value : 0.15724
             Prevalence : 0.61851
         Detection Rate : 0.09956
   Detection Prevalence : 0.38422
      Balanced Accuracy : 0.20738

       'Positive' Class : 1
```

The accuracy achieved is very low – abysmal to be more precise. The failure of logistic regression might be due to the high number various dimensionalities to factor, which logistic regression has some trouble to do so.

KNN Classification

This method takes about 22 minutes to train. The accuracy achieved is also about 62 percent, which is equal to the benchmark accuracy. The sensitivity is also quite low, with the specificity being oddly high.

```
          Reference
Prediction    1    0
         1 1586  206
         0 2036 2028

               Accuracy : 0.6171
                 95% CI : (0.6046, 0.6296)
    No Information Rate : 0.6185
    P-Value [Acc > NIR] : 0.5908

                  Kappa : 0.2988

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.4379
            Specificity : 0.9078
         Pos Pred Value : 0.8850
         Neg Pred Value : 0.4990
             Prevalence : 0.6185
         Detection Rate : 0.2708
   Detection Prevalence : 0.3060
      Balanced Accuracy : 0.6728

       'Positive' Class : 1
```

Neural Nets

As we know, Neural nets are good function approximators. In such a case, where we have a large number of predictor variables, it seemed like a good model to try out which can hopefully capture some hidden relationship between the predictors and the dependent variable.

```
          Reference
 Prediction    1    0
         1 2713  478
         0  909 1756

               Accuracy : 0.7631
                 95% CI : (0.752, 0.774)
    No Information Rate : 0.6185
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.516

  Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7490
            Specificity : 0.7860
         Pos Pred Value : 0.8502
         Neg Pred Value : 0.6589
             Prevalence : 0.6185
         Detection Rate : 0.4633
   Detection Prevalence : 0.5449
      Balanced Accuracy : 0.7675

       'Positive' Class : 1
```

This model takes about 4-5 minutes to train. As can be observed, the accuracy is 76.31%. The sensitivity is also quite low.

Support Vector Machine

Considering that SVMs are prone to overfitting, and suffer if the number of predictors is larger than the number of samples, we wanted to give it a try on our dataset, which can be called medium at best in terms of size.

```
          Reference
Prediction    1    0
         1 3162  460
         0  717 1517

               Accuracy : 0.799
                 95% CI : (0.7885, 0.8092)
    No Information Rate : 0.6624
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5645

 Mcnemar's Test P-Value : 8.526e-14

            Sensitivity : 0.8152
            Specificity : 0.7673
         Pos Pred Value : 0.8730
         Neg Pred Value : 0.6791
             Prevalence : 0.6624
         Detection Rate : 0.5400
   Detection Prevalence : 0.6185
      Balanced Accuracy : 0.7912

       'Positive' Class : 1
```

The accuracy is about 80% and the sensitivity value is also higher.

Linear Discriminant Analysis

Even though we applied Logistic Regression to our Binary Classification problem, we still wanted to try out LDA as it is said to overcome some of the drawbacks of Logistic Regression. It is also one of the most efficient modelling techniques, and hence requires very little data to work with.

```
          Reference
Prediction    1    0
         1 3058  564
         0  616 1618

               Accuracy : 0.7985
                 95% CI : (0.788, 0.8087)
    No Information Rate : 0.6274
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5711

 Mcnemar's Test P-Value : 0.1376

            Sensitivity : 0.8323
            Specificity : 0.7415
         Pos Pred Value : 0.8443
         Neg Pred Value : 0.7243
             Prevalence : 0.6274
         Detection Rate : 0.5222
   Detection Prevalence : 0.6185
      Balanced Accuracy : 0.7869

       'Positive' Class : 1
```

We get an accuracy of 79.85% with a higher sensitivity rate.

Decision Tree

We wanted to try Decision Trees to see if we can generate simple rules from which we can draw insights so that it is easier for the Airline to take action.

```
          Reference
Prediction    1    0
         1 3465 1710
         0  157  524

               Accuracy : 0.6812
                 95% CI : (0.6691, 0.6931)
    No Information Rate : 0.6185
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2206

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9567
            Specificity : 0.2346
         Pos Pred Value : 0.6696
         Neg Pred Value : 0.7695
             Prevalence : 0.6185
         Detection Rate : 0.5917
   Detection Prevalence : 0.8837
      Balanced Accuracy : 0.5956

       'Positive' Class : 1
```

As can be observed, the accuracy is much lower than the other methods. However, it has an unnaturally high Sensitivity, which implies that it can catch on very accurately to negative feedback.

Random Forest

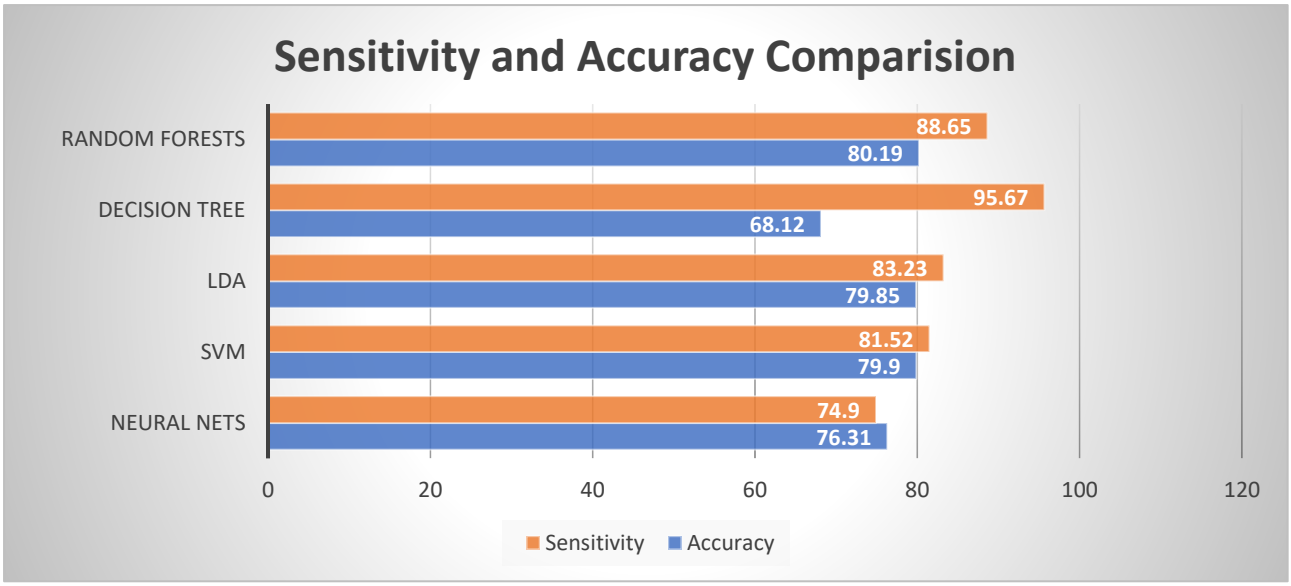
Since Decision Tree resulted in an accuracy we were not satisfied with, we wanted to try one last modelling technique before we started to compare them to one another. Random Forests are highly useful in classification problems, with a large tendency to overfit.

```
          Reference
Prediction    1    0
         1 3211  749
         0  411 1485

               Accuracy : 0.8019
                 95% CI : (0.7915, 0.8121)
    No Information Rate : 0.6185
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5677

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8865
            Specificity : 0.6647
         Pos Pred Value : 0.8109
         Neg Pred Value : 0.7832
             Prevalence : 0.6185
         Detection Rate : 0.5483
   Detection Prevalence : 0.6762
      Balanced Accuracy : 0.7756

       'Positive' Class : 1
```

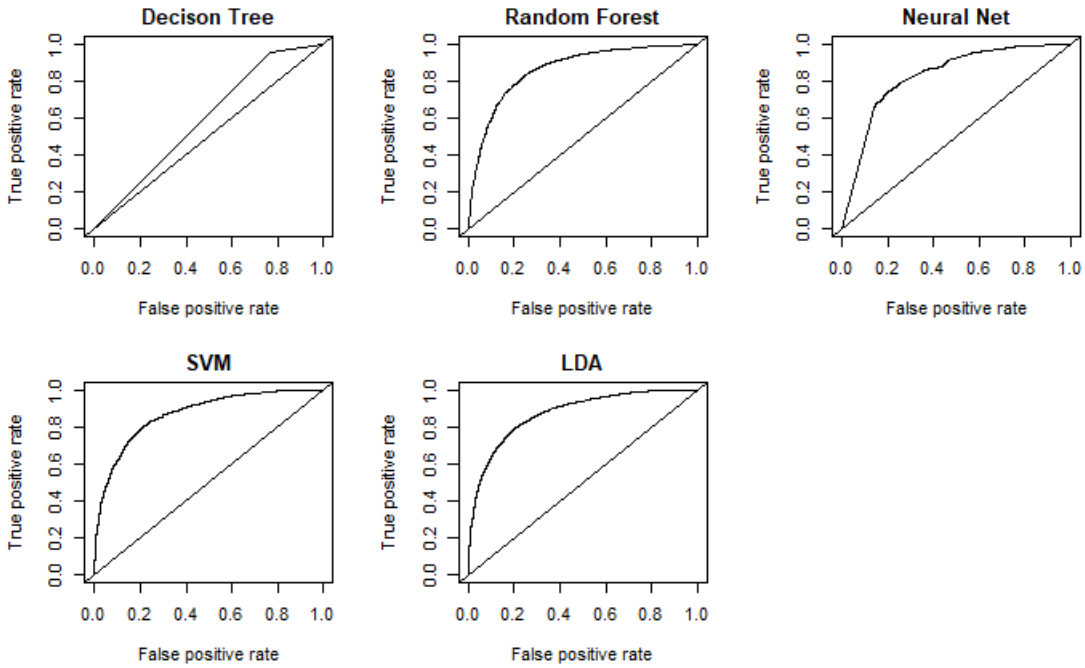
There is a significant increase in accuracy in Random Forests (80.19%) when compared to Decision Trees. The sensitivity is also very high.

Performance Evaluation

From the values obtained above, the models worth taking another look into are Neural Nets, LDA, SVM, Random Forest and Decision Tree. We are including Decision tree, despite its low accuracy, because the high sensitivity implies that it is good at classifying negative feedback, which is a very valuable feature.



The above is chart comparing the models' Sensitivity and Accuracy. We see that Decision Tree has the highest sensitivity, while Random Forest has the highest accuracy.



From the above chart, it is clear that SVM, LDA and Random Forest have the highest AUC values.

```
AUC Values
Neural Net =  0.8277686
SVM =  0.8698031
LDA =  0.8721164
Decision Tree =  0.5956053
Random Forest =  0.8614004
```

The above AUC values correspond to what we deduced from the ROC curve. The AUC values are quite similar to each other.

Discussion and Recommendation

Overall, we feel selecting the Random Forest model would be the most optimal. The accuracy is almost equal to other models (SVM and LDA), however, it has the second highest sensitivity among all models. Since our objective was to check if we can successfully classify negative feedback first, its sensitivity, coupled with its accuracy, it is the most sensible model.

However, if we were to prioritize complete negative feedback classification, we would recommend Decision Tree model.

Another special mention would be KNN Classification - though the accuracy is lower than the benchmark accuracy, it has a high specificity value implying that it can correctly classify positive feedback.

One more thing to try out might be clustering – since we know that there are at least 2 discrete groups, perhaps clustering might produce some useful results, to which a Supervised Machine Learning model (for example, LDA) can be applied, which may result in a far more superior model.

Summary

The objective of our case study was to try to classify a tweet, which is unstructured text data, into positive or negative sentiment.

We went ahead and decided to prioritize negative feedback as it is more important to analyse such sentiment and improve upon it to make any required changes to further grow as a company.

We pre - processed text data to convert it to appropriate form from which features could be extracted, that is, we converted the text data into a corpus, and then later into word vectors.

From these word vectors, we were able to extract the frequency of the words as features, thus making the data ready for building Machine Learning models.

We built different Machine Learning models, tried out various configurations (especially for neural nets and SVM), and eventually settled on what we felt were the best models for our problem.

We did not try out Bayesian Classification as it requires only categorical variables, and though it possible to create dummy variables for the frequency of each word, we felt it would become computationally too extreme to infer anything from the data.

The performance evaluation was very valuable as were able to learn a lot about how one model works better than the other, and get a sense as to what type of model is suitable for our particular problem.

Overall, it was a very instructive case study as were able to experience the whole Machine Learning pipeline and gained valuable insight into different models.

Appendix: R Code

Data Visualization and Exploration

```
library(dplyr)
library(tidyr)
library(lubridate)
library(ggplot2)
```

```{r}
Tweets$airline_sentiment[Tweets$airline_sentiment=="neutral"]<-"positive"
Tweets%>%
  group_by(airline,airline_sentiment)%>%
  summarise(count=n())%>%
  ggplot(aes(y=count,x=airline,fill=airline_sentiment))+
  geom_bar(stat="identity",position = "dodge")+
  ylab("No. of Tweets")+
  xlab("Airline")+
  scale_fill_hue(name="Airline sentiment",labels=c("Negative","Postive"))+
  ggtitle("Number of tweets based on sentiment type")+
  theme(plot.title = element_text(hjust = .5,
                                  face="bold.italic",color="#663300",size=16),
        legend.title = element_text(hjust = .5),
        legend.text = element_text(size = 10),
        strip.background = element_blank(),
        axis.ticks = element_blank())
ggsave("Bar.png",dpi = 1080)
```

```
```{r}

```
Tweets%>%
  group_by(airline,airline_sentiment,tweet_created)%>%
  summarise(count=n())%>%
  ggplot(aes(x=tweet_created,y=count,colour=airline))+geom_line()+
  facet_wrap(~airline_sentiment)+
  xlab("Tweet Date")+
  ylab("No. of tweets")+
  scale_color_hue(name="Airline name")+
  ggtitle("Trend of tweets for different Airlines")+
  theme(plot.title = element_text(hjust = .5,
                                  face="bold.italic",color="#663300",size=16),
        legend.title = element_text(hjust = .5),
        legend.text = element_text(size = 10),
        strip.background = element_blank())


ggsave("Trend.png",dpi = 1080)
```

```{r}
Tweets%>%
  group_by(negativereason,airline)%>%
  summarise(count=n())%>%na.omit()%>%
  ggplot(aes(x=airline,y=negativereason,fill=count))+
  geom_tile()+
  scale_fill_continuous(name="No. of Incidents",type = "viridis")+
  ylab("Reason")+
```

```
  xlab("Airline name")+
  ggtitle("Heatmap showing Negative reasons")+
  theme(plot.title = element_text(hjust = .5,
                                  face="bold.italic",color="#663300",size=16),
        legend.title = element_text(hjust = .5),
        legend.text = element_text(size = 10),
        strip.background = element_blank())
ggsave("Heatmap.png",dpi = 1080)
```

```{r}

Tweets%>%
  group_by(airline,airline_sentiment)%>%
  summarise(count=n(),values=mean(airline_sentiment_confidence))%>%na.omit()%>%
  ggplot(aes(x=values,y=count,shape=airline_sentiment,colour=airline))+
  geom_point()+
  ylab("No of Tweets")+
  xlab("Airline Sentiment confidence")+
  ggtitle("Plot of airline sentiment with there confidence level")+
  theme(plot.title = element_text(hjust = .5,
                                  face="bold.italic",color="#663300",size=16),
        legend.title = element_text(hjust = .5),
        legend.text = element_text(size = 10),
        strip.background = element_blank())+
  scale_color_hue(name="Airline")+
  scale_shape(name="Sentiment type",label=c("Negative","Positive"))
ggsave("confidence.png",dpi = 1080)
```

Preprocessing and Model Implementation

```
library(caret)
library(gmodels)
library(forecast)
library(ggplot2)
library(stringr)
library(readr)
library(dplyr)
library(tm)
library(SnowballC)
library(randomForest)
library(neuralnet)
library(e1071)
library(MASS)
library(ROCR)
library(rpart)
library(rpart.plot)

txtprproc <- function(x){

x <- Corpus(VectorSource(x))  # Converting to Corpus
x <- tm_map(x,PlainTextDocument)  # Plain Text Document creation
x <- tm_map(x,tolower)  # Converting the data to lower case.
x <- tm_map(x,removePunctuation)  # Removing Punctuation
x <- tm_map(x,removeWords,c(stopwords(kind = "en"))) # Removing unnecessary stopwords.
x <- tm_map(x,stemDocument)  # Creating stems of words to remove inflections.
return(x)
}

twts <- twts[,c("tweet_id","airline_sentiment","airline","text")]
```

```
twtscorp <- txtprproc(twts$text)


twtscorpfrq <- DocumentTermMatrix(twtscorp)
twtscorpfrq <- removeSparseTerms(twtscorpfrq,0.995)
twtscorpfrqdf <- as.data.frame(as.matrix(twtscorpfrq))

colnames(twtscorpfrqdf) = make.names(colnames(twtscorpfrqdf))
twtscorpfrqdf$sent <- twts$airline_sentiment

twtscorpfrqdf$sent <- factor(twtscorpfrqdf$sent,levels = c("negative","neutral","positive"),labels = c(1,0,0))

twtscorpfrqdf$virginamerica = NULL
twtscorpfrqdf$delta = NULL
twtscorpfrqdf$southwest = NULL
twtscorpfrqdf$southwestair = NULL
twtscorpfrqdf$american = NULL
twtscorpfrqdf$americanair = NULL
twtscorpfrqdf$usairway = NULL
twtscorpfrqdf$jetblu = NULL
twtscorpfrqdf$â.œjetblu = NULL


set.seed(100)
sample <- sample.int(n = nrow(twtscorpfrqdf),size = floor(0.6*nrow(twtscorpfrqdf)),replace = FALSE)
tr1 <- twtscorpfrqdf[sample,]
te1 <- twtscorpfrqdf[-sample,]
```

prop.table(table(te1$sent))

Logistic Regression

```{r }
logrgrfit <- glm(sent~.,data = tr1,family = binomial(link = "logit"))

logrgrpred <- predict(logrgrfit,te1,type = "response")

logrgrclpred <- ifelse(logrgrpred > 0.5,1,0)
confusionMatrix(factor(logrgrclpred),te1$sent)

```

KNN - This will take about 20-22 minutes to train.
```{r }

ktr1_lbls <- tr1$sent
knntr1 <- tr1
knntr1$sent <- NULL
kte1_lbls <- te1$sent
knnte1 <- te1
knnte1$sent <- NULL
knnclass <- train(knntr1,ktr1_lbls,method = "knn")
plot(knnclass)
knnpredte1 <- predict(knnclass,knnte1)
```

confusionMatrix(factor(knnpredte1),kte1_lbls)

```

Decision Tree

```{r }
dtfit <- rpart(sent~.,data = tr1,method = "class")
dtpred <- predict(dtfit,te1,type = "prob")
dtclpred <- predict(dtfit,te1,type = "class")
rpart.plot(dtfit)
confusionMatrix(dtclpred,te1$sent)

```

DT ROC

```{r }
dtpred <- data.frame(dtpred)
dtperf <- prediction(dtpred$X1,te1$sent)
plot(performance(dtperf,"tpr","fpr"))
abline(a=0,b=1)

```

Random Forest

```{r }
rffit <- randomForest(sent~.,data = tr1)
rfpred <- predict(rffit,te1,type = "prob")
rfclpred <- predict(rffit,te1,type = "class")

confusionMatrix(rfclpred,te1$sent)

```

RF Roc

```{r }

rfpred <- data.frame(rfpred)

rfperf <- prediction(rfpred$X1,te1$sent)

plot(performance(rfperf,"tpr","fpr"))

abline(a=0,b=1)

```

Neural Nets

```{r }

twtsnn <- twtscorpfrqdf

twtsnn$negative<-twtsnn$sent==0

twtsnn$positive<-twtsnn$sent==1

set.seed(100)

sample <- sample.int(n = nrow(twtsnn),size = floor(0.6*nrow(twtsnn)),replace = FALSE)

tr1nn <- twtsnn[sample,]

te1nn <- twtsnn[-sample,]

```

```{r }

twitter.nn <- neuralnet(negative+positive~., data = tr1nn[,-which(names(tr1nn)=="sent")],

linear.output = F, hidden = 2)

twitter.nn.predict <- compute(twitter.nn, te1nn[,-which(names(te1nn)=="sent"|names(te1nn)=="negative"|names(te1nn)=="positive")])

predicted.class=apply(twitter.nn.predict$net.result,1,which.max)-1

df<-as.factor(predicted.class)

confusionMatrix(df, te1nn$sent)

```

NN Roc

```{r }
nnpred <- predict(twitter.nn,te1,type = "class")
nnpred <- data.frame(nnpred)
nnperf <- prediction(nnpred$X2,te1$sent)
plot(performance(nnperf,"tpr","fpr"))
abline(a=0,b=1)

```

SVM

```{r }
twitter.svm <- svm(tr1[,-which(names(tr1)=="sent")], tr1[,which(names(tr1)=="sent")],probability = TRUE)
twitter.svm.pred <- predict(twitter.svm, te1[,-which(names(te1)=="sent")])
confusionMatrix(te1$sent,twitter.svm.pred)

```

SVM Roc

```{r }
svmprob <- predict(twitter.svm,te1[,-which(names(te1)=="sent")],probability = TRUE)
svmprob <- attr(svmprob,"probabilities")
svmprob <- data.frame(svmprob1)

svmperf <- prediction(svmprob$X1,te1$sent)
plot(performance(svmperf,"tpr","fpr"))
abline(a=0,b=1)
```
```

```

LDA

```{r }
twittet.lda <- lda(sent~., tr1)
twittet.lda.predict <- predict(twittet.lda, te1[,-which(names(te1)=="sent")])
prediction<-as.factor(twittet.lda.predict$class)
confusionMatrix(te1$sent,prediction)
```

```

LDA ROC

```{r }
ldaprob <- data.frame(twittet.lda.predict$posterior)
ldaperf <- prediction(ldaprob$X1,te1$sent)
plot(performance(ldaperf,"tpr","fpr"))
abline(a=0,b=1)
```

```

ROC
```

```{r  }
par(mfrow = c(2,3))
plot(performance(dtperf,"tpr","fpr"),main = "Decison Tree")
abline(a=0,b=1)
plot(performance(rfperf,"tpr","fpr"),main = "Random Forest")
abline(a=0,b=1)
plot(performance(nnperf,"tpr","fpr"),main = "Neural Net")
abline(a=0,b=1)
plot(performance(svmperf,"tpr","fpr"),main = "SVM")
abline(a=0,b=1)
plot(performance(ldaperf,"tpr","fpr"),main = "LDA")
abline(a=0,b=1)

```

AUC

```{r }
nnauc <- performance(nnperf,measure = "auc")
svmauc <- performance(svmperf,measure = "auc")
ldaauc <- performance(ldaperf,measure = "auc")
dtauc <- performance(dtperf,measure = "auc")
rfauc <- performance(rfperf,measure = "auc")
cat("AUC Values")
cat("\nNeural Net = ",nnauc@y.values[[1]])
cat("\nSVM = ",svmauc@y.values[[1]])
cat("\nLDA = ",ldaauc@y.values[[1]])
cat("\nDecision Tree = ",dtauc@y.values[[1]])
cat("\nRandom Forest = ",rfauc@y.values[[1]])
```