DJIA PREDICTION USING TEXTUAL ANALYSIS OF DAILY NEWS

ISEN 613 – EDA PROJECT REPORT

Pratik Iyer

Nishant Sinha

Saurabh Rawool

Vani Pujari

Contents

1. Executive Summary

Amongst the many known factors that has a huge impact on the movement of stock prices, one of them is the daily news and its effect on the people's sentiment who are in the trading business. In this project we have aimed to develop a predictive model that can read through the daily news and predict DJIA movement for the next day. In short, the **objective** is to eradicate the hassle of manually reading the news and predicting the market movement as done by the traders.

Previous work and research has been more on the sentiment analysis but our research is more on the frequency and its weight and how it impacts the overall text. The more the word is used in the news text the better its frequency and thus has more impact on the prediction. This is how we have done things differently in our research. Using text mining we have transformed the data and applied machine learning algorithms like **Extreme gradient boosting (XGBoost), Support Vector Machine (SVM), Random Forest** and **Ridge regression**. To get a better understanding of our research and finding the best possible model, we have used different NGram tokenizers. The scope of project covers unigram, bigram, trigram, 4gram and few others.

The biggest **challenge** we faced were the vast nature of text mining and research that goes into it. There are so many algorithms at our disposal but finding the right takes a lot of effort, time and cost. After creating 72 models and analyzing them on the basis of 5 metrics i.e. accuracy, misclassification rate, AUC, ROC and confusion matrix the best model for our dataset was extreme gradient boosting. The AUC of that model was well above 0.5 which indicates that our model is reliable and better than random guessing. It definitely can be used as a system to predict next days' index movement of DJIA by traders thus fulfilling our objective.

2. Introduction

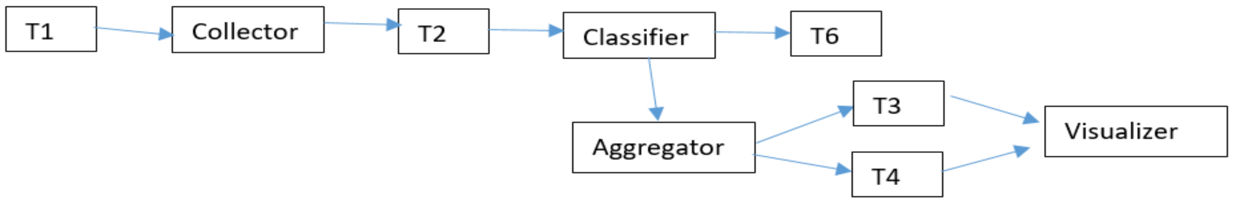
There has been numerous research made on developing predictive model for stock market prediction. One thing all the research agree on is it is extremely difficult to create predictive model that can accurately predict market index movement. To speak at the basic level, there are mainly two approaches that is used for this prediction in real world: Fundamental and Technical approaches [1]. In the fundamental approach, data of involving the earnings, investors share and news determine future forecasts. Our scope is to use the textual analysis of the news. News such as global events, company's quarterly report can be hugely impacting the index of Dow Jones. We have tried to find the keywords from the headlines that can have an impact and then create a sparse matrix of these keywords. Now these keywords become our predictors and the labels become our observations. Multiple algorithms in combination with different tokenizers is tried and the best model is found on the basis of parameters like misclassification rate, accuracy, ROC, AUC and confusion matrix. Previously, similar research has been made where the models were created using news and stock movement of the same day was predicted. The **objective** of our research is: (1) predict the movement for the next day by analyzing current news, (2) predict the movement for the next by analyzing news from past three days, and (3) predict the movement for the next by analyzing news of past 5 days. At the end of the research we will have 72 models to compare and find the best fit.

3. Literature review

Reviewed by: Nishant Sinha
Complete reference: News Analytics and Sentiment Analysis to Predict Stock Price Trends[1]
Objective:
Key problem - The key idea of the paper is to predict the sentiment associated with the stock prices from the news available and also find how strong is this association.
Reason for addressing this problem - The news is always coming in and changing every now and then

[1] "News Analytics and Sentiment Analysis to Predict Stock Price Trends." http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.433.1426&rep=rep1&type=pdf. Accessed 12 Dec. 2016.

which in turn effects the sentiments associated with the stock. Sentiment becomes the pivotal point for decision making for the investors. It is difficult for investors to predict change in index as well changes in the volume by manually analyzing the news.
Key objective - News has always been a huge contributor in building perception about the market but due to a huge volume of news available it is becoming difficult for the investors to draw any idea from it.
How are authors addressing this problem? Authors have compiled news from different sources, created a model to forecast the market movement and thus help investors get valuable information so that they can gain maximum profit from their investment.
Findings and implications - They were able to find a strong correlation of 67% between positive sentiment and stock price. Also, the accuracy rate for the model is 70% with Mathew' correlation coefficient of 0.4. It implies that news carries huge correlation with the market movement and thus can be a very important factor to predict the change in the prices of stock and its volume. The news, overall, makes up a sense for investor to harness its vitality.
Approach:



Results: For evaluating the model accuracy, the author has shown the confusion matrix.

Confusion Matrix		Target	
		Positive	Negative
Model	Positive	330	144
	Negative	53	132

The sensitivity and specificity of the model comes out to be 86.16 and 47.83 with the overall accuracy of 70.1. The accuracy is moderate thus leaves a lot of room for improvement. The correlation coefficient is 0.4 and coefficient of variance is 1.009.

Summary:

- Every news has some information relevant to the specific company which helps the model build a perception about its growth and performance. A trader will be willing to put in more money if the model gives a positive sentiment about the company's news or pull out invested money from that company if the sentiment is negative.
- Whenever the model is ran using some set of news particular to a company it calculates the sentiment with an average accuracy of 70%. Also, the bias in the news is less as it is all real time.

Reviewed by: Nishant Sinha
Complete reference: CS224N Final Project: Sentiment analysis of news articles for financial signal prediction [2]
Objective:
Key problem - The volatility of the stock market has led the authors to work on a way to predict its movement. In this article they have discussed how this fluctuation are in correlation with the news report.
Reason for addressing this problem - The crowd that invests in stock market heavily relies on how the people in the world feel about the market or a share in particular.
Key objective - Traders uses speed reading computers to scan through the news or blogs and then depends on the machine to tell how it is affecting the market.
How are authors addressing this problem? The author has assessed two approach: manual and

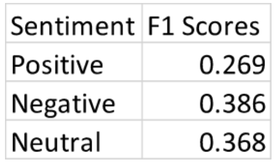
[2] "CS224N Final Project: Sentiment analysis of news articles for financial ...." http://nlp.stanford.edu/courses/cs224n/2011/reports/nccohen-aatreya-jameszjj.pdf. Accessed 12 Dec. 2016.

automatic. The author uses human to read the news and produce sentiments accordingly while in the automatic approach the sentiments are calculated from the predictive model.

Findings and implications - They found that the extraction of sentiment from the relevant news is a tedious task. The news is the most important source of such kind of project but it difficult to analyze them. The model gave better results for training data but were less effective for predictive analytics i.e. testing data.

Approach: The dataset was not large enough for automatic classification so S&P 500 was taken for dataset. Using log terms for creating the model volatility of the model was tested. The classification of the news was done on the basis of one keyword: stock. Then the metadata was filtered by taking out title, body and individual tags. This improved the F1 score but was still below the par. The predictive capability of model was analyzed based on the sentiment it produced by analyzing the news for that particular day. Every article was tagged +1 for positive, 0 for neutral and -1 for negative, summed the scores for the day and thus an overall sentiment for that day was calculated.

Results: The results are not satisfactory for news articles in correlation with the stock market prediction.



Sentiment	F1 Scores
Positive	0.269
Negative	0.386
Neutral	0.368

The table here shows the value of F1 score for the three kind of sentiments for unigram and bigram sentiment analysis. The values are poor and thus tells us that the articles and news are not the best predictors for this analysis. This could be because the journalists who wrote these articles used complex sentences with mix of positive and negative annotations and thus making it difficult for the system to give the sentiment attached to it. The author believes that this could perform better if there was a larger system for model to be tested or a new approach like moving average would result with better efficiency.

Summary:

- There is a chance that the correlation of S&P news and market sentiment is poor when compared to The NY Times article and market sentiment. The accuracy is 70% which is lower than average and thus model is considered to be poor.
- The other method that have been used in the past have shown efficiency of less than 50% and simulated a return on the trading around 2%.

Reviewed by: Saurabh Rawool

Complete reference: Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFinText System[3]

Objective:

Key problem - To predict the direction and magnitude of stock movement 20 minutes after the release of news article. This paper discusses different predictive machine learning approaches to analyze the financial news and predict the stock price using different methodologies of textual representation.

Reason for addressing this problem - Stock market prediction is important to investors since it can lead to higher gains and prevent losses. Greater the prediction accuracy, greater is the investor safeguarded from loss (in case of downward market movement) and has an opportunity to make profit (in case of upward moving market).

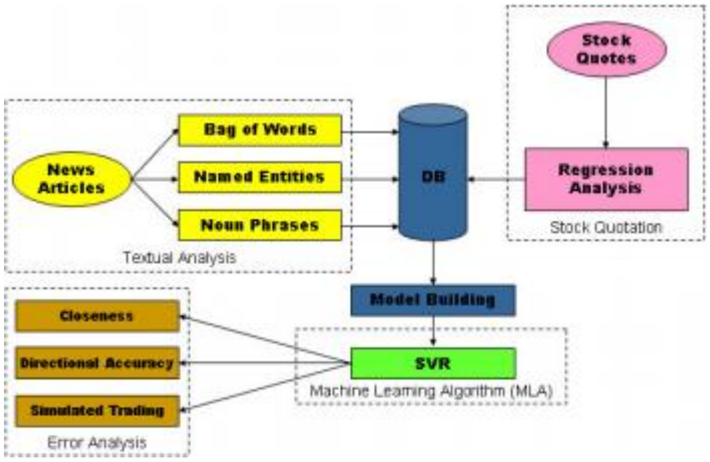
Key objective - The key objective of the paper is to compare different textual representation techniques & algorithms and test their accuracy in terms of predicting stock prices and direction of movement.

How are authors addressing this problem? The authors gathered around 9211 financial news articles and 10,259,042 stock quotes over 5week period. They tailored the Support Vector Machine derivatives to run with numeric predictions and stock specific variables.

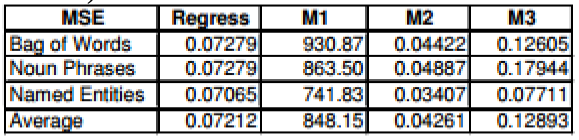
Findings and implications - They found that statistical model containing actual stock price at the time of article release and article terms did a better job at predicting the future stock price with MSE of 0.04261 and directional accuracy of 57.1%. Also they found out that proper noun textual representation of articles works better than bag of words representation.

Approach:

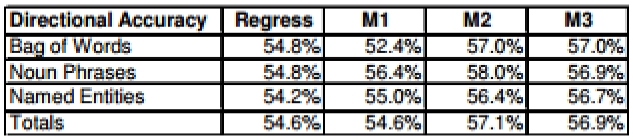
[3] "Textual Analysis of Stock Market Prediction Using ... - Semantic Scholar." https://pdfs.semanticscholar.org/5386/adf8abdaabc6f6b27c7b37ca29a3a8f760af.pdf. Accessed 6 Dec. 2016.



Results: The following table shows the error for three textual representations in terms of three dimensions of analysis - measure of closeness (left table) & directional accuracy (right table). It can be seen that model M2 (which includes stock price at the time of news release and the article terms outperformed the other models)



MSE	Regress	M1	M2	M3
Bag of Words	0.07279	930.87	0.04422	0.12605
Noun Phrases	0.07279	863.50	0.04887	0.17944
Named Entities	0.07065	741.83	0.03407	0.07711
Average	0.07212	848.15	0.04261	0.12893



Directional Accuracy	Regress	M1	M2	M3
Bag of Words	54.8%	52.4%	57.0%	57.0%
Noun Phrases	54.8%	56.4%	58.0%	56.9%
Named Entities	54.2%	55.0%	56.4%	56.7%
Totals	54.6%	54.6%	57.1%	56.9%

Summary: Major takeaways from this article is as follows-

- A combination of vector representation of text and bag of words can be used for better analysis of text data.
- There is evidence that news data has reasonable impact on the price of the shares.

Reviewed by: Saurabh Rawool

Complete reference: STOCK TREND PREDICTION USING NEWS SENTIMENT ANALYSIS[4]

Objective:

Key problem - This paper is trying to use non quantifiable data like company news to predict the future trend for its stock price. This can help investors process large amount of historical as well as current news data without having to manually read them.

Reason for addressing this problem - It is difficult for individual investor to digest all historical and current information about the company and make future investment decision. So this project aims at analyzing all the information using natural language processing and guide the investor with his decision making.

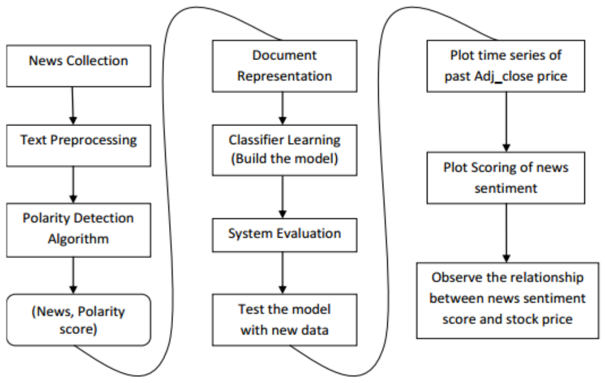
Key objective - To analyze the historical and current news data of a company and predict the sentiment of the people with respect to company stock. More the positive sentiment, better are the chances of the stock to go up.

How are authors addressing this problem? - The authors are detecting polarity of news articles by using bag of words (dictionary based) approach in which they have a dictionary of words from financial world which help them determine the sentiment of the news. These marked bag of words are then fed to text classification algorithms to determine the direction of stock movement on the date of the article release.

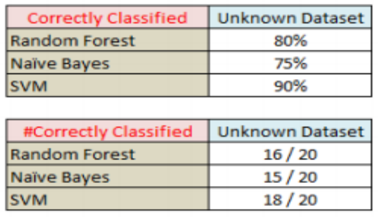
Findings and implications - They found that they were able to predict future stock trends with reasonable accuracy. This implies that the news sentiment does help in shaping the investor's decision in buying or selling stock of the company.

Approach: The diagram below describes the steps used by authors.

[4] "stock trend prediction using news sentiment analysis - arXiv.org." https://arxiv.org/pdf/1607.01958. Accessed 9 Dec. 2016.


News Collection
Text Preprocessing
Polarity Detection Algorithm
(News, Polarity score)
Document Representation
Classifier Learning (Build the model)
System Evaluation
Test the model with new data
Plot time series of past Adj_close price
Plot Scoring of news sentiment
Observe the relationship between news sentiment score and stock price

Results: The SVM model outperformed all the three algorithms that they tested.



Correctly Classified	Unknown Dataset
Random Forest	80%
Naive Bayes	75%
SVM	90%

#Correctly Classified	Unknown Dataset
Random Forest	16 / 20
Naive Bayes	15 / 20
SVM	18 / 20

Figure 3: Result of testing models with new data

Summary:

- Random Forest and SVM are efficient algorithms when it comes to text classification.
- There is enough evidence that news shape investors outlook towards the stock of a company and thus they are correlated.

Reviewed by: Pratik Iyer
Complete Reference: A gentle introduction to text mining using R[5]
Objective:
Key Problem - Text mining capabilities of R, which is open source statistical software, does not always work as it is advertised. This paper aims at dealing with the basic concepts of text mining capabilities and gives a novice introduction about the same. It also talks about the capabilities and limitations of text mining and how a user can go about making use of text mining in R.
Reason for addressing this problem - Data is usually stored in tables in the form of rows and columns (traditional RDBMS). It becomes tedious if the user wants to add an extra column to the table and all the rows in the table get affected because of it. Also, there might be a case in analysis, where the question is not being answered by the existing entities and thus developers always use a catchall column to store any such information. Predictably, this can be of multiple lines and in some cases as long as a paragraph and it's not possible to analyze them using traditional RDBMS tools.
Key Objective - The key objective of this paper is to introduce its readers to the basic, intermediate and advanced concepts of text mining in R and explaining how it can be used in multiple applications for the purpose of data analysis.
How are the authors addressing this problem? The author starts from the basic concepts of installing R, to loading the data in R for analysis. Later on he introduces the concept of stemming where all the words in the text are converted into stem words or their common root. This helps in reducing the redundancy thereby helping in faster analysis.

5 "A gentle introduction to text mining using R | Eight to Late." 27 May. 2015, https://eight2late.wordpress.com/2015/05/27/a-gentle-introduction-to-text-mining-using-r/. Accessed 12 Dec. 2016.

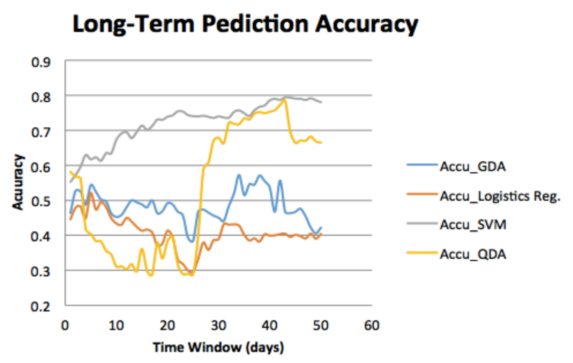
Approach: The author is working for an organization and was involved in doing an in-depth analysis of the data captured by the IT service management tool in his organization. While analyzing this data he realized that a lot of information had been captured in the free text column and it was becoming difficult for him to analyze as he was not sure what exactly he as looking for. He soon realized that this was a classic text-mining problem, which can be solved by using the text mining capabilities of R. The author starts off at a very novice level by introducing the readers to R. Later, he explains how to install R studio on the target system and introduces the basic packages (tm, SnowballC, ggplot2, wordcloud) used in text mining and plotting graphs. The data set that he used was from his own blog. Then he explains how to load the data into R studio and introduces the concept of corpus that is nothing but a collection of documents. Once the corpus is created, the next step is to clean and process the data by removing white spaces, punctuations and other unnecessary details.
Results: The author in this paper demonstrates an example where he returns all the words that have been repeated for more than 80 times in the entire document. He used the *findFreqTerms()* function as follows:

```
findFreqTerms(dtmr,lowfreq=80)
[1] "action" "approach" "base" "busi" "chang" "consult" "data" "decis" "design"
[10] "develop" "differ" "discuss" "enterpris" "exampl" "group" "howev" "import" "issu"
[19] "like" "make" "manag" "mani" "model" "often" "organ" "peopl" "point"
[28] "practic" "problem" "process" "project" "question" "said" "system" "thing" "think"
[37] "time" "understand" "view" "well" "will" "work"
```

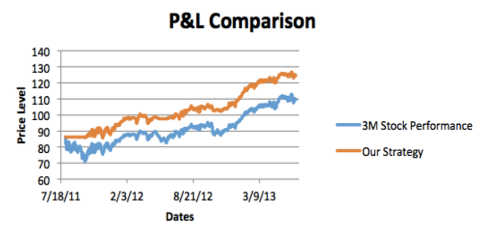
Summary: This paper is a really good starting point for novice users who want to get introduced to the concept of text mining in R. The paper explains all the usages of text mining and is a comprehensive guide to text analysis.

Reviewed by: Pratik Iyer
Complete Reference: Machine Learning in Stock Price Trend Forecasting[6]
Objective:
Key problem - The amount of interest generated by forecasting the stock price trend started as a starting point for authors to come up with a way of using Machine Learning techniques to forecast the stock price trend.
Reasons for addressing this problem - Machine Learning is popular because of their ability to handle massive amounts of data and make accurate predictions regarding stock price.
Key Objective - The objective of this paper is to apply supervised learning techniques to forecast the stock price trend. The authors based created a trading strategy on the stock price.
How are the authors addressing this problem? The authors realized that the U.S. stock market is a semi-strong efficient market and thus predictive analytics can only be used to gain long-term gains and not short-term. Their initial next day prediction had a accuracy of around 50%. But when they tried to predict the long-term performance their accuracy went up to 79%
Findings and Implications - They concluded SVM model provided highest accuracy of 77% on long-term basis. Their trading strategy gave positive results and outran stock performance.
Approach: The training data in this project was collected from Bloomberg Database and they used 3M stocks for their application. They used two labels viz. "1" and "-1". If the closing price is higher than the previous day then it was labeled as 1 otherwise it was labeled as -1. The PE ratio, PX volume, PX ebdita, S&P 500 index etc. are predictor variables and label would be Y. They came up with two models, for next day prediction and other for long-term prediction.

[6] "Machine Learning in Stock Price Trend Forecasting - CS 229: Machine ...." http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf. Accessed 12 Dec. 2016.



From the above graph in the paper, we can see that the accuracy increases as the time increases especially in the case of SVM and QDA. SVM gave the highest accuracy.
Initially, they used 70% of their data to fit their model. Later, they used the new information and updated their predictor for each trading date.
Result:



From the above graph, we can see that their strategy outran the stock performance with a return of 19.3% v 12.5%.
Summary: Paper gives a good example of how predictive analytics can be used to forecast stock price trend.

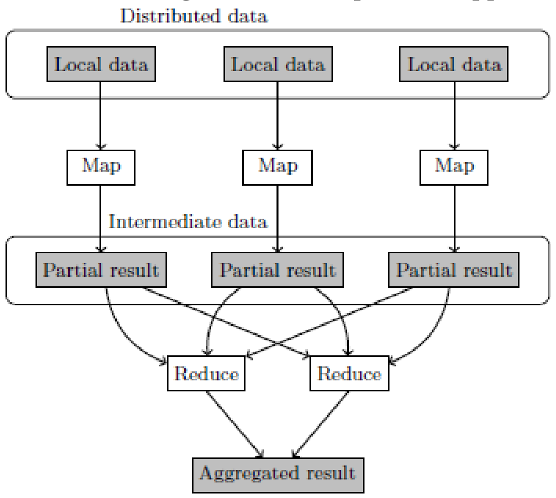
Reviewed by: Vani Pujari
Complete Reference: Text Mining Infrastructure in R[7]
Objective:
Key Problem - Text mining has become very popular over the last 10 years. So, the authors are addressing the need for framework in R for text mining applications.
Reason for addressing the problem - The reason why addressing the problem is important is - without having framework or techniques, R will not be able to handle analysis where text mining is an integral part.
Key Objective - To establish the infrastructure required for text mining in R.
How are the authors addressing this problem? The authors are addressing the problem by researching on packages available in R and checking if they can be used for the functions that are required in text mining.
Findings & Implications - They found tm package to be useful. However, they developed a new framework for step by step process of text mining. R is adept at handling basic data analysis using text mining applications. For advanced text analysis, one can use the various extension packages that are available.
Approach: The research that the authors conducted can be split into four steps. First step was to identify the requirements of a text mining framework conceptually. Second step was to design a generic framework, decide on the data structures and create ways to customize the framework. Third step was to implement processes like preprocessing, data cleaning, whitespace removal, stemming etc. Fourth step was to demonstrate text mining tasks like count based evaluation, text clustering and text classification etc.

[7] "Text Mining Infrastructure in R - Semantic Scholar." 16 Mar. 2008, https://pdfs.semanticscholar.org/37c4/095eb7fc4cfdb6c0218ce260c5125ce1e2ce.pdf. Accessed 12 Dec. 2016.

Results: The authors found that tm package can also be used to apply most of the existing methods to complicated data structures like text. It can be used to interface with other tools that are built specifically for text mining. Thus we can leverage text mining methods from third party in the R environment.
Summary:

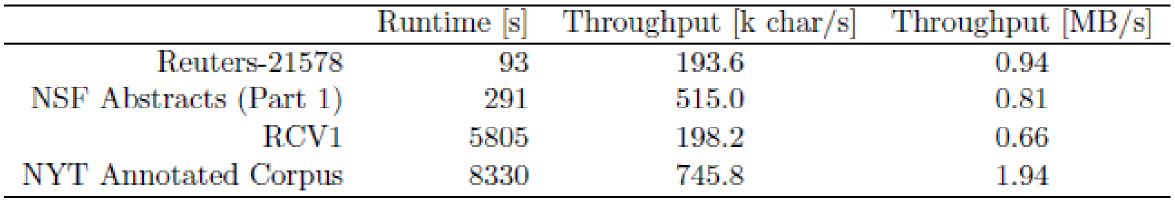
- The authors tried to find packages in R that could be used for text mining. They studied the various packages that have methods that classically ran on structured data structures like data frames and tried to adapt them to handle unstructured data type like text. They also explored open source toolkits like OpenNLP and Weka.
- The tm package is a very useful package for the purpose of text mining. We have used it in our project for creating a corpus, removing stop words, stemming, create a document matrix etc.

Reviewed by: Vani Pujari
Complete reference: A TM PLUG-IN FOR DISTRIBUTED TEXT MINING IN R [8]
Objective: To explore distributed computing for text mining on large data sets.
Key problem – The paper addresses the problem of space complexity when running text mining applications.
Reason for addressing this problem – With the huge amount of data that needs to be processed now-a-days, using a single server or machine is not feasible. The computing capacity is limited by the RAM available on the machine.
Key objective – The key objective is to utilize the power of processing on distributed systems. The paper also aims at leveraging the power of MapReduce and Hadoop along with tm package in R for text mining applications.
How are authors addressing this problem? - The authors explore the tm package in detail. Then they identify the challenges related to using tm package for big data sets and find solutions.
Findings and implications - The data sets that typically consist of billions of documents cannot be processed on a single machine as they cannot fit into main memory like corpora. Hence, parallel computation needs to be used to make it feasible. MapReduce was built for large scale processing of big data. Hadoop file system was built for distributed computing. The authors researched about how tm package can be deployed on such systems and text mining can be performed on distributed systems to reduce computing time.
Approach: The diagram below depicts the approach used by the authors



Results: The performance of parallel computation can be summarized in the table below.

[8] "A tm Plug-In for Distributed Text Mining in R - Journal of Statistical ...." 16 Nov. 2012, https://www.jstatsoft.org/article/view/v051i05/v51i05.pdf. Accessed 12 Dec. 2016.

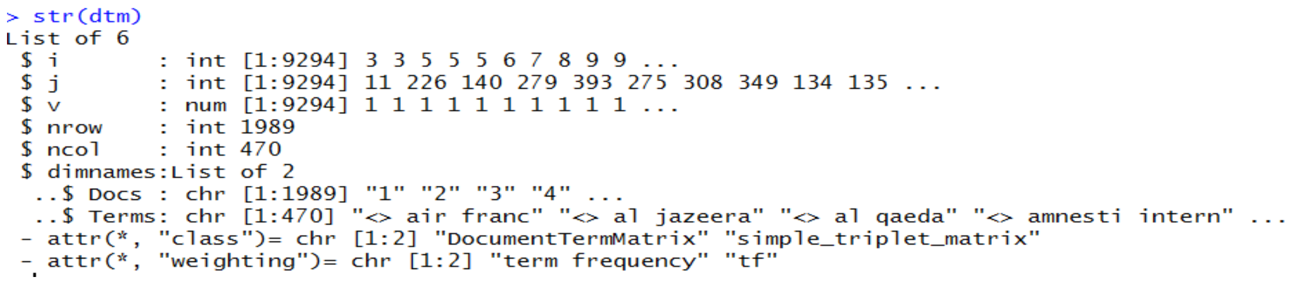


	Runtime [s]	Throughput [k char/s]	Throughput [MB/s]
Reuters-21578	93	193.6	0.94
NSF Abstracts (Part 1)	291	515.0	0.81
RCV1	5805	198.2	0.66
NYT Annotated Corpus	8330	745.8	1.94

Summary:
The paper successfully demonstrated that distributed systems can be used to leverage processing of large data sets. Parallel computation is made possible by using packages like tm.plugin.dc.

4. Dataset description

The dataset has two files with DJIA news and Reddit news each. The Reddit news headlines are compiled by crawling into the news page and the DJIA news is compiled from Yahoo Finance. The main file which we have used for our analysis has all the news compiled for a single day which is named as “Combined_News_DJIA”. The dataset we have at our disposal is completely in text and thus requires preprocessing to convert it into numerical format. We created a document term matrix using this data which is further used for modeling and analysis. An example of dtm is shown below. We have **3066** attributes and **1989** instances for a dtm of trigram.



```
> str(dtm)
List of 6
 $ i       : int [1:9294] 3 3 5 5 5 6 7 8 9 9 ...
 $ j       : int [1:9294] 11 226 140 279 393 275 308 349 134 135 ...
 $ v       : num [1:9294] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 1989
 $ ncol    : int 470
 $ dimnames:List of 2
  ..$ Docs : chr [1:1989] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:470] "<> air franc" "<> al jazeera" "<> al qaeda" "<> amnesti intern" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

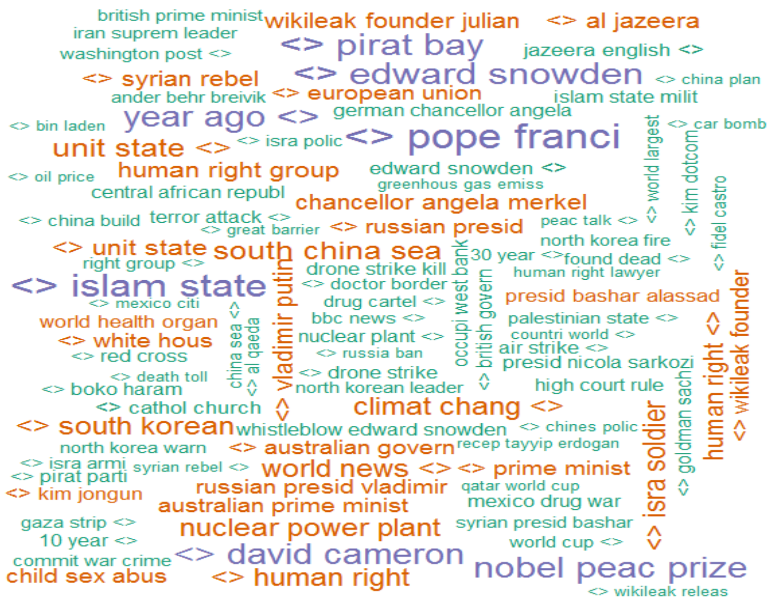


Figure 1Word Cloud

Word Cloud:
To summarize the words and highlight the most important keywords in the text we have a word cloud. The most important keywords are clearly visualized.

5. Project approach

5.1. Code flowchart

This is the basic flow of our code and all the steps have been explained in the later part of the report

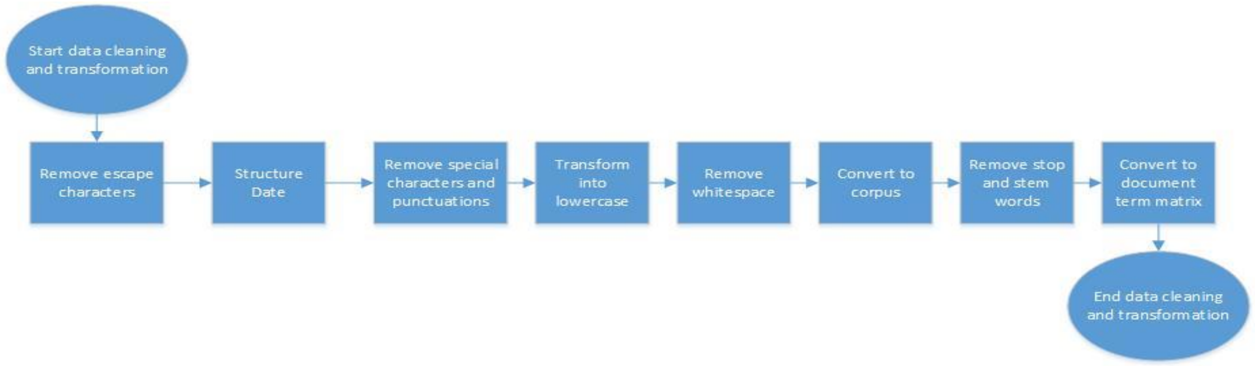

Start data cleaning and transformation
Remove escape characters
Structure Date
Remove special characters and punctuations
Transform into lowercase
Remove whitespace
Convert to corpus
Remove stop and stem words
Convert to document term matrix
End data cleaning and transformation

Figure 2 Data transformation and cleaning

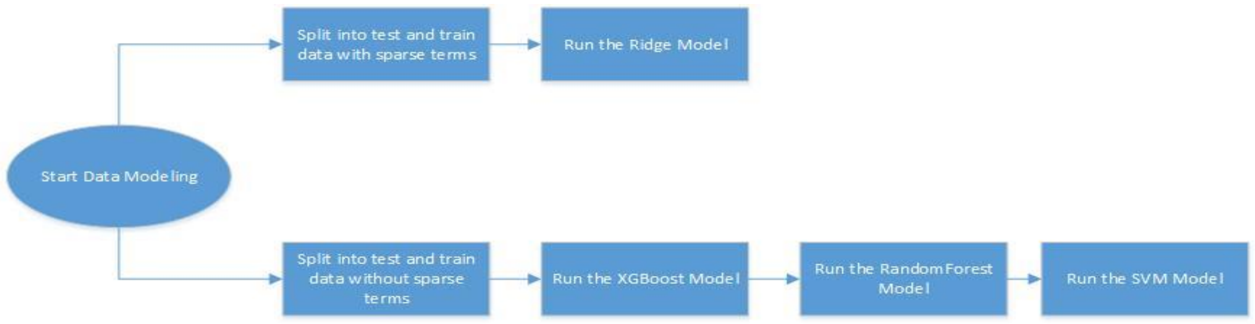

Start Data Modeling
Split into test and train data with sparse terms
Run the Ridge Model
Split into test and train data without sparse terms
Run the XGBoost Model
Run the RandomForest Model
Run the SVM Model

Figure 3 Data Modeling

5.2. Description of new techniques used

5.2.1. Text mining

It is the process of extracting information from textual data. The basis of this analysis is converting the data into numbers. The way we do it is by cleaning the data first and then converting into corpus which then is transformed into a document term matrix (DTM). DTM is basically a matrix where document are rows and terms i.e. words in the document become columns and it stores the frequency of those terms.

5.2.2. Extreme gradient boosting

It is a type of boosted trees that uses the concept of regularization to reduce losses. It also performs pruning at the same time thus encapsulating the advantages of pruning and random forest at the same time. To support this, you will see how XGBoost is better than Random forest in our study below.

6. Text Mining Data Transformation and Cleaning

The initial effort is established in transforming and cleaning the data to create the corpus and data transformation matrix (dtm).

- Date variable has to be reformatted which is used in the later stages for splitting the data into test and train.
  ```
  data$Date <- as.Date(data$Date, "%m/%d/%Y")
  ```

- The data from multiple columns is concatenated into one text blob
  ```
  data$all <- paste(data$Top1, data$Top2, data$Top3, data$Top4, data$Top5, data$Top6, data$Top7,
  data$Top8, data$Top9, data$Top10, data$Top11, data$Top12, data$Top13, data$Top14,
  data$Top15, data$Top16, data$Top17,  data$Top18, data$Top19, data$Top20, data$Top21,
  data$Top22, data$Top23, data$Top24, data$Top25, sep=' <s> ')
  ```

- All the news for ‘n’ number of days has to be compiled into a single text blob. ‘n’- this variable is used to control the number of days of past news. If n=3 then we are taking news from past 3 days, if n=5 then we are taking news from past 5 days and so on.
  ```
  n=1 x=1 y=0
  for (i in 1:nrow(data)){
   if (i<=n) {
    y <- y+1
   }else{
    x <- x+1
    y <- y+1
   }

   z <-""
   for (j in x:y) {
    z = paste(z ,data$all[j])
   }
   data$all_n[i] <- z
  }
  ```

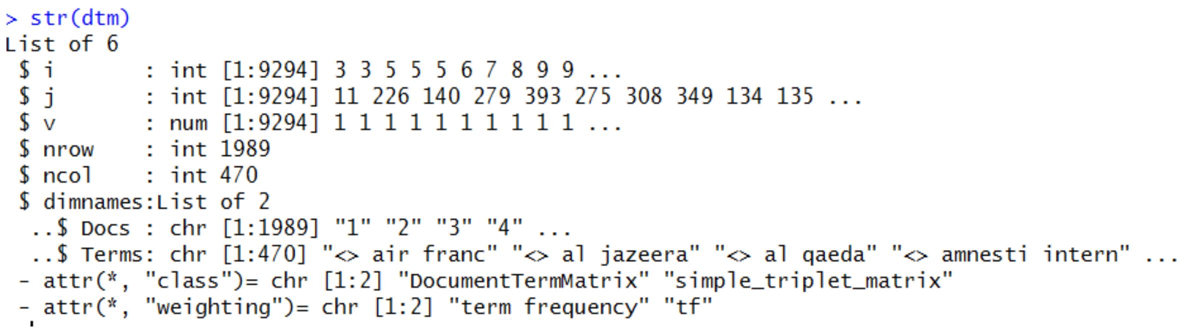
- Irrelevant characters, punctuation, pesky b's and backslashes are removed from the dataset.
  ```
  data$all_n <- gsub('b"|b\'|\\\\', "", data$all_n)
  data$all_n <- gsub("([<>])|[[:punct:]]", "\\1", data$all_n)
  ```

- Dataset has to be converted into lower case
  ```
  data$all_n <- tolower(data$all_n)
  ```

- Numbers are removed as there are no weightage attached to it as far as text mining is concerned.

  ```
  data$all_n <- removeNumbers(data$all_n)
  ```

- Whitespaces are removed
  ```
  data$all_n <- stripWhitespace(data$all_n)
  ```



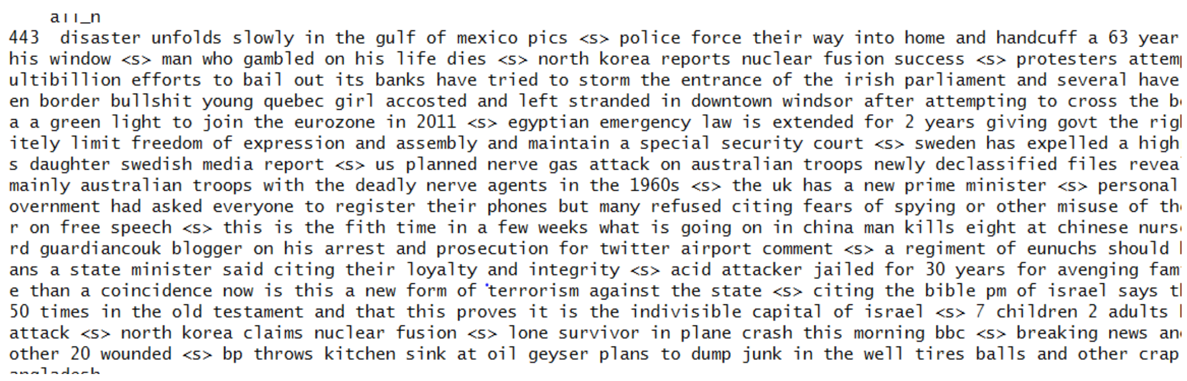
```
> str(dtm)
List of 6
 $ i       : int [1:9294] 3 3 5 5 5 6 7 8 9 9 ...
 $ j       : int [1:9294] 11 226 140 279 393 275 308 349 134 135 ...
 $ v       : num [1:9294] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 1989
 $ ncol    : int 470
 $ dimnames:List of 2
  ..$ Docs : chr [1:1989] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:470] "<> air franc" "<> al jazeera" "<> al qaeda" "<> amnesti intern" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

6.1. Creation of Corpus

This data that has been cleaned has to transformed into a corpus and then a data transformation matrix. Stopwords, whitespace (created by removing stopwords) and stem words are removed from the corpus.

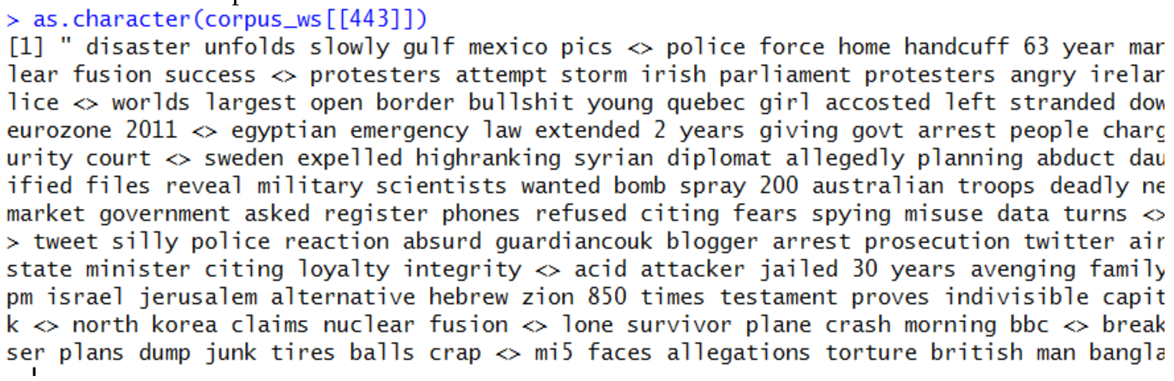
```
corpus<- Corpus(VectorSource(data$all_n))
corpus_1 <- tm_map(corpus, removeWords, stopwords("SMART"))
corpus_2 <- tm_map(corpus_1, removeWords, stopwords("english"))
corpus_ws <- tm_map(corpus_2, stripWhitespace)
corpus_stem <- tm_map(corpus_ws, stemDocument)
```

This is how the data looked before it was reformed into a corpus



```
    all_n
443  disaster unfolds slowly in the gulf of mexico pics <s> police force their way into home and handcuff a 63 year
his window <s> man who gambled on his life dies <s> north korea reports nuclear fusion success <s> protesters attem
ultibillion efforts to bail out its banks have tried to storm the entrance of the irish parliament and several have
en border bullshit young quebec girl accosted and left stranded in downtown windsor after attempting to cross the b
a a green light to join the eurozone in 2011 <s> egyptian emergency law is extended for 2 years giving govt the rig
itely limit freedom of expression and assembly and maintain a special security court <s> sweden has expelled a high
s daughter swedish media report <s> us planned nerve gas attack on australian troops newly declassified files revea
mainly australian troops with the deadly nerve agents in the 1960s <s> the uk has a new prime minister <s> personal
overnment had asked everyone to register their phones but many refused citing fears of spying or other misuse of th
r on free speech <s> this is the fith time in a few weeks what is going on in china man kills eight at chinese nurs
rd guardiancouk blogger on his arrest and prosecution for twitter airport comment <s> a regiment of eunuchs should
ans a state minister said citing their loyalty and integrity <s> acid attacker jailed for 30 years for avenging fam
e than a coincidence now is this a new form of terrorism against the state <s> citing the bible pm of israel says t
50 times in the old testament and that this proves it is the indivisible capital of israel <s> 7 children 2 adults
attack <s> north korea claims nuclear fusion <s> lone survivor in plane crash this morning bbc <s> breaking news an
other 20 wounded <s> bp throws kitchen sink at oil geyser plans to dump junk in the well tires balls and other crap
```

This is how the corpus looks after the transformation



```
> as.character(corpus_ws[[443]])
[1] " disaster unfolds slowly gulf mexico pics <> police force home handcuff 63 year mar
lear fusion success <> protesters attempt storm irish parliament protesters angry irelar
lice <> worlds largest open border bullshit young quebec girl accosted left stranded dow
eurozone 2011 <> egyptian emergency law extended 2 years giving govt arrest people charg
urity court <> sweden expelled highranking syrian diplomat allegedly planning abduct dau
ified files reveal military scientists wanted bomb spray 200 australian troops deadly ne
market government asked register phones refused citing fears spying misuse data turns <>
> tweet silly police reaction absurd guardiancouk blogger arrest prosecution twitter air
state minister citing loyalty integrity <> acid attacker jailed 30 years avenging family
pm israel jerusalem alternative hebrew zion 850 times testament proves indivisible capit
k <> north korea claims nuclear fusion <> lone survivor plane crash morning bbc <> break
ser plans dump junk tires balls crap <> mi5 faces allegations torture british man bangla
```

6.2. Tokenization and Document Term Matrix

For our project we have taken 6 types of n-gram tokenizers: Unigram, Bigram, Trigram, 4Gram and other ngram tokenizers. As an example we have shown here a DTM using a Uni-Bi-Trigram tokenizer. Similarly, dtm for other tokenizers will be created. The value of minimum and maximum in Weka_control changes as per the ngram token required.

```
UniBiTrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 3))
control <- list(
  tokenize=UniBiTrigramTokenizer,
  bounds = list(global = c(10, 500))
)
dtm <- DocumentTermMatrix(corpus_stem, control=control)
```

The label i.e. the market movement is given a lead to make our model useful for future prediction.

```
data$LeadLabel <- lead(data$Label,1,default = as.factor(0))
```

7. Data Modeling

7.1. Splitting data into test and train

- The data is split based on date. So a split index is created as follows
```
split_index <- (data$Date <= '2014-12-31')
```

- The columns with <> are removed as they do not hold any meaning
```
has_tokens <- grepl("<>", colnames(as.matrix(dtm)))
```

- The data is split into train and test using the split index.
```
ytrain <- data$Label[split_index]
xtrain <- as.matrix(dtm)[split_index, !has_tokens]
ytest <- data$Label[!split_index]
xtest <- as.matrix(dtm)[!split_index, !has_tokens]
```

- A model is created. We have considered four models in our project – Ridge model, XGBoost model, RandomForest and SVM model. The ridge model is shown below for example.
```
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)
```

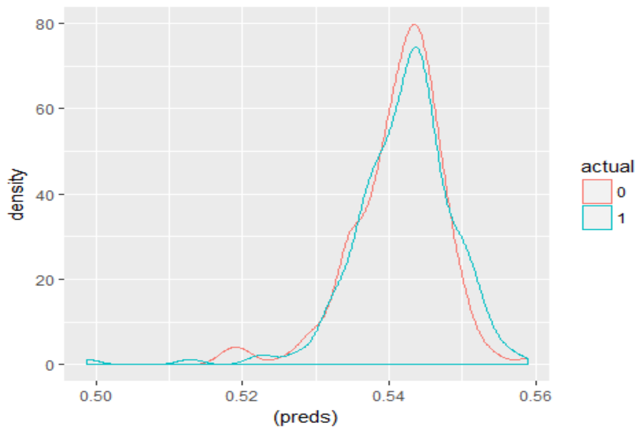
- Predictions are generated
```
Preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")
preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
```

- Confusion matrix is plotted. Accuracy and misclassification rate are calculated.
```
table(preds_y,ytest)

##        ytest
## preds_y   0   1
##       0   0   1
##       1 186 191

#Accuracy
mean(preds_y==ytest)

## [1] 0.505291
```

```
#Misclassification rate
mean(preds_y!=ytest)

## [1] 0.494709
```

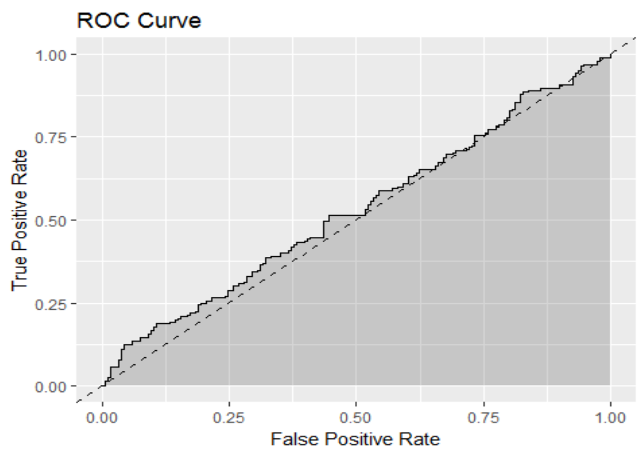
- AUC is calculated and AUC curve is plotted

```
results <- data.frame(pred=(preds), actual=ytest)
ggplot(results, aes(x=(preds), color=actual)) + geom_density()

prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc

## [1] 0.5300179
```



- ROC Curve is plotted

```
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



7.2. Creating the models

As mentioned earlier, we have considered three cases: (1) Predicting the DJIA movement based on previous day news (2) Predicting the DJIA movement based on previous 3 days' news (3) Predicting the DJIA movement based on previous 5 days' news. For each case we tokenized our text into: unigram, bigram, trigram, 4gram and other mixed Ngrams. The four main algorithms we tested were: Extreme gradient boosting (XGBoost), Support Vector Machine (SVM), Random Forest and Ridge regression.

7.2.1. Ridge Regression

Since the number of features is huge, we needed a shrinkage technique and hence, we decided to use ridge regression. It performs L2 regularization. It also helps reduce the collinearity between the huge number of predictors.

```
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)
```

7.2.2. XGBoost Model

"XGBoost" stands for Extreme Gradient Boosting. It is more powerful than the traditional algorithms like Random Forest or Neural Network. It is very fast as it has the ability to perform parallel computation. It can be used for multiple functions like regression, classification and ranking. Moreover, it is a very favored model since there are quite a few parameters using which we can customize the model.

```
xgb_mod <- xgboost(xtrain, label=ytrain, nrounds=100, objective="binary:logistic",
verbose=0)
```

7.2.3. RandomForest

Random Forest was chosen since it is a very good technique for dimensionality reduction. Given the enormous size of the data, it was an obvious choice. In this technique, multiple trees are grown and the best tree is chosen as the final tree. Even here, we can choose the number of trees, size of the tree (number of predictors) etc.

```
rf <- randomForest(xtrain,as.factor(ytrain),ntree = 500, mtry = sqrt(ncol(xtrain)))
```

7.2.4. SVM

SVM is a technique which is built for high dimensionality. It works well where the number of predictors or dimensions is greater than the number of samples. It is also very memory efficient since it uses subset of data for modeling.

```
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
```

8. Data Analysis

We chose four evaluation metrics: (1) AUC of Probability-Density curve (2) Misclassification rate (3) Accuracy (4) ROC. AUC of the Probability-Density curve helps us understand how well our model performs to random guessing of the labels. The proportion of wrongly predicted test data is determined by the misclassification rate. Accuracy determines the proportion of test data that were rightly predicted. ROC curve, plotted for FPR vs TPR, determines how sensitivity varies with specificity for different threshold levels.

9. Results - Evaluation and Discussion

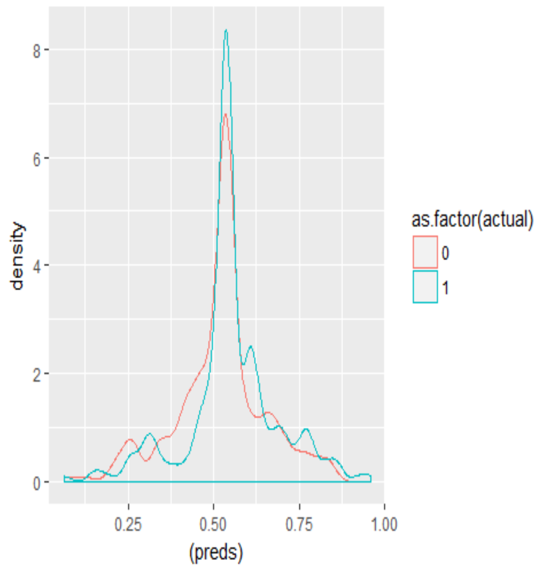
The table shows the performance of 72 models with respect to four evaluation metrics. The highest priority is given to AUC of each model. The models which have AUC above 0.5 means that they are better than random guessing. The values highlighted in red are the ones which are the top performing models and are further scrutinized below to find the best model.

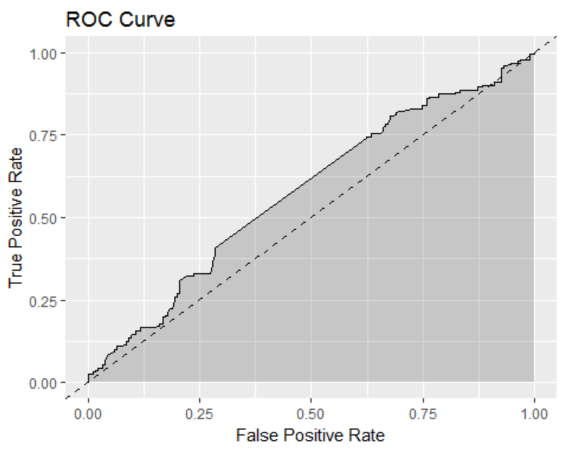
Table 1 Table for AUC (Area under the curve), Accuracy, MR (Misclassification rate)

n	Gram	Ridge			XGBoost			Random Forest			SVM		
		AUC	Accuracy	MR	AUC	Accuracy	MR	AUC	Accuracy	MR	AUC	Accuracy	MR
1	1	0.52554	0.50794	0.49206	0.50818	0.49735	0.50265	0.49110	0.50265	0.49735	0.46223	0.45767	0.54233
	2	0.53002	0.50529	0.49471	0.46547	0.47355	0.52646	0.49881	0.50113	0.49887	0.50493	0.48677	0.51323
	3	0.50000	0.50794	0.49206	0.57625	0.56085	0.43915	0.45662	0.50243	0.49757	0.48108	0.53175	0.46825
	1,2,3	0.56622	0.50794	0.49206	0.47404	0.48413	0.51587	0.51990	0.55340	0.44660	0.54533	0.43386	0.56614
	2,3	0.50000	0.50794	0.49206	0.47187	0.45767	0.54233	0.51009	0.50021	0.49979	0.50823	0.49735	0.50265
	4	0.52575	0.52381	0.47619	0.54314	0.52646	0.47355	0.50111	0.49226	0.50774	0.52034	0.51587	0.48413
3	1	0.50000	0.50794	0.49206	0.50300	0.50265	0.49735	0.51068	0.50794	0.49206	0.50792	0.47884	0.52116
	2	0.50000	0.50794	0.49206	0.53156	0.50265	0.49735	0.49808	0.51109	0.48891	0.48508	0.51587	0.48413
	3	0.50000	0.50794	0.49206	0.45102	0.48148	0.51852	0.49888	0.50400	0.49600	0.51493	0.49206	0.50794
	1,2,3	0.50000	0.50794	0.49206	0.48426	0.50265	0.49735	0.46771	0.49098	0.50902	0.51935	0.49471	0.50529
	2,3	0.50000	0.50794	0.49206	0.51669	0.50000	0.50000	0.49001	0.46230	0.51220	0.48127	0.53175	0.46825
	4	0.50000	0.50794	0.49206	0.48107	0.49735	0.50265	0.51000	0.50112	0.49888	0.50900	0.48677	0.51323
5	1	0.50000	0.50794	0.49206	0.54612	0.52910	0.47090	0.48223	0.47619	0.52381	0.51722	0.50529	0.49471
	2	0.50000	0.50794	0.49206	0.49594	0.49206	0.50794	0.54110	0.48524	0.51476	0.51131	0.48942	0.51058
	3	0.50000	0.50794	0.49206	0.48209	0.47884	0.52116	0.46151	0.49834	0.50166	0.52697	0.46825	0.53175
	1,2,3	0.50000	0.50794	0.49206	0.48331	0.50794	0.49206	0.49999	0.46563	0.53437	0.52299	0.48148	0.51852
	2,3	0.50000	0.50794	0.49206	0.52151	0.52910	0.47090	0.48891	0.50111	0.49889	0.50602	0.49735	0.50265
	4	0.50000	0.50794	0.49206	0.47187	0.45767	0.54233	0.48621	0.51058	0.48942	0.46771	0.50265	0.49735

The fundamental level of analysis is the news of how many days is expected to impact the trend of DJIA. From the table we have summarized the best model for every value of n (i.e. 1,3,5). A quick look of the chart will give us the best model for every algorithm we have modelled and their ROC, confusion matrix and AUC is as follows:

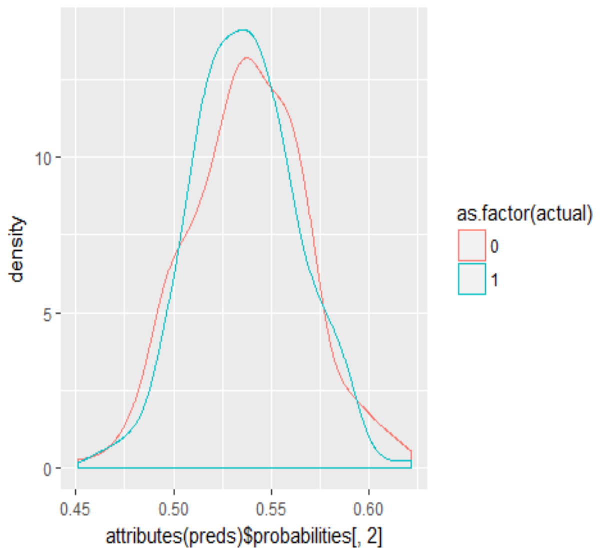
1- n=1, XGboost with Trigram tokenizer

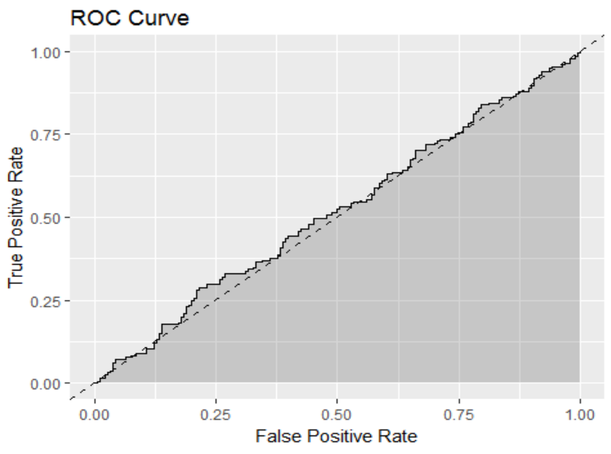




```
table(preds_y,ytest)

##        ytest
## preds_y   0   1
##       0  62  42
##       1 124 150
```

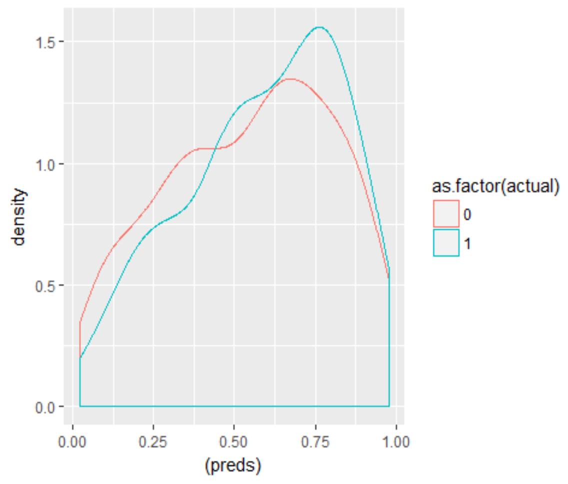
2- n=3, SVM with Ngram tokenizer (2 to 3)

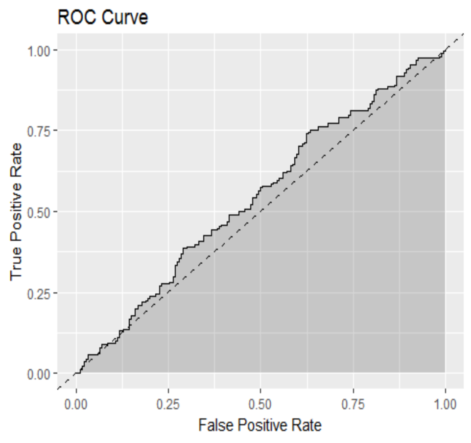




```
#Confusion Matrix
table(preds_y,ytest)

##        ytest
## preds_y   0   1
##       0  79  70
##       1 107 122
```

3- n=5, XGboost with unigram tokenizer





```
#Confusion Matrix
table(preds_y,ytest)

##        ytest
## preds_y   0   1
##       0  77  69
##       1 109 123
```

After analyzing the AUC, ROC, accuracy and misclassification rate we can fairly say that XGboost with Trigram tokenizer and n=1 is the best performing model of all.

10. Conclusions and Future Directions

Our first conclusion is that our model using XGBOOST algorithm which uses Trigram tokenizer with n=1 performs better than the rest of the algorithms. It has the AUC of 0.57625 with accuracy of 0.56085 and misclassification rate of 0.43915. This is because of the models inherent capability to perform efficiently with respect to all other algorithms. Also, as mentioned earlier, this model makes use of parallel computation which makes it much faster and easier to work with.
Also another conclusion of our research is that n=1 works better than different values of n. This means that one-day old news does better job at predicting DJIA's direction rather than 3 day or 5-day old news. One explanation to this could be the self-correcting nature of the stock market. As more and more people get aware of the news, they react to it (either by investing or selling stocks) and bring the market to stability. Also people react to the news instantly and as the news gets old, it tends to have less impact on the market. In future we can also consider including the news body along with the headlines. That will give more context to the headlines and add to the sentiment of the news. Also, instead of using global news we can just focus on the financial news or the news of companies which are part of Dow Jones.

11. Roles and responsibilities

Task	**Responsibilities**
Data Scrutiny	Pratik & Nishant
Initial Data Accumulation	Saurabh & Vani
Data quality check	Nishant
Explorative Analysis	All
Model Selection	All
Cleaning and preparation	All
Render test design	Saurabh & Vani
Build model	All
Interpret model results	All
Rectify and repeat	Pratik & Nishant
Final report preparation	All

12. References

[1] Technical-Analysis, The Trader's Glossary of Technical Terms and Topics. 2005, http://www.traders.com.

13. APPENDIX

13.1. Proposal Sheet (Updated)

Stock Markets play a vital role in the financial world. The movements in the share prices not only impact the economy, but also significantly affect the consumer confidence. There is high volatility in the index of stock market and this fluctuation has a direct correlation with sentiment associated with the financial news. A common hypothesis about the existence of efficient market states that a liquid market index fully reflects all available information. Apart from that there is the inclusion of non-public information about the market like mergers, acquisitions etc. that enforces us to categorizes the market into weak, semi-strong and strong. In general, the traders try to make inferences from the news and available information, predicts or rather makes an intelligent speculation of the stock market and invests accordingly. In short, a positive environment creates a favorable atmosphere for investors and a negative events make consumers more apprehensive about investing.

The data fed to the system will be news stories pulled from various resources and compiled into csv files. We will try to find a correlation between stock prices movement and relevant features in the news stories that affects these changes. Another important aspect of the text mining will be to identify the important features of the news. Appropriate machine learning and text mining techniques will be applied to find correlation between Dow Jones index and the features in the news articles. We can use Naive Bayesian classifier or Support Vector Machines (SVM) to develop the predictive model and train them using training set. Next, we would take new set news as testing data and classify them to “Up” or “Down” trend using the model. Using the results from the model, investors can make a wiser decision while investing and thus improve their chances of making profits.

Possible challenges with R coding

Text mining is a vast field which consists of numerous theoretical approaches and methods. Finding the right approach or method and subsequently finding the libraries in R that can help us achieve text mining can be a challenge.
Implementing data cleansing can be a challenge as dirty data can play havoc. Finding the right package for transformation of the data can be a challenge. If an exact match is not found, we might have to customize functions to achieve transformation.
A large text file typically contains words that have the same root word – increase, increasing, increased etc. Stemming is the method used to extract the root word from such words. Simple stemming algorithms are not very effective. They work by simply eliminating the ends of the words. But this technique will not work for more complicated words. Hence, it can be a challenge to find the right algorithms and equivalent functions that will not result in loss of data accidentally.
A lot of current implementations of natural language processing(NLP) algorithms are in Python. Finding similar packages and implementing the same functionalities in R will require a detailed understanding of NLP related R packages and functions. If functions that achieve the required result are not available, then creating functions to do the intended task can be a time taking and cumbersome process.Apart from the challenges of implementing in R, the challenges of NLP also add to the complexity of the project.

Project Timeline

#	Task		Start	End	Effort
1	Data Preparation	Data Audit	23-Nov	23-Nov	1
		Initial Data Collection	23-Nov	24-Nov	2

		Data Description	24-Nov	24-Nov	1
		Data quality verification	25-Nov	25-Nov	1
		Exploratory Analysis	25-Nov	26-Nov	2
2	Modeling	Selection	27-Nov	27-Nov	1
		Cleaning	27-Nov	27-Nov	1
		Construction	27-Nov	28-Nov	2
		Integration	28-Nov	29-Nov	2
		Formatting	29-Nov	30-Nov	2
3	Evaluation	Generate test design	30-Nov	30-Nov	1
		Build model	01-Dec	01-Dec	1
		Interpret model results	01-Dec	01-Dec	1
		Model evaluation	02-Dec	02-Dec	1
		Refine and repeat	02-Dec	02-Dec	1
4	Final Submission	Assessment of results	03-Dec	03-Dec	1
		Produce and develop a final report	04-Dec	06-Dec	3

13.2. Code

```
#Installing packages
install.packages("rpart", dependencies = TRUE)
install.packages("e1071", dependencies = TRUE)
install.packages("xgboost", dependencies = TRUE)
install.packages("Matrix", dependencies = TRUE)
install.packages("SparseM", dependencies = TRUE)
install.packages("parallel", dependencies = TRUE)
install.packages("RWeka", dependencies = TRUE)
install.packages("magrittr", dependencies = TRUE)
install.packages("Matrix", dependencies = TRUE)
install.packages("glmnet", dependencies = TRUE)
install.packages("ROAuth", dependencies = TRUE)
install.packages("RCurl", dependencies = TRUE)
install.packages("stringr", dependencies = TRUE)
install.packages("slam", dependencies = TRUE)
install.packages("NLP", dependencies = TRUE)
install.packages("tm", dependencies = TRUE)
install.packages("ggplot2", dependencies = TRUE)
install.packages("ggmap", dependencies = TRUE)
install.packages("dplyr", dependencies = TRUE)
install.packages("plyr", dependencies = TRUE)
install.packages("wordcloud", dependencies = TRUE)
install.packages("randomForest", dependencies = TRUE)
install.packages("SnowballC", dependencies = TRUE)
install.packages("ROCR", dependencies = TRUE)

##############################################################################
#########

#Loading libraries
library(rpart)
library(e1071)
library(xgboost)
library(Matrix)
library(SparseM)
library(parallel)
library(RWeka)
library(magrittr)
library(Matrix)
library(glmnet)
library(ROAuth)
library(RCurl)
library(stringr)
library(NLP)
library(tm)
library(ggplot2)
library(ggmap)
library(dplyr)
library(plyr)
library(wordcloud)
library(randomForest)
library(SnowballC)
library(ROCR)

##########################################################################
#########

#Set the path of input file
setwd("C:/Users/Jumper/Documents/Jumper/Study/SEM3/ISEN 613/Project")
#Verify if the input file is present
list.files()
#Before loading, remove all escape characters from the excel- \a, \b, \f, \n, \r, \t, \v

#Read in the data
data <- read.csv("Combined_News_DJIA1.csv", stringsAsFactors = FALSE)
#head(data)

#Convert 'Date' column into Date object to make train/test splitting easier
data$Date <- as.Date(data$Date, "%m/%d/%Y")

#Concatenate all columns into one
data$all <- paste(data$Top1, data$Top2, data$Top3, data$Top4, data$Top5, data$Top6,
data$Top7, data$Top8, data$Top9, data$Top10, data$Top11, data$Top12, data$Top13,
data$Top14, data$Top15, data$Top16, data$Top17,   data$Top18, data$Top19,
data$Top20, data$Top21, data$Top22, data$Top23, data$Top24, data$Top25, sep=' <s> ')

##########################################################################
#########

#The value of n measures how many days back in past you want to read the news.
n=0
#n days loop begins
for (z in 1:3) {
  if (z==1) {
    n=1
  }else if (z==2) {
    n=3
  }else {
    n=5
  }

  #Combine n days news into one.
  print(n)
  x=1
  y=0

  for (i in 1:nrow(data)){
    if (i<=n) {
      y <- y+1
    }else {
      x <- x+1
      y <- y+1
  }

  z <-""
  for (j in x:y) {
    z = paste(z ,data$all[j])
  }
  data$all_n[i] <- z
}

#Introduce lead in the label for reflecting n day later index movement
data$LeadLabel <- lead(data$Label,1,default = as.factor(0))

#write.csv(data, file = "C:/Users/Nishant2014sinha/Downloads/stocknews (1)/xyz6.csv")

#Data cleaning and transformation
#Get rid of b's and backslashes
data$all_n <- gsub('b"|b\'|\\\\', "", data$all_n)

#Get rid of all punctuation except headline separators
data$all_n <- gsub("([<>])|[[:punct:]]", "\\1", data$all_n)
#data$all_n[443]

#Convert to lower case
data$all_n <- tolower(data$all_n)

#Remove numbers
#data$all_n <- removeNumbers(data$all_n)

#Remove whitespace
data$all_n <- stripWhitespace(data$all_n)

#write.csv(data, file = "xyz61.csv")

#Convert columns into corpus to take advantage of textmining package
corpus<- Corpus(VectorSource(data$all_n))
corpus[[443]]
as.character(corpus[[443]])

#Remove SMART & English stopwords
corpus_1 <- tm_map(corpus, removeWords, stopwords("SMART"))
#corpus_1[[443]]
#as.character(corpus_1[[443]])

corpus_2 <- tm_map(corpus_1, removeWords, stopwords("english"))
#corpus_2[[443]]
#as.character(corpus_2[[443]])

#Get rid of extra spaces
corpus_ws <- tm_map(corpus_2, stripWhitespace)
#corpus_ws[[443]]
#as.character(corpus_ws[[443]])
```

```
  #Convert all word forms to base words
  corpus_stem <- tm_map(corpus_ws, stemDocument)
  #corpus_stem[[443]]
  #as.character(corpus_stem[[443]])

  #dtm- Document term matrix
  #For Ridge-Sparse the dtm(sparse=1), for XGBOOST, Random forest and SVM-Do not
sparse the dtm(sparse=2)
  for (sparse in 1:2) {
    #Select one out of six type of Ngrams
    for (gram in 1:6) {
      dtm <- switch(
        gram,
        {DocumentTermMatrix(corpus_stem)},
        {BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max =
2))
        control <- list(
          tokenize=BigramTokenizer,
          bounds = list(global = c(20, 500))
        )
        DocumentTermMatrix(corpus_stem,control=control)},
        {TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max =
3))
        control <- list(
          tokenize=TrigramTokenizer,
          bounds = list(global = c(10, 500))
        )
        DocumentTermMatrix(corpus_stem, control=control)},
        {UniBiTrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1,
max = 3))
        control <- list(
          tokenize=UniBiTrigramTokenizer,
          bounds = list(global = c(10, 500))
        )
        DocumentTermMatrix(corpus_stem, control=control)},
        {BiTrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2,
max = 3))
        control <- list(
          tokenize=BiTrigramTokenizer,
          bounds = list(global = c(15, 500))
        )
        DocumentTermMatrix(corpus_stem,control=control)},
        {FourgramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 4, max
= 4))
        control <- list(
          tokenize=FourgramTokenizer,
          bounds = list(global = c(5, 50))
        )
        DocumentTermMatrix(corpus_stem, control=control)}
      )

      #print(dtm)
```

```
if (sparse==1) {
  #Creating train and test using sparse matrix
  split_index <- (data$Date <= '2014-12-31')
  #Remove columns with <>
  has_tokens <- grepl("<>", colnames(as.matrix(dtm)))
  ytrain <- as.factor(data$LeadLabel[split_index])
  xtrain <- Matrix(as.matrix(dtm)[split_index,!has_tokens], sparse=TRUE)
  ytest <- as.factor(data$LeadLabel[!split_index])
  xtest <- Matrix(as.matrix(dtm)[!split_index,!has_tokens], sparse=TRUE)
  #Ridge Model begins
  #Train the model
  set.seed(50)
  glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)
  #Generate predictions
  preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")
  preds_y <- rep(0,length(ytest))
  preds_y[preds>0.5] <- 1
  #Confusion Matrix
  table(preds_y,ytest)
  #Accuracy
  mean(preds_y==ytest)
  #Misclassification rate
  mean(preds_y!=ytest)
  # Put results into dataframe for plotting.
  results <- data.frame(pred=(preds), actual=ytest)
  ggplot(results, aes(x=(preds), color=actual)) + geom_density()
  prediction <- prediction((preds), ytest)
  perf <- performance(prediction, measure = "tpr", x.measure = "fpr")
  auc <- performance(prediction, measure = "auc")
  auc <- auc@y.values[[1]]
  auc
  roc.data <- data.frame(fpr=unlist(perf@x.values),
                         tpr=unlist(perf@y.values))
  ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
  #Ridge Model ends
  #Sparse matrix data split ends
}else {
  #Creating train and test without sparse matrix
  split_index <- (data$Date <= '2014-12-31')
  #Remove columns with <>
  has_tokens <- grepl("<>", colnames(as.matrix(dtm)))
  ytrain <- data$Label[split_index]
  xtrain <- as.matrix(dtm)[split_index, !has_tokens]
  ytest <- data$Label[!split_index]
  xtest <- as.matrix(dtm)[!split_index, !has_tokens]
  #All the three models begin
```

```
for (np in 1:3) {
  if (np==1) {
    #XGBOOST Model begins
    xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)
    preds <- predict(xgb_mod, xtest)
    preds_y <- rep(0,length(ytest))
    preds_y[preds>0.5] <- 1
    #Confusion Matrix
    table(preds_y,ytest)
    #Accuracy
    mean(preds_y==ytest)
    #Misclassification rate
    mean(preds_y!=ytest)
    # Put results into dataframe for plotting.
    results <- data.frame(pred=(preds), actual=ytest)
    # Plot dual densities
    ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
    prediction <- prediction((preds), ytest)
    perf <- performance(prediction, measure = "tpr", x.measure = "fpr")
    auc <- performance(prediction, measure = "auc")
    auc <- auc@y.values[[1]]
    auc
    roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))
    ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
      geom_ribbon(alpha=0.2) +
      geom_line(aes(y=tpr)) +
      geom_abline(slope=1, intercept=0, linetype='dashed') +
      ggtitle("ROC Curve") +
      ylab('True Positive Rate') +
      xlab('False Positive Rate')
    #XGBOOST Model ends
  } else if(np==2) {
    #Random Forest Model begins
    set.seed(25)
    rf <- randomForest(xtrain,as.factor(ytrain),ntree = 500, mtry = sqrt(ncol(xtrain)))
    preds <- predict(rf, newdata = xtest, type = "prob")
    preds_y <- predict(rf, newdata = xtest)
    #Confusion Matrix
    table(preds_y,ytest)
    #Accuracy
    mean(preds_y==ytest)
    #Misclassification rate
    mean(preds_y!=ytest)
    # Put results into dataframe for plotting.
    results <- data.frame(pred=preds[,2], actual=ytest)
    # Plot dual densities
    ggplot(results, aes(x=preds[,2], color=as.factor(actual))) + geom_density()
```

```
      prediction <- prediction(preds[,2], ytest)
      perf <- performance(prediction, measure = "tpr", x.measure = "fpr")
      auc <- performance(prediction, measure = "auc")
      auc <- auc@y.values[[1]]
      auc
      roc.data <- data.frame(fpr=unlist(perf@x.values),
                             tpr=unlist(perf@y.values))
      ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
        geom_ribbon(alpha=0.2) +
        geom_line(aes(y=tpr)) +
        geom_abline(slope=1, intercept=0, linetype='dashed') +
        ggtitle("ROC Curve") +
        ylab('True Positive Rate') +
        xlab('False Positive Rate')
      #Random Forest Model ends
     } else {
      #SVM Model begins
      svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
      preds <- predict(svm_model, xtest, probability = T)
      preds_y <- predict(svm_model, xtest, probability = F)
      #Confusion Matrix
      table(preds_y,ytest)
      #Accuracy
      mean(preds_y==ytest)
      #Misclassification rate
      mean(preds_y!=ytest)
      # Put results into dataframe for plotting.
      results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)
      # Plot dual densities
      ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
      prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
      perf <- performance(prediction, measure = "tpr", x.measure = "fpr")
      auc <- performance(prediction, measure = "auc")
      auc <- auc@y.values[[1]]
      auc
      roc.data <- data.frame(fpr=unlist(perf@x.values),
                             tpr=unlist(perf@y.values))

      ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
        geom_ribbon(alpha=0.2) +
        geom_line(aes(y=tpr)) +
        geom_abline(slope=1, intercept=0, linetype='dashed') +
        ggtitle("ROC Curve") +
        ylab('True Positive Rate') +
        xlab('False Positive Rate')
      #SVM Model ends
     }
     #All the three models end
    }
    #Non Sparse matrix data split ends
   }
```

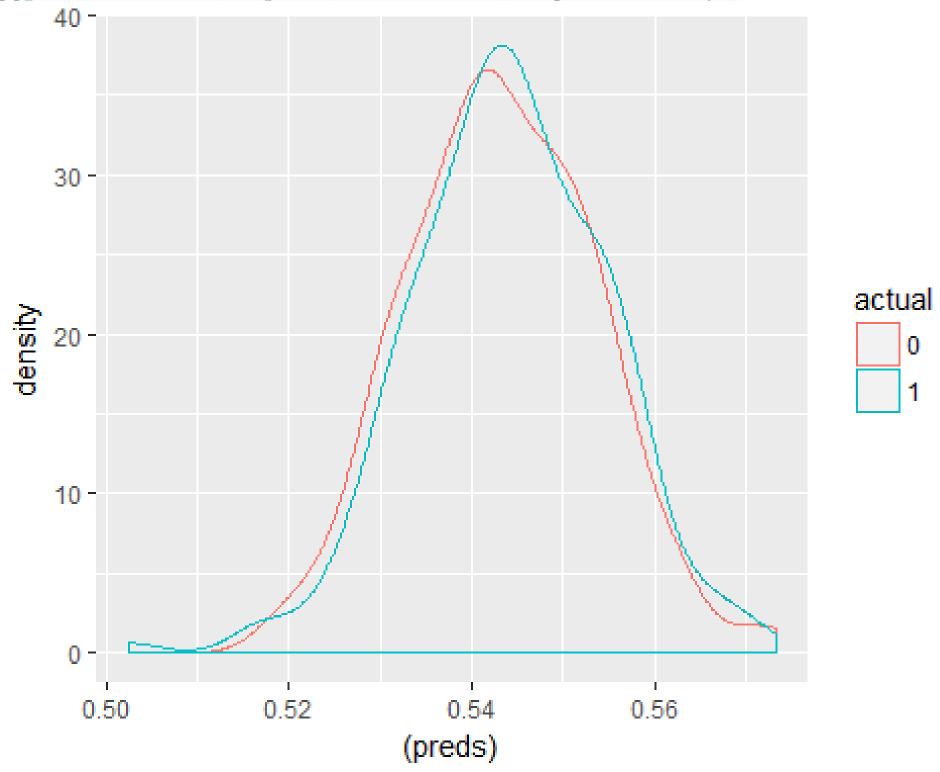
```
      #N gram loop ends
     }
    #Sparse loop ends
   }
  #n days loop ends
 }
```

13.3. Results

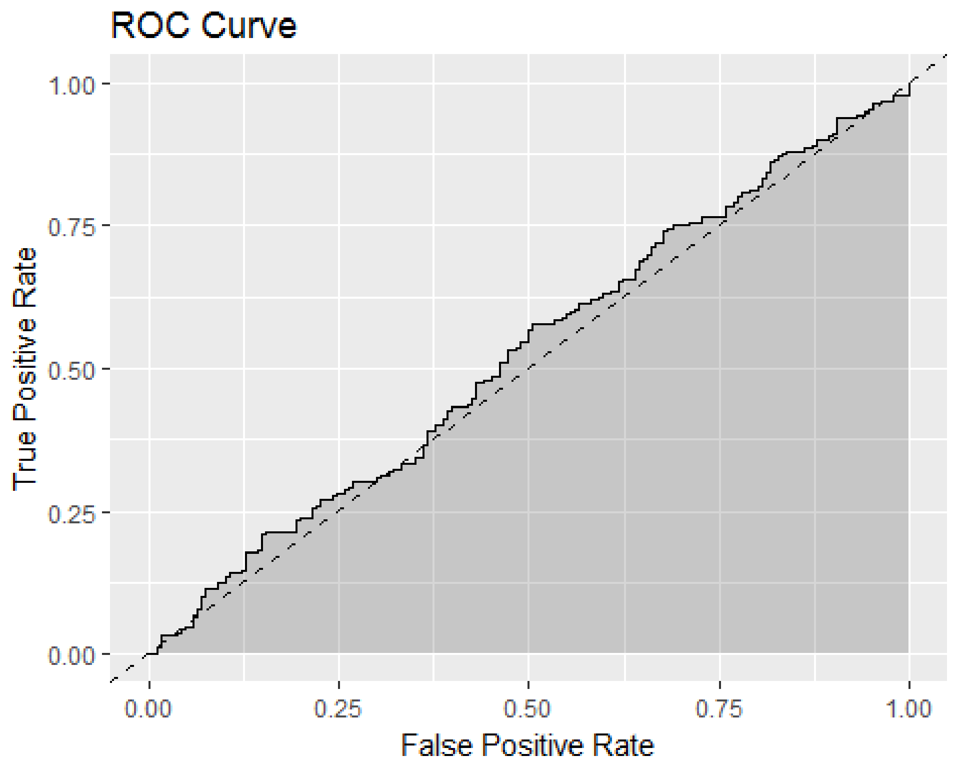
N1G1

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5255376
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

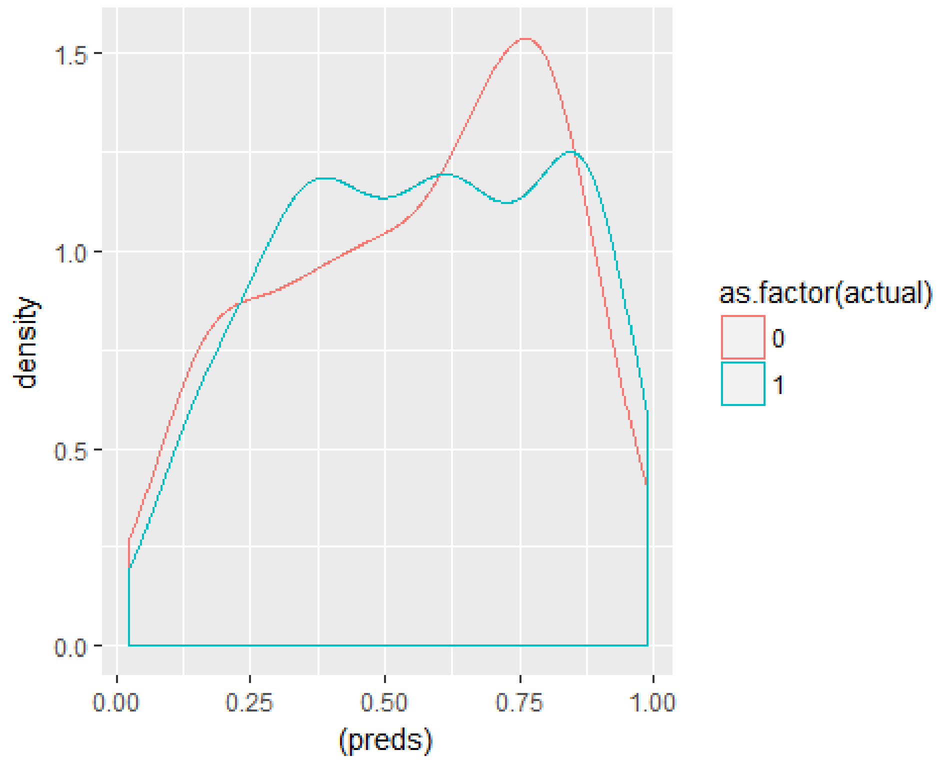


```
#Creating train and test
split_index <- (data$Date <= '2014-12-31')
#Remove columns with <>
has_tokens <- grepl("<>", colnames(as.matrix(dtm)))

ytrain <- data$Label[split_index]
xtrain <- as.matrix(dtm)[split_index, !has_tokens]

ytest <- data$Label[!split_index]
xtest <- as.matrix(dtm)[!split_index, !has_tokens]
```

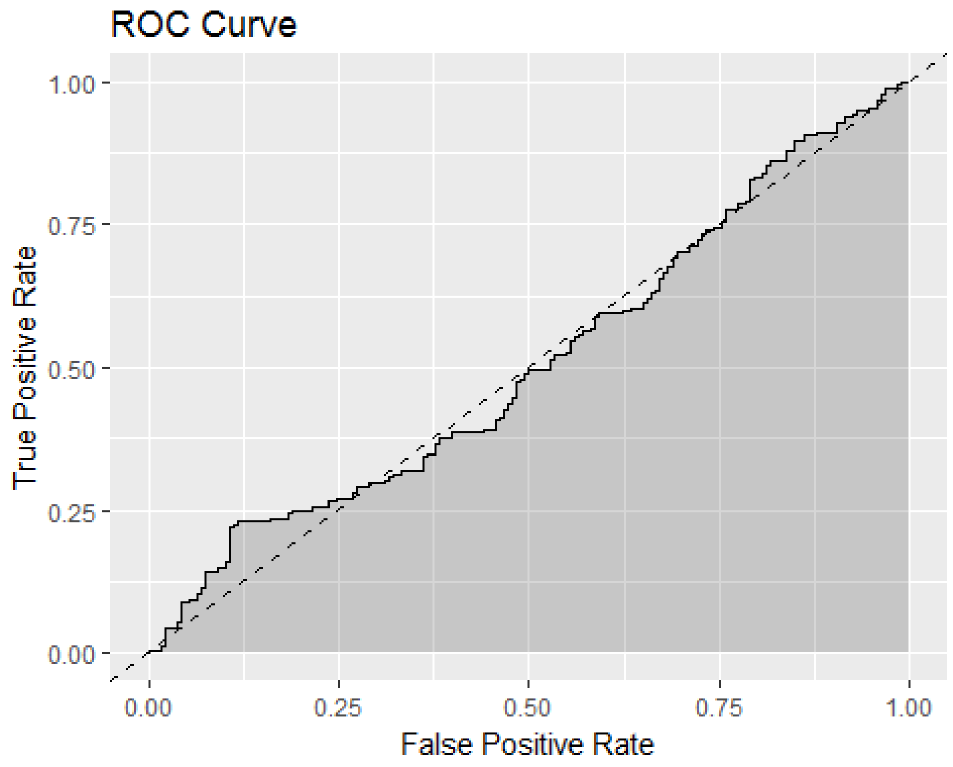
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
```

```
xgb_mod <- xgboost(xtrain,
              label=ytrain,
              nrounds=100,
              objective="binary:logistic",
              verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  77  81
##       1 109 111
#Accuracy
mean(preds_y==ytest)
## [1] 0.4973545
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5026455
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

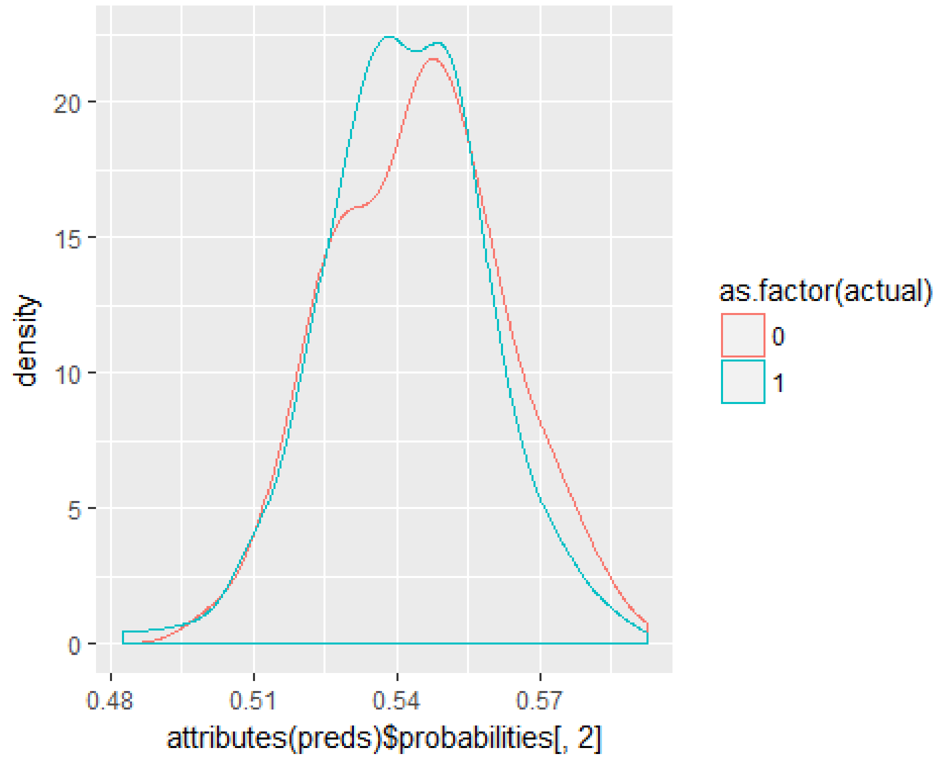


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5081765
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

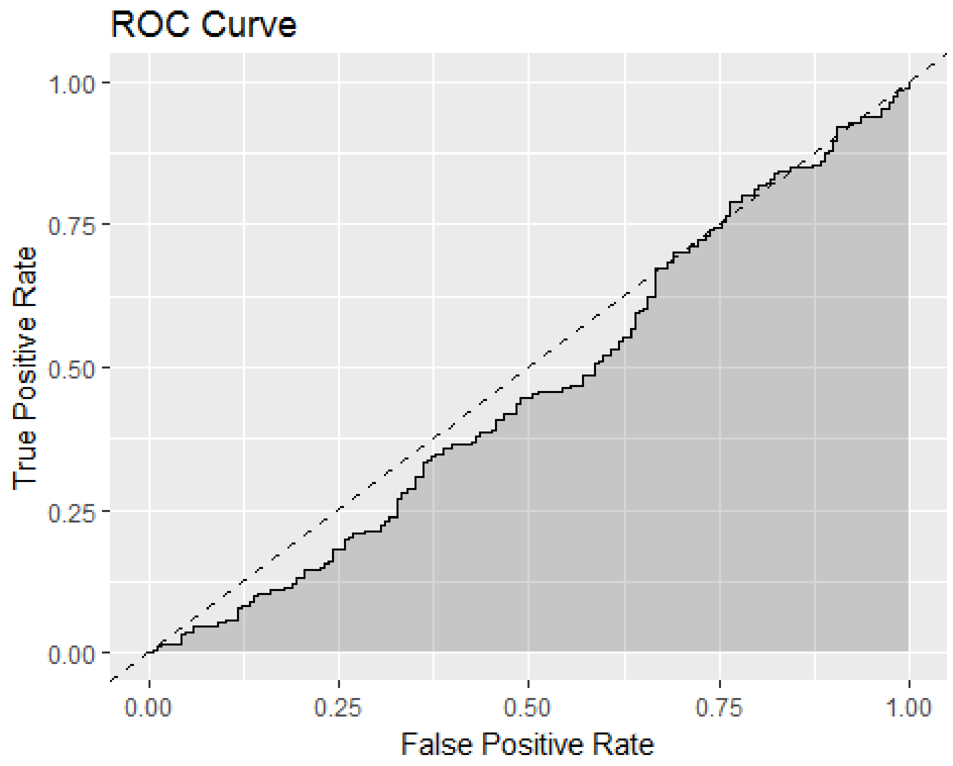


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'X001' and 'X007' and 'X0077' and 'X014' and 'X060'
## and 'X068as' and 'X100000th' and 'X1000km' and 'X1000x' and 'X100200' and
## 'X1006' and 'X100abarrel' and 'X100mstretch' and 'X100mw' and 'X1017' and
## 'X1030' and 'X10300' and 'X1050' and 'X1054' and 'X105carat' and 'X1061c'
## and 'X107000' and 'X108f' and 'X10barrel' and 'X10day' and 'X10fold' and
## 'X10foot' and 'X10side' and 'X10thcenturi' and 'X1100room' and 'X110m' and
## 'X1117' and 'X112000' and 'X1125' and 'X115000' and 'X11571' and 'X115m'
## and 'X1175' and 'X117k' and 'X117yearold' and 'X11c' and 'X11ftlong' and
## 'X120000km2' and 'X1200yrold' and 'X120ft' and 'X120metrelong' and 'X1215'
## and 'X1238' and 'X12400' and 'X12460' and 'X124yearold' and 'X127bn' and
## 'X127m' and 'X1280' and 'X12gigawatt' and 'X12million' and 'X12nation'
## and 'X12tn' and 'X1300yrold' and 'X131' and 'X1310' and 'X132' and
## 'X134yearold' and 'X136' and 'X1362' and 'X1370' and 'X13month' and 'X1433'
## and 'X1440' and 'X14811425' and 'X14b' and 'X14gb' and 'X14tn' and 'X14yr'
## and 'X1500mile' and 'X1500yrold' and 'X150kwh' and 'X151' and 'X1536'
## and 'X15431292' and 'X1566' and 'X15b' and 'X15cm' and 'X15thcenturi' and
## 'X15yo' and 'X1600022000' and 'X1600yearold' and 'X1611' and 'X161774' and
## 'X1621' and 'X1652' and 'X1667' and 'X166th' and 'X16brl' and 'X16member'
## and 'X16page' and 'X16thcenturi' and 'X17000ft' and 'X17000km' and 'X170bn'
## and 'X1729hour' and 'X173' and 'X173000' and 'X1752' and 'X176ounc' and
## 'X1770' and 'X1785' and 'X17hour' and 'X17th20th' and 'X183' and 'X1843'
## and 'X1845' and 'X18600' and 'X187bn' and 'X187m' and 'X1881' and 'X1898'
## and 'X18b' and 'X18cmlong' and 'X18m' and 'X18monthold' and 'X1901' and
## 'X1906' and 'X1912' and 'X19151917' and 'X19171991' and 'X195000' and
## 'X1951' and 'X196000' and 'X19701992' and 'X19791981' and 'X199798' and
## 'X1998are' and 'X19mile' and 'X200000rmb' and 'X20052007' and 'X20082013'
## and 'X20092013' and 'X200million' and 'X201015' and 'X2015106' and 'X2026'
## and 'X204bn' and 'X206' and 'X208' and 'X2089' and 'X20m14m' and 'X20metr'
## and 'X20metrehigh' and 'X20thcenturi' and 'X2110' and 'X212' and 'X213' and
## 'X218m' and 'X2200strong' and 'X2226' and 'X222m' and 'X2234' and 'X224'
## and 'X225bn' and 'X22i' and 'X22year' and 'X2300km' and 'X2300plus' and
## 'X23420' and 'X239400' and 'X23year' and 'X241' and 'X2411' and 'X2428'
## and 'X2500yearold' and 'X250mworth' and 'X250yearold' and 'X2550' and
## 'X25metr' and 'X25mph' and 'X25year' and 'X260000' and 'X2600yrold' and
## 'X262' and 'X26219' and 'X2628' and 'X265000' and 'X267000' and 'X2686'
## and 'X268k' and 'X269000' and 'X26year' and 'X270000' and 'X2700year' and
## 'X279bn' and 'X27th' and 'X285' and 'X286' and 'X28bn' and 'X28millionyear'
## and 'X294bn' and 'X29billion' and 'X29year' and 'X2day' and 'X2km' and
## 'X2seater' and 'X2tn' and 'X2way' and 'X2yr' and 'X3000yr' and 'X300foot'
## and 'X300kwh' and 'X303bn' and 'X3040' and 'X305' and 'X30500' and
## 'X308' and 'X30barrel' and 'X30bn' and 'X31555' and 'X316' and 'X318000'
## and 'X31m' and 'X31yearold' and 'X3218' and 'X32570' and 'X329' and
## 'X330ft' and 'X331kg' and 'X3360' and 'X337' and 'X338' and 'X343' and
## 'X34million' and 'X3500yearold' and 'X3500yrold' and 'X350500' and 'X3508'
## and 'X350m' and 'X350yearold' and 'X3512' and 'X356' and 'X35hour' and
## 'X368' and 'X3681' and 'X37000yrold' and 'X3700mile' and 'X374mph' and
## 'X37703' and 'X37billion' and 'X37metrelong' and 'X381' and 'X3827'
## and 'X3869' and 'X389' and 'X3month' and 'X400000yearold' and 'X4000kg'
## and 'X4000strong' and 'X4060' and 'X413' and 'X42195' and 'X4227' and
## 'X42783' and 'X42850' and 'X42yearold' and 'X4327' and 'X438b' and 'X43m'
## and 'X4400' and 'X4400yearold' and 'X442' and 'X450m' and 'X452' and
## 'X45b' and 'X45billionyearold' and 'X46000' and 'X464' and 'X470m' and
## 'X479m' and 'X480' and 'X481' and 'X483' and 'X485' and 'X4859barrel'
## and 'X49000' and 'X490m' and 'X494' and 'X498' and 'X499' and 'X49th' and
## 'X4andahalf' and 'X5000000' and 'X50000km2' and 'X500billion' and 'X500mn'
## and 'X500yearold' and 'X504' and 'X50700' and 'X508567' and 'X509' and
## 'X50footlong' and 'X50k' and 'X50yearold' and 'X51m' and 'X520' and 'X5248'
## and 'X5300yearold' and 'X533' and 'X547' and 'X550km' and 'X553' and
## 'X554' and 'X560000' and 'X5680' and 'X58000' and 'X5909' and 'X59foot'
## and 'X5hour' and 'X6000ft' and 'X6000km' and 'X600kmh' and 'X600mile' and
## 'X600strong' and 'X603kmh' and 'X60bln' and 'X60k' and 'X610' and 'X61st'
## and 'X620000' and 'X621' and 'X625' and 'X63f' and 'X6400' and 'X640000'
## and 'X641000' and 'X647250' and 'X65c' and 'X670' and 'X670000' and
## 'X67pchuryumovgerasimenko' and 'X681m' and 'X68magnitud' and 'X69000' and
## 'X6900plant' and 'X694' and 'X6bn7bn' and 'X7000ton' and 'X700kayear' and
## 'X7100' and 'X717' and 'X71km' and 'X71yearold' and 'X7251120' and 'X730'
## and 'X749' and 'X74bn' and 'X74pound' and 'X750million' and 'X75cent'
## and 'X75km' and 'X75year' and 'X7600' and 'X76000' and 'X761' and 'X763'
## and 'X769' and 'X769lbs' and 'X76m' and 'X770000' and 'X7800' and 'X7845'
## and 'X785kg' and 'X79magnitud' and 'X79th' and 'X7eleven' and 'X7month'
## and 'X80000120000' and 'X800bn' and 'X804' and 'X808' and 'X813' and
## 'X81m' and 'X830' and 'X83k' and 'X83rd' and 'X8500' and 'X850000' and
## 'X860' and 'X863' and 'X8700' and 'X870m' and 'X872' and 'X872billion'
## and 'X8cm' and 'X8pm' and 'X8thcenturi' and 'X900021000' and 'X907'
## and 'X9115' and 'X916' and 'X91st' and 'X9200' and 'X927' and 'X92bn'
## and 'X933' and 'X93yo' and 'X940' and 'X946' and 'X9525' and 'X958' and
## 'X960000' and 'X962year' and 'X973' and 'X989' and 'X9899' and 'X991' and
## 'X997' and 'X9997' and 'X99millionyrold' and 'X9squaremet' and 'a10' and
## 'a976' and 'aab' and 'abat' and 'abbass' and 'abdelfattah' and 'abdeslam'
## and 'abou' and 'abrini' and 'abseil' and 'abstin' and 'abstract' and
## 'acceleratestemperatur' and 'accrington' and 'acidif' and 'acivist' and
## 'acquaint' and 'adblock' and 'adriat' and 'adwar' and 'aegean' and 'afd'
## and 'affection' and 'afri' and 'afrikaan' and 'afrileak' and 'agencybond'
## and 'agerel' and 'agerestrict' and 'aggres' and 'agua' and 'ah1z' and
## 'aidsinfect' and 'aidth' and 'aiib' and 'aircraftat' and 'airdef' and
## 'airless' and 'aksum' and 'ala' and 'alabadi' and 'alain' and 'alaksa'
## and 'alaraqib' and 'alaskaa' and 'alazhar' and 'albanna' and 'albin'
## and 'alchin' and 'alcoholmetabol' and 'alcoholrel' and 'aldehydestabil'
## and 'aldi' and 'aldouri' and 'alevel' and 'alexandrov' and 'alfais' and
## 'alfalfa' and 'algal' and 'algorithm' and 'alhodeidah' and 'alhusseini'
## and 'alienhunt' and 'alienlik' and 'alledg' and 'allergi' and 'allmal'
## and 'almodovar' and 'alnimr' and 'aloft' and 'alphago' and 'alraqqa'
## and 'alsisi' and 'alsissi' and 'alvarez' and 'alyn' and 'alzawahiri' and
## 'alzor' and 'amber' and 'ambien' and 'amenhotep' and 'americanisra' and
## 'americanseuropean' and 'americantrain' and 'amitrol' and 'amjad' and
## 'ampl' and 'amran' and 'amun' and 'amyloid' and 'analyt' and 'ananta' and
## 'ananya' and 'anbar' and 'ancestoror' and 'anchovi' and 'andincomplet' and
## 'andreessen' and 'angelesbas' and 'angkor' and 'anglia' and 'anheuserbusch'
## and 'animalclon' and 'animalfriend' and 'anisa' and 'annot' and 'anomal'
## and 'anomali' and 'anonsec' and 'anorex' and 'anot' and 'antalya'
## and 'antelop' and 'anthropogen' and 'antiair' and 'antiassimil' and
## 'antibioticfre' and 'anticharli' and 'anticircumvent' and 'antidepress' and
## 'antidiabet' and 'antidiscriminatori' and 'antiencrypt' and 'antiepilepsi'
## and 'antierdogan' and 'antiestablish' and 'antieu' and 'antiforeign' and
## 'antigmo' and 'antih' and 'antikythera' and 'antimigr' and 'antimonarch'
## and 'antiradicalis' and 'antireclaim' and 'antirefuge' and 'antisatellit'
## and 'antittip' and 'antiturkish' and 'antiub' and 'antivacc' and 'aoif' and
## 'apj' and 'apollostyl' and 'apologist' and 'arabi' and 'arabica' and 'arak'
## and 'aramco' and 'arar' and 'arbitrarili' and 'arcad' and 'archaelogist'
## and 'arduous' and 'arepa' and 'aristegui' and 'aristotl' and 'armata' and
## 'army2015' and 'arn' and 'arrrest' and 'ars' and 'arthriti' and 'artwash'
## and 'asana' and 'asean' and 'asgard' and 'ashley' and 'ashram' and 'aslyum'
## and 'aso' and 'asparagus' and 'asphalt' and 'assaad' and 'assn' and
## 'astrophys' and '
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  71  90
##       1 115 102
#Accuracy
mean(preds_y==ytest)
## [1] 0.457672
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.542328
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



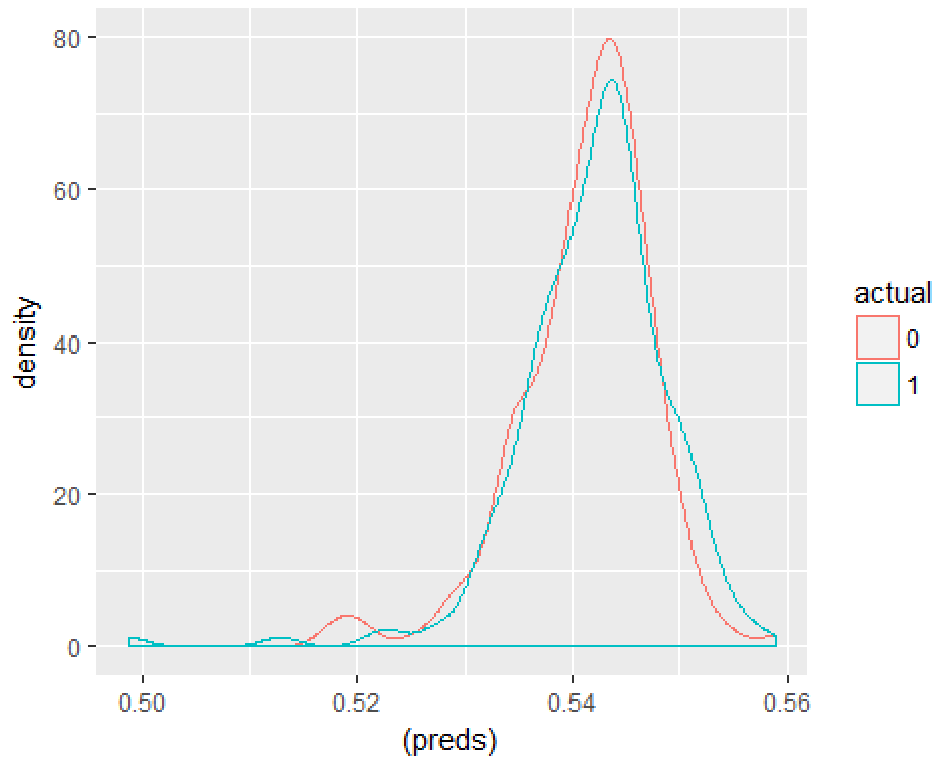
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4622256
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



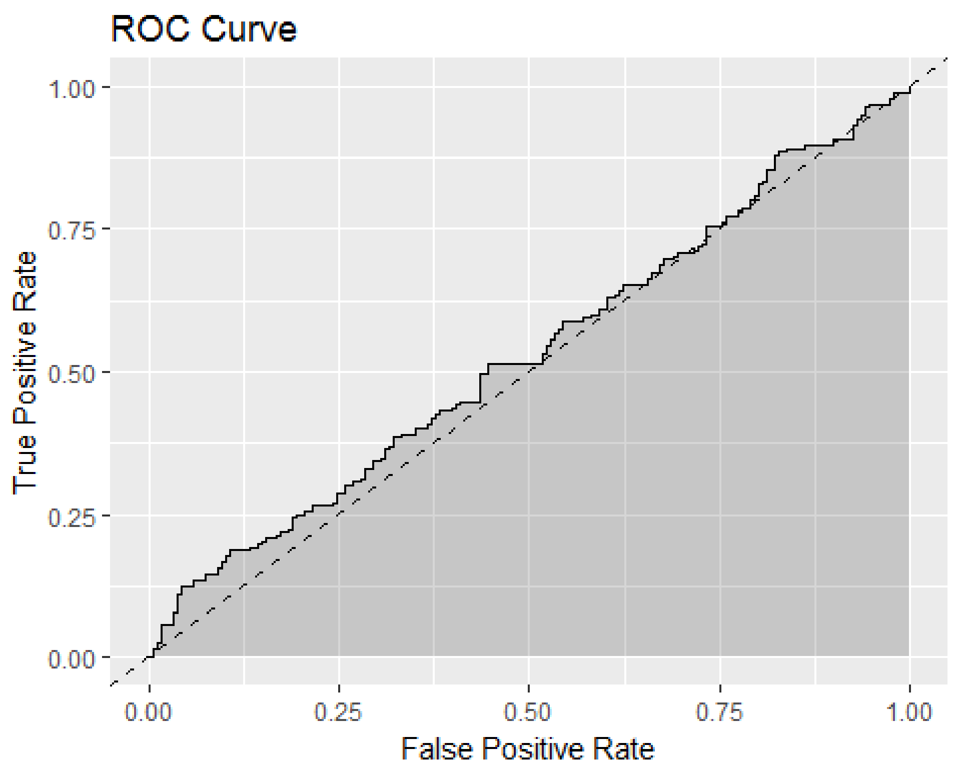
N1G2

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0   0   1
##       1 186 191
#Accuracy
mean(preds_y==ytest)
## [1] 0.505291
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.494709
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



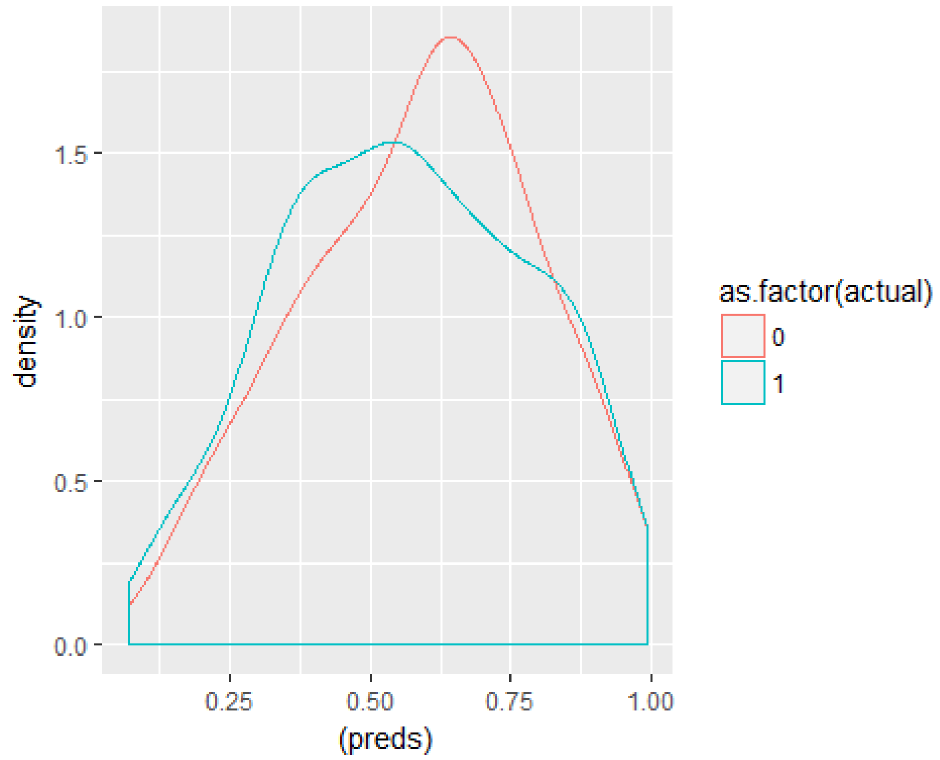
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5300179
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



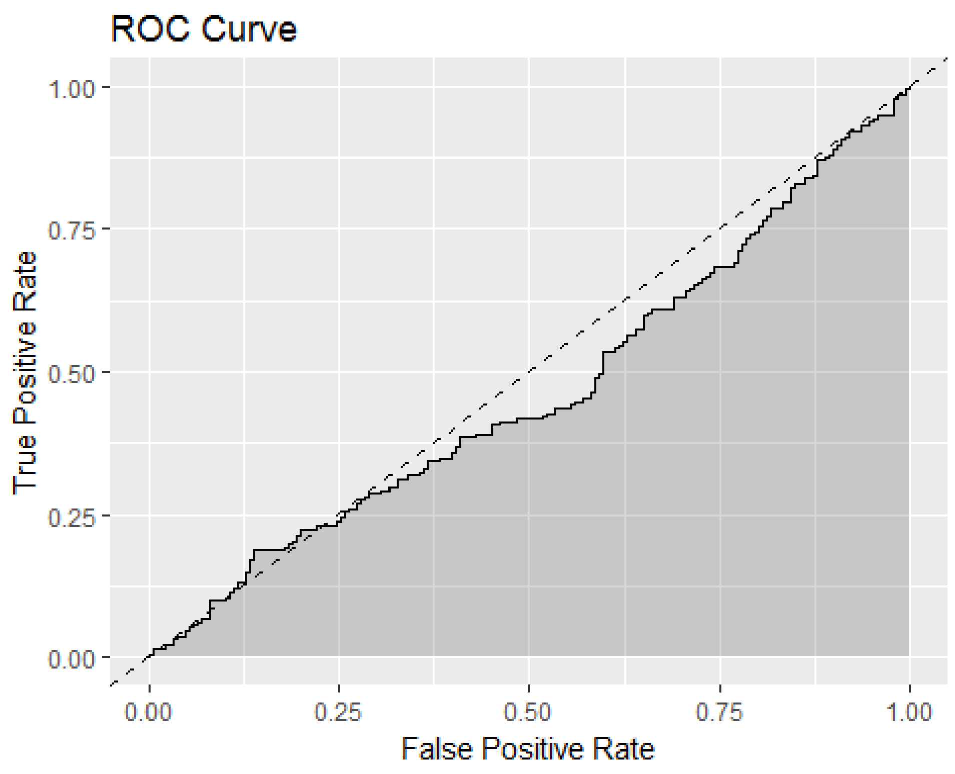
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  63  76
##       1 123 116
#Accuracy
mean(preds_y==ytest)
## [1] 0.473545
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.526455
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

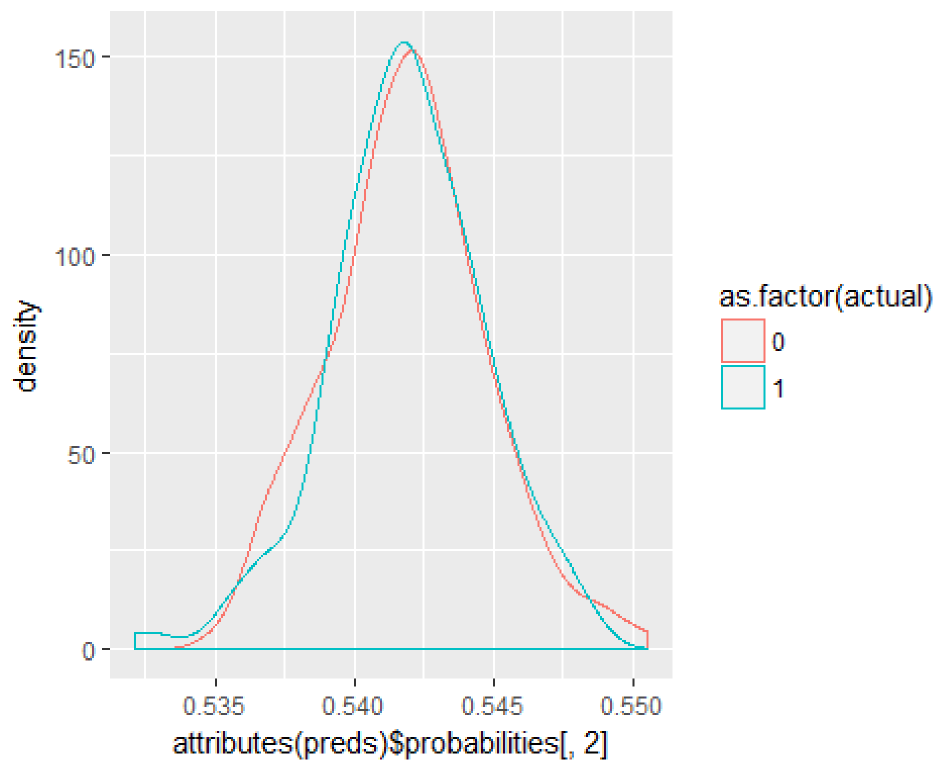


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4654738
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

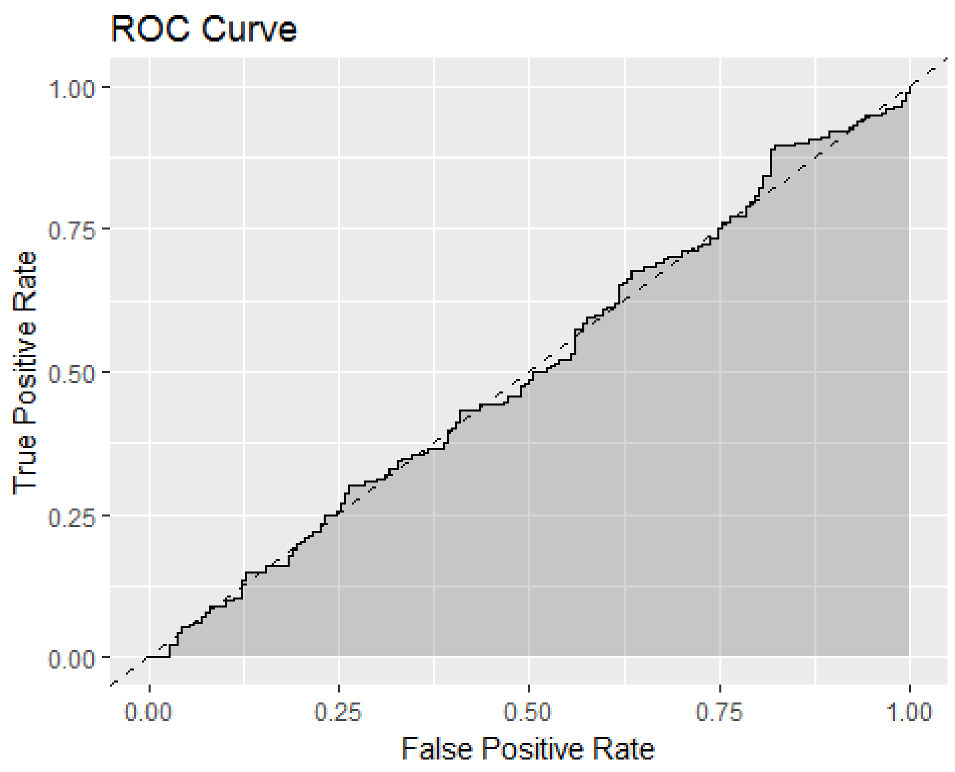


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'charli.hebdo' and 'zika.virus' constant. Cannot scale
## data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  83  91
##       1 103 101
#Accuracy
mean(preds_y==ytest)
## [1] 0.4867725
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5132275
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



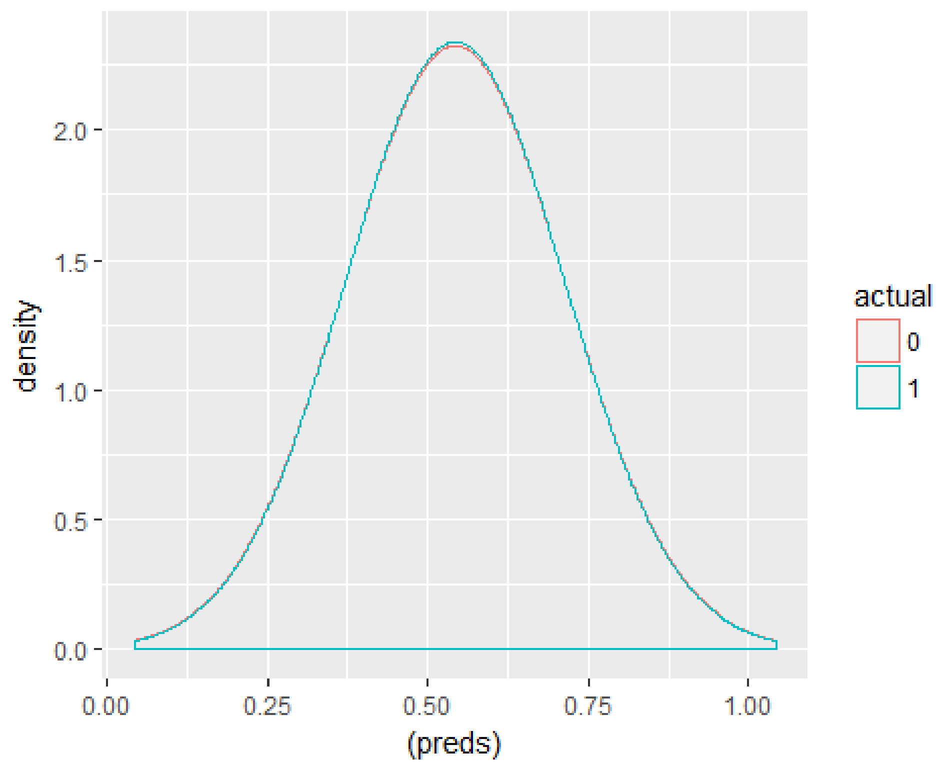
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5049283
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



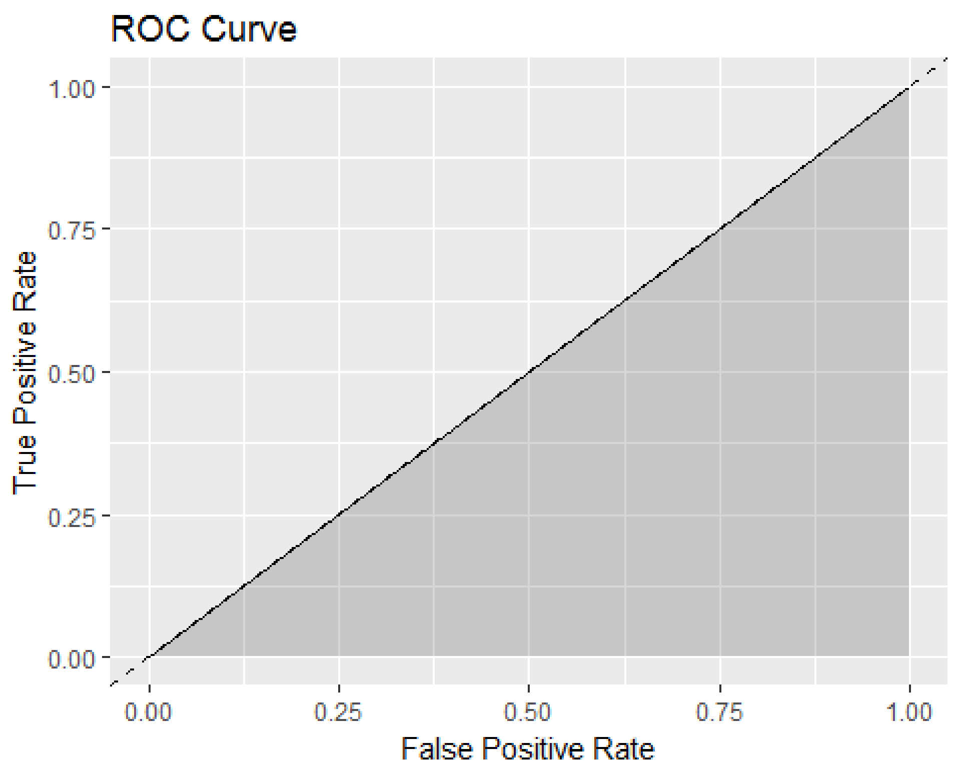
N1G3

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



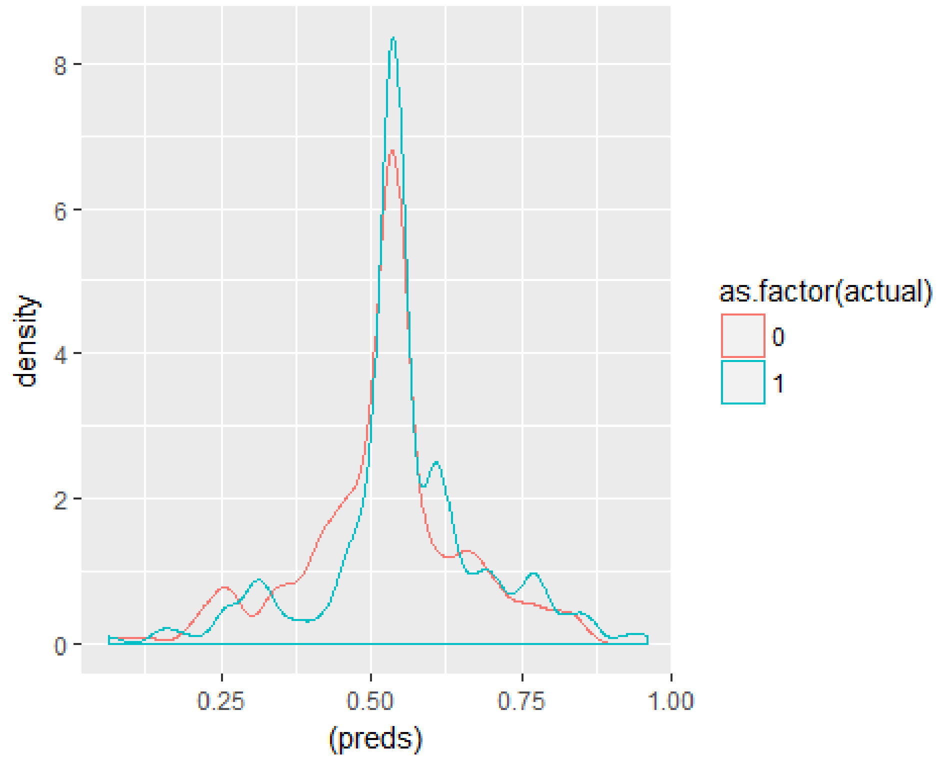
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



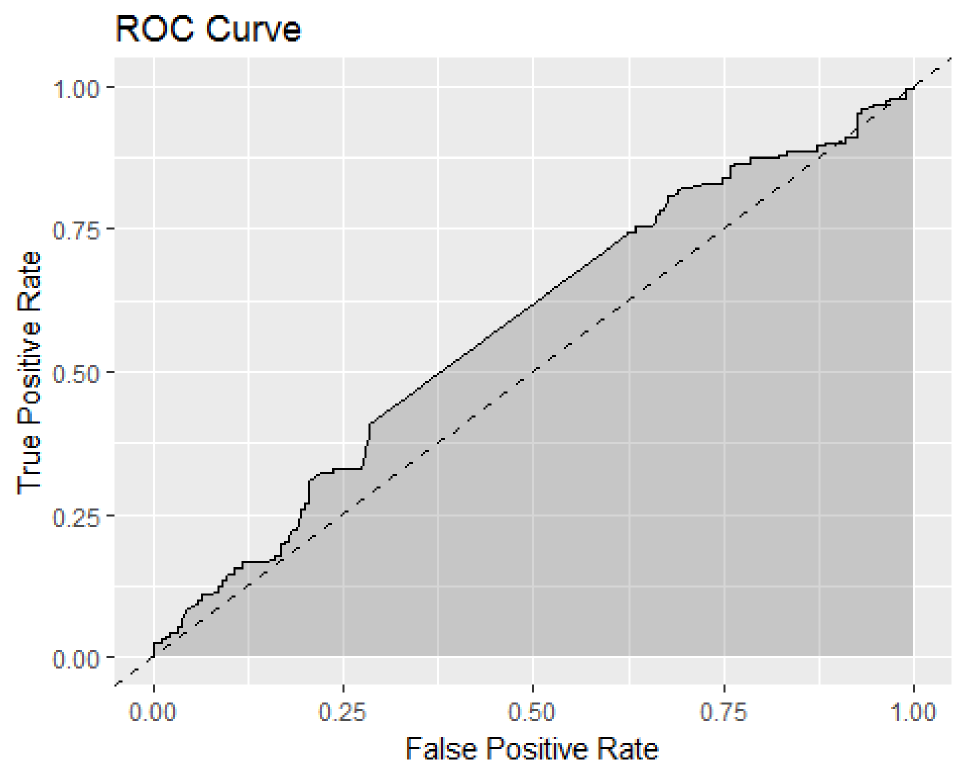
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  62  42
##       1 124 150
#Accuracy
mean(preds_y==ytest)
## [1] 0.5608466
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4391534
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

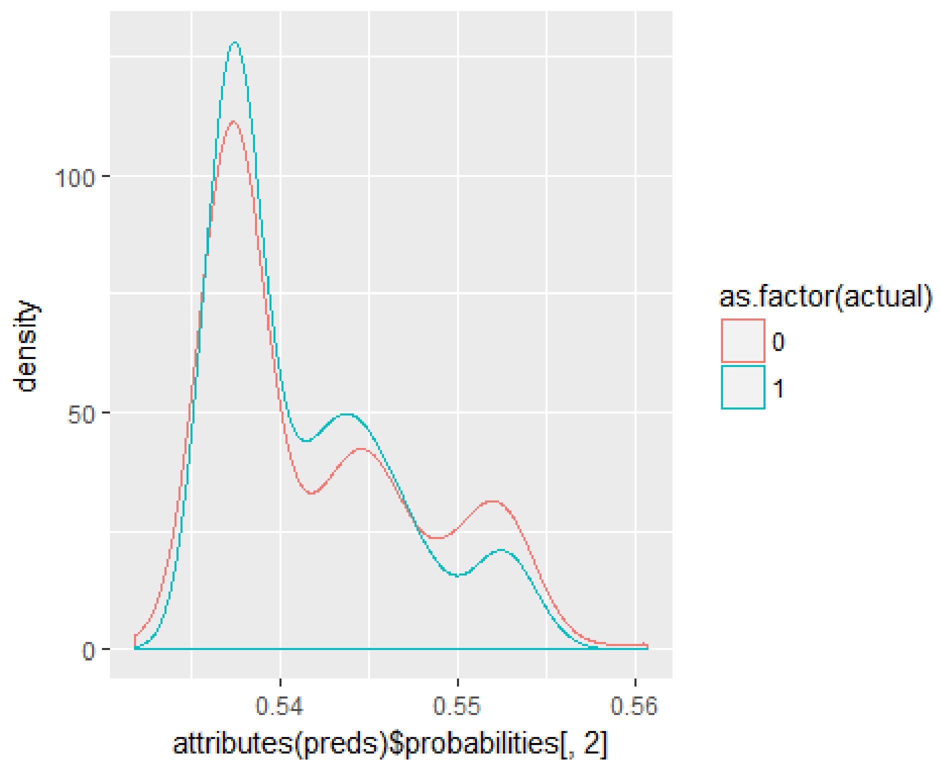


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5762489
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

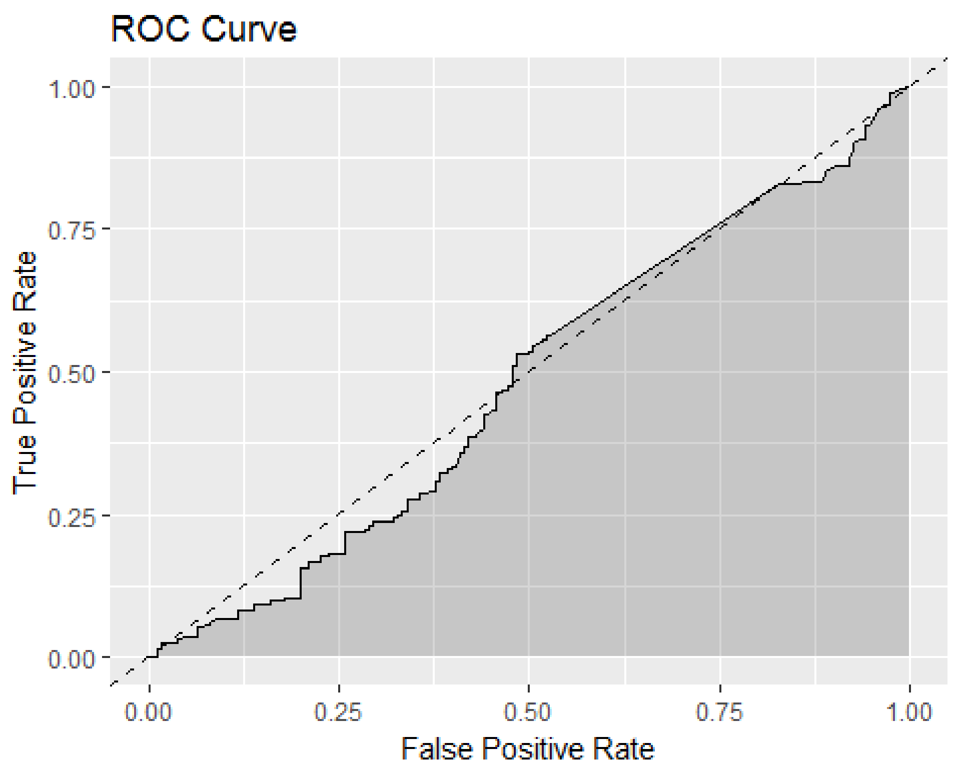


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  55  46
##       1 131 146
#Accuracy
mean(preds_y==ytest)
## [1] 0.531746
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.468254
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



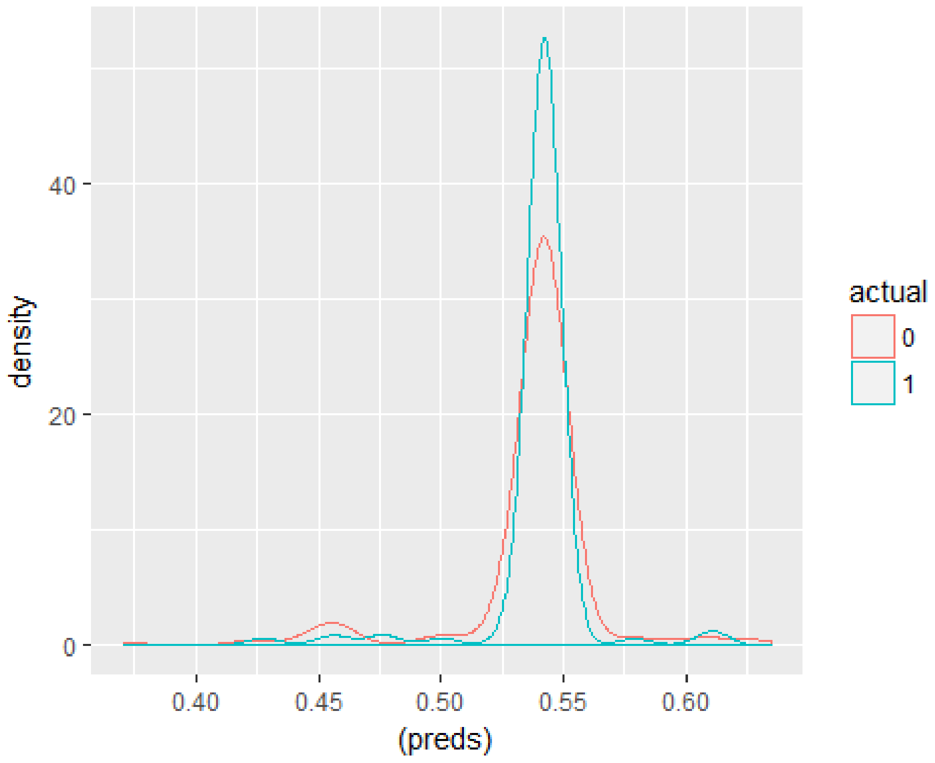
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4810848
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



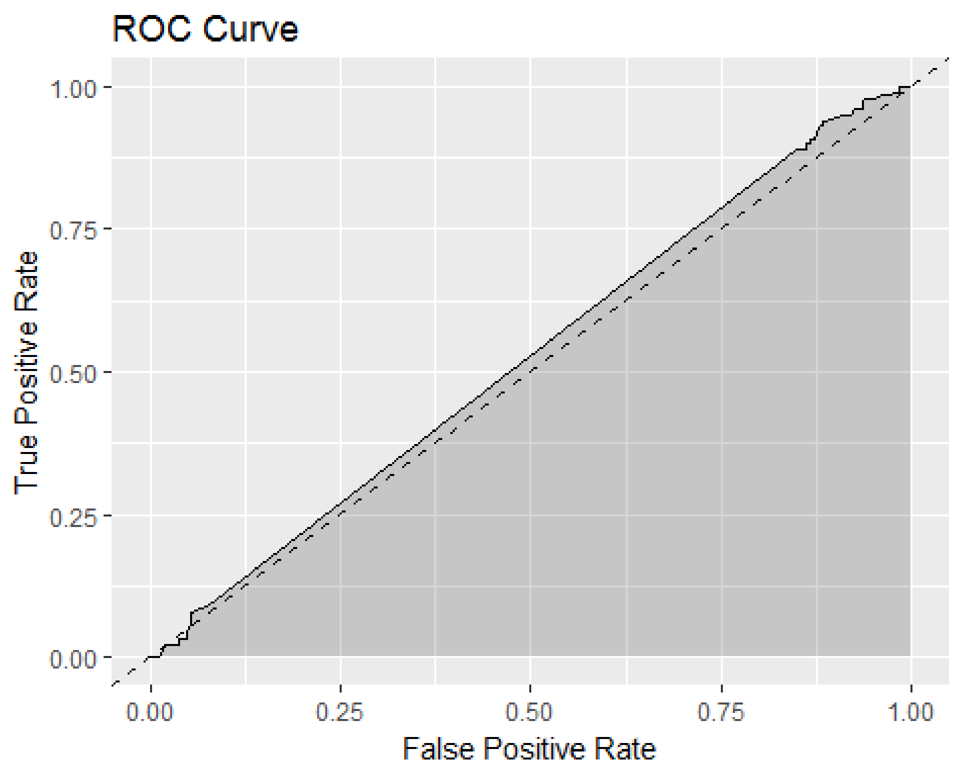
N1G4

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  14   8
##       1 172 184
#Accuracy
mean(preds_y==ytest)
## [1] 0.5238095
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4761905
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



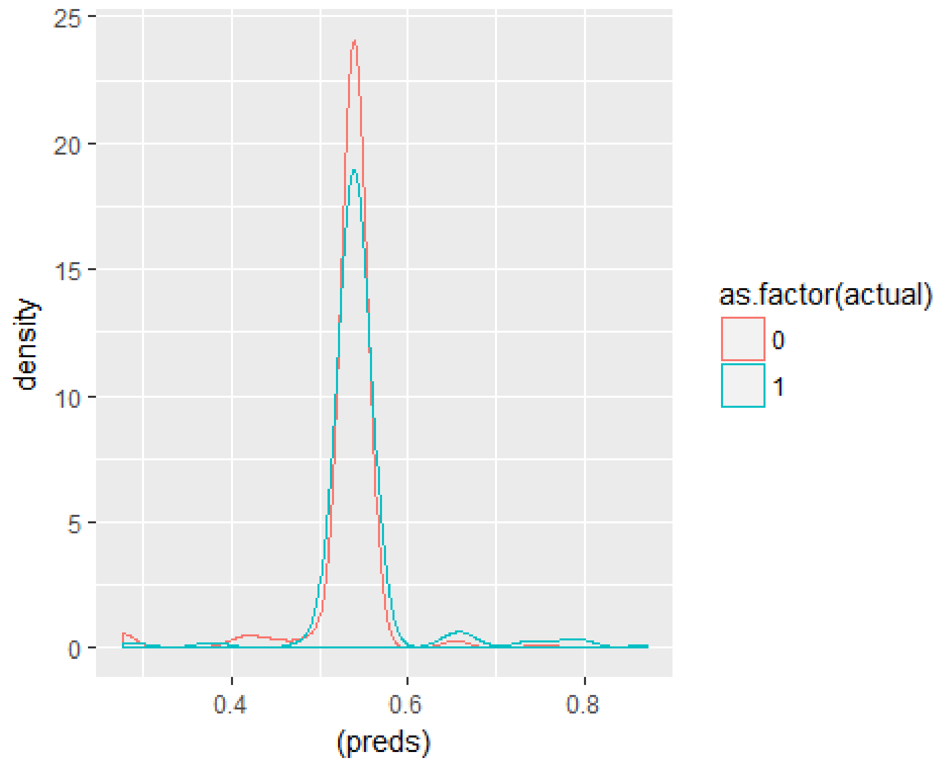
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5257476
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



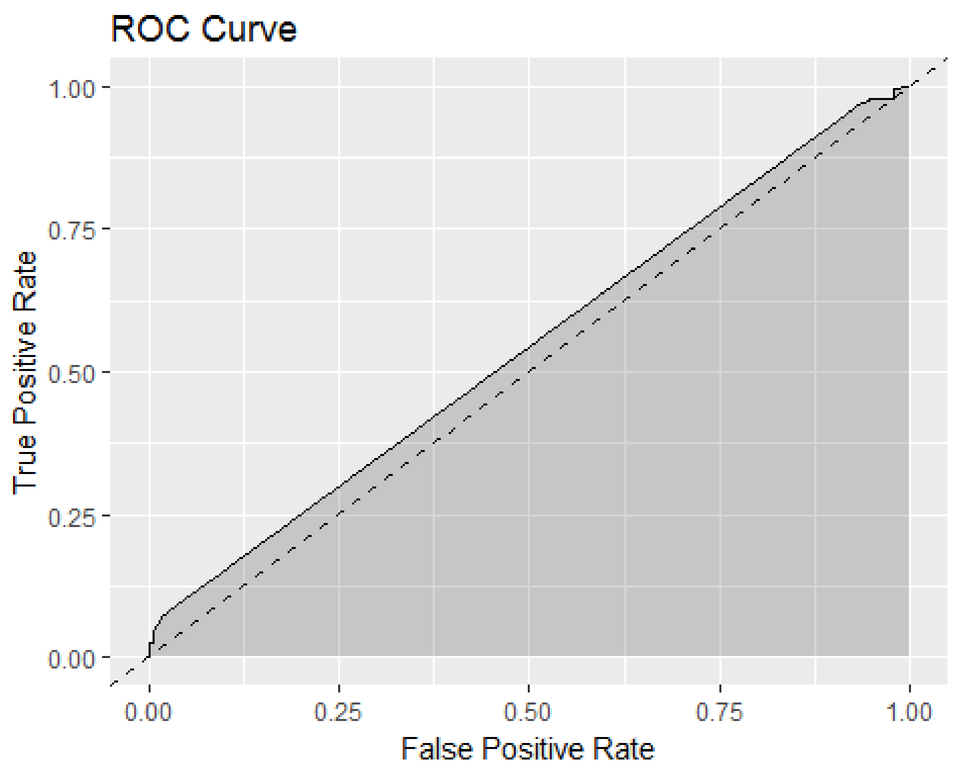
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  13   6
##       1 173 186
#Accuracy
mean(preds_y==ytest)
## [1] 0.526455
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.473545
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



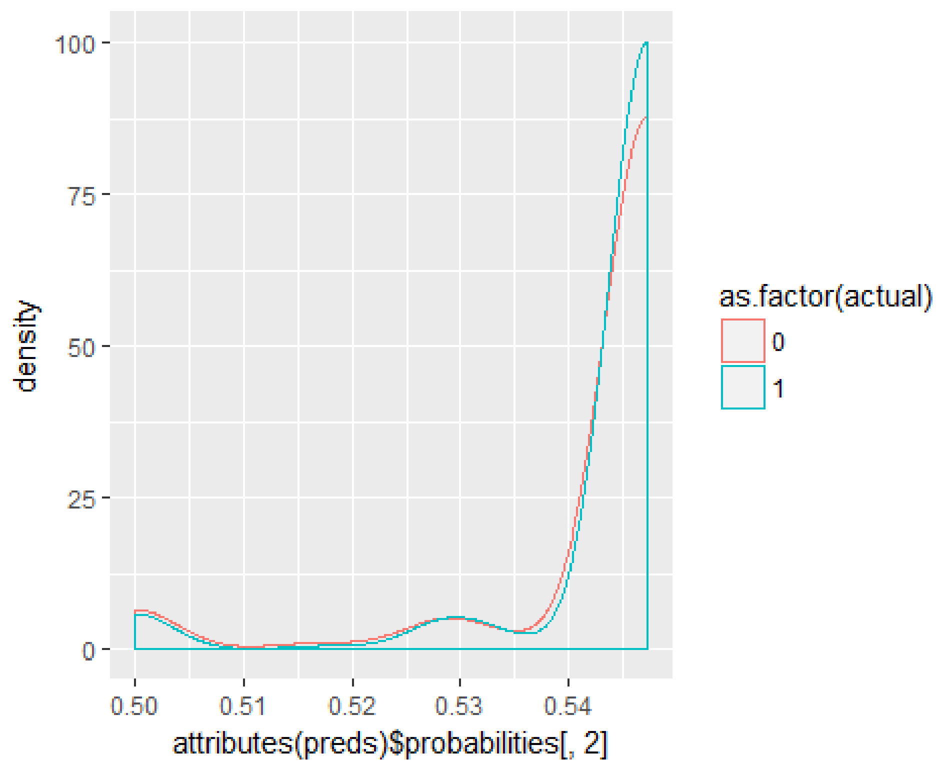
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5431368
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



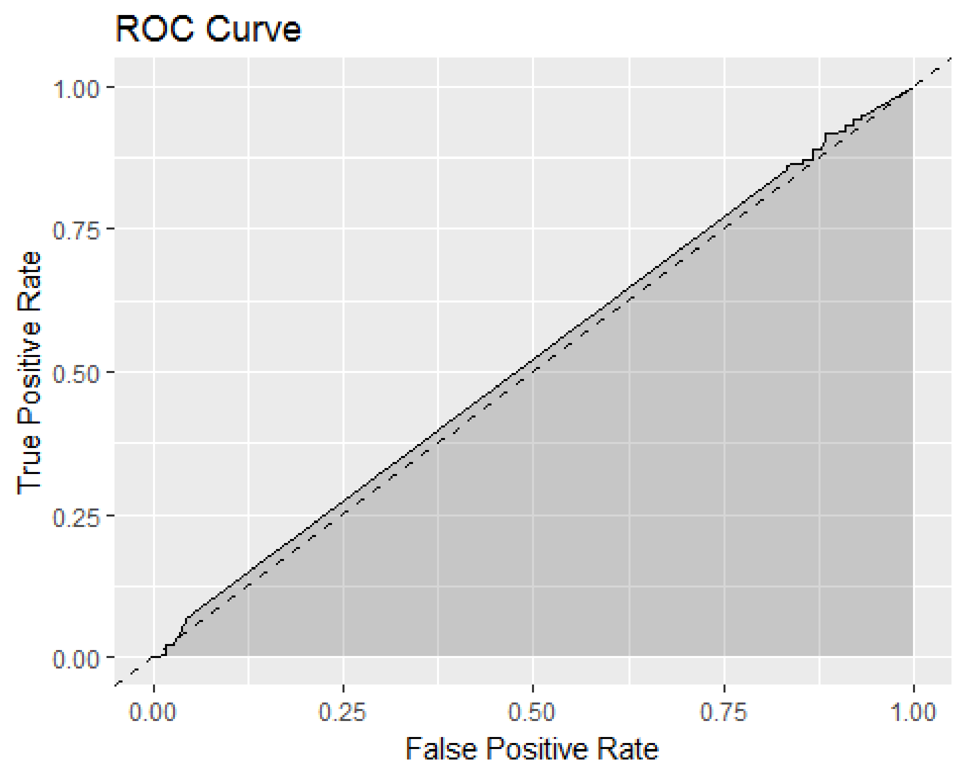
RandomForest Model

SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  14  11
##       1 172 181
#Accuracy
mean(preds_y==ytest)
## [1] 0.515873
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.484127
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

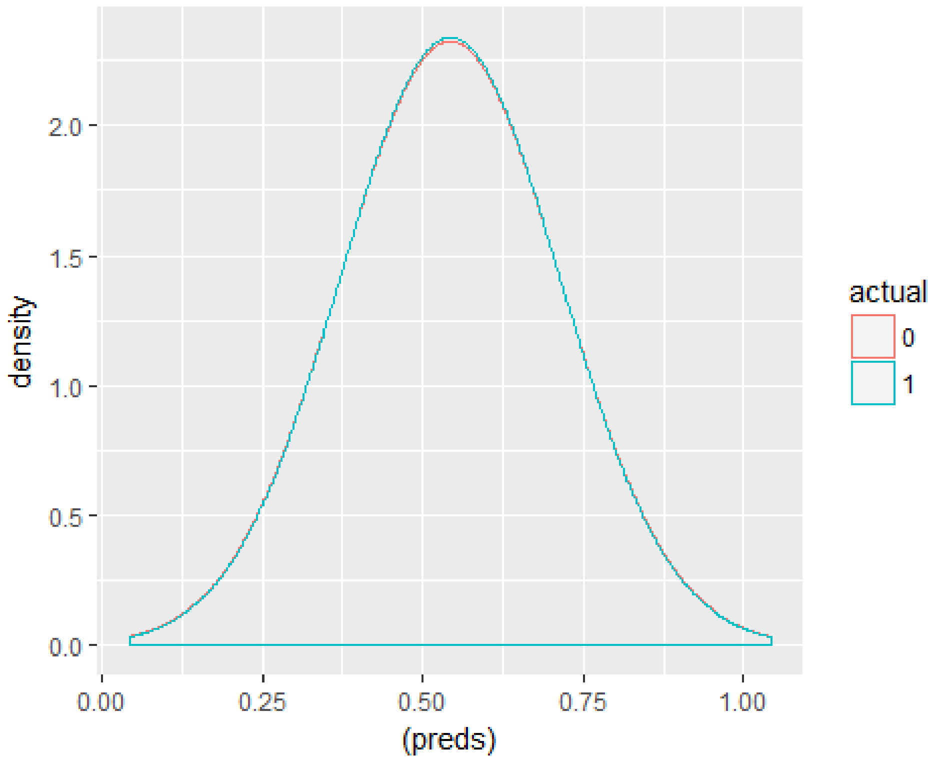


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5203433
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

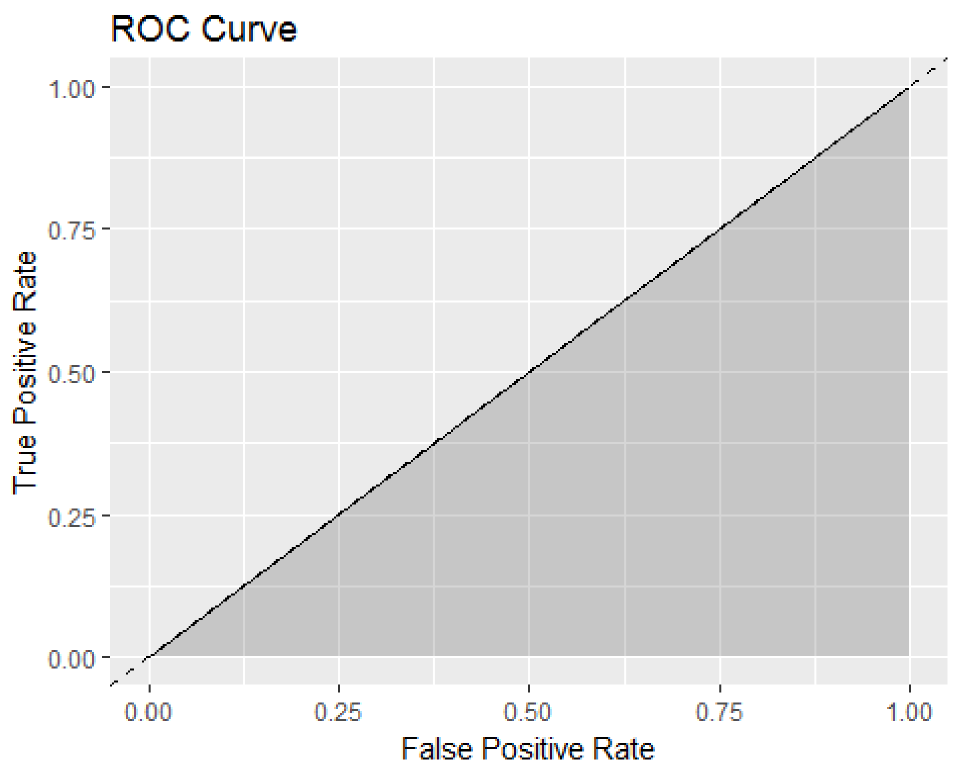


N1G23
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



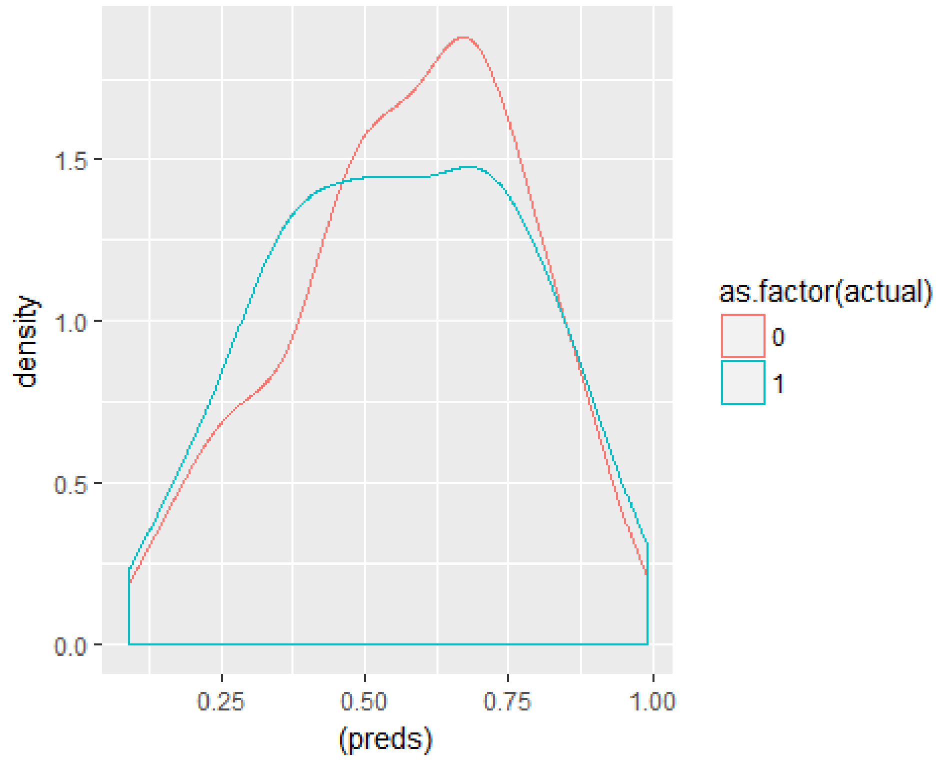
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



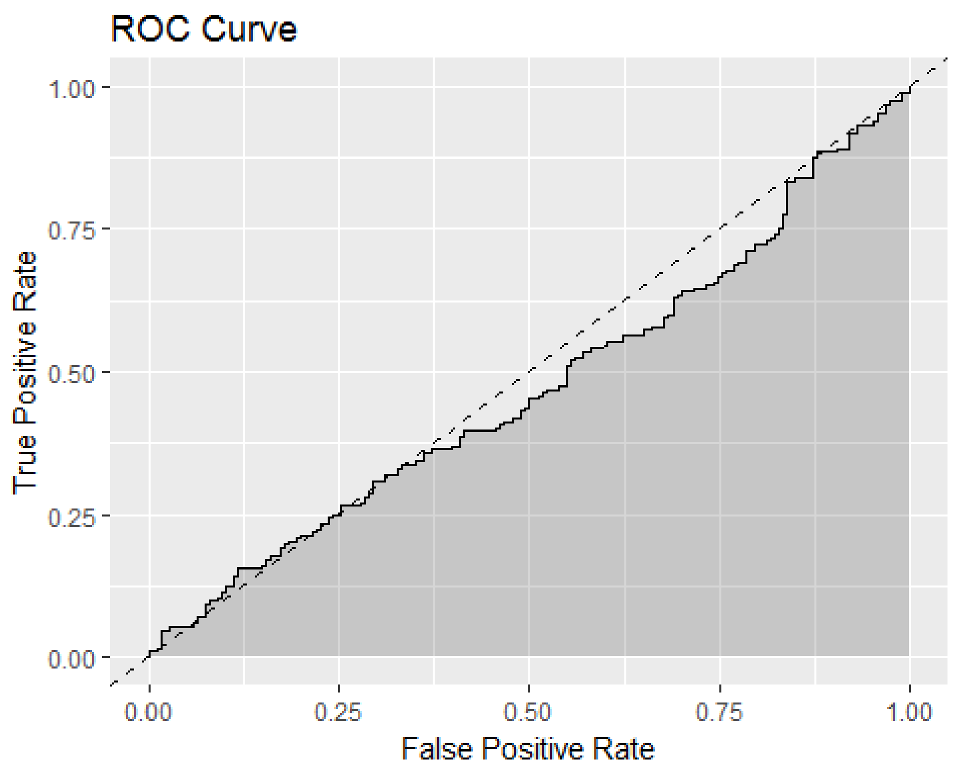
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  60  79
##       1 126 113
#Accuracy
mean(preds_y==ytest)
## [1] 0.457672
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.542328
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

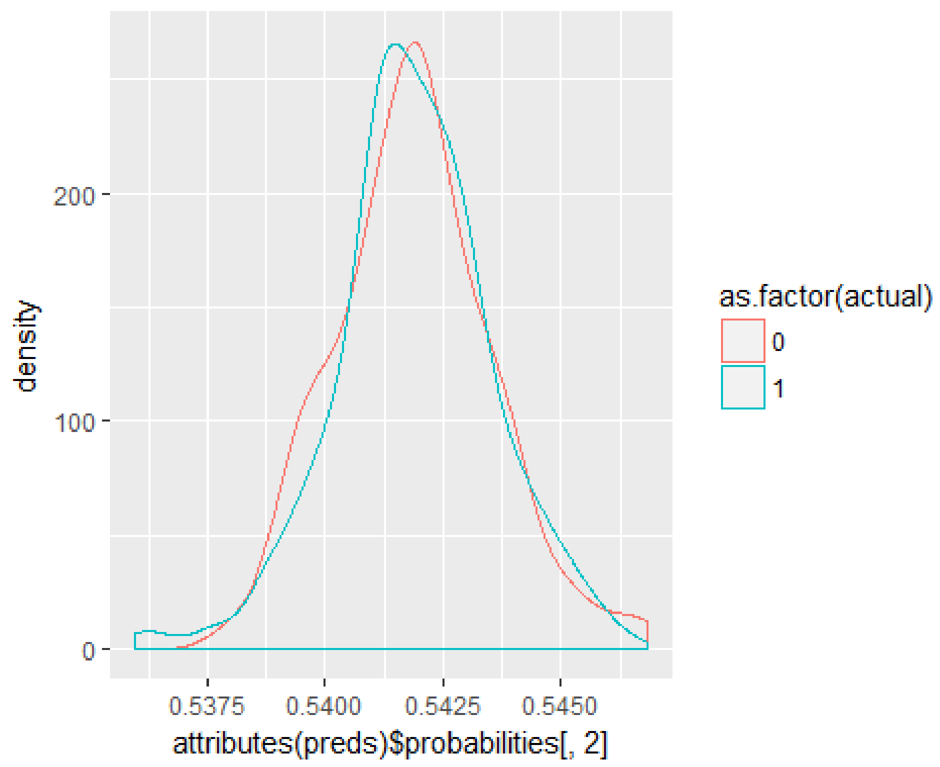


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4718722
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

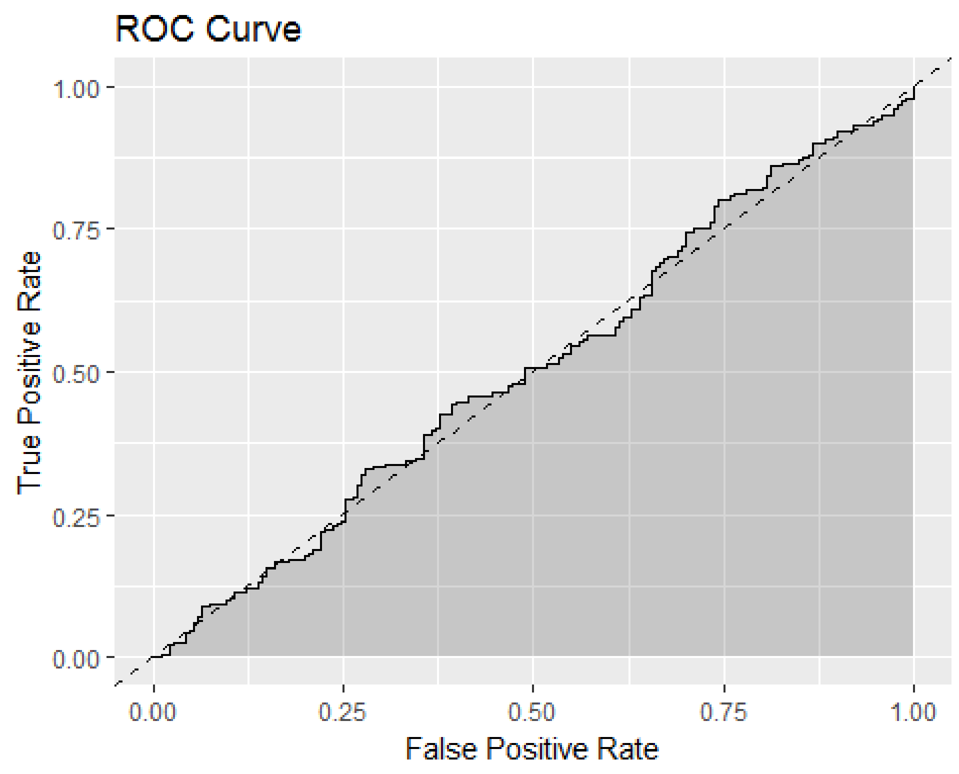


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'charli.hebdo' and 'zika.virus' constant. Cannot scale
## data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y  0  1
##       0 89 93
##       1 97 99
#Accuracy
mean(preds_y==ytest)
## [1] 0.4973545
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5026455
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

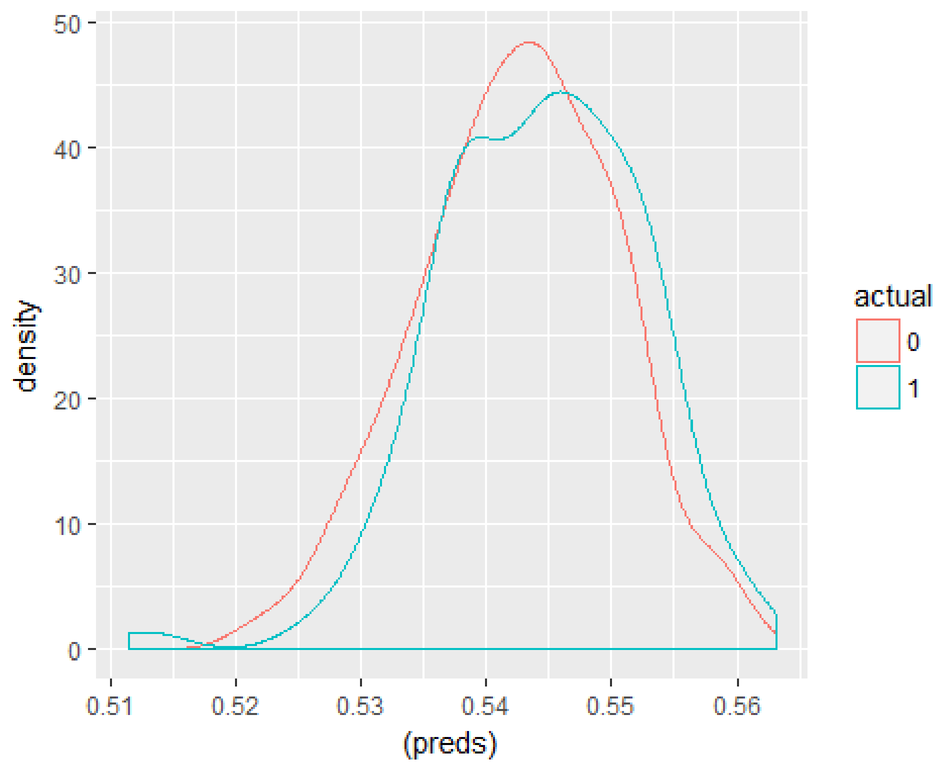


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5082325
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

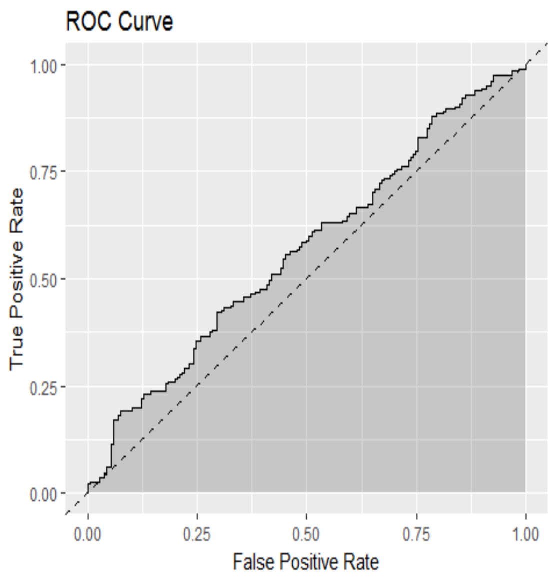


N1G123
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```

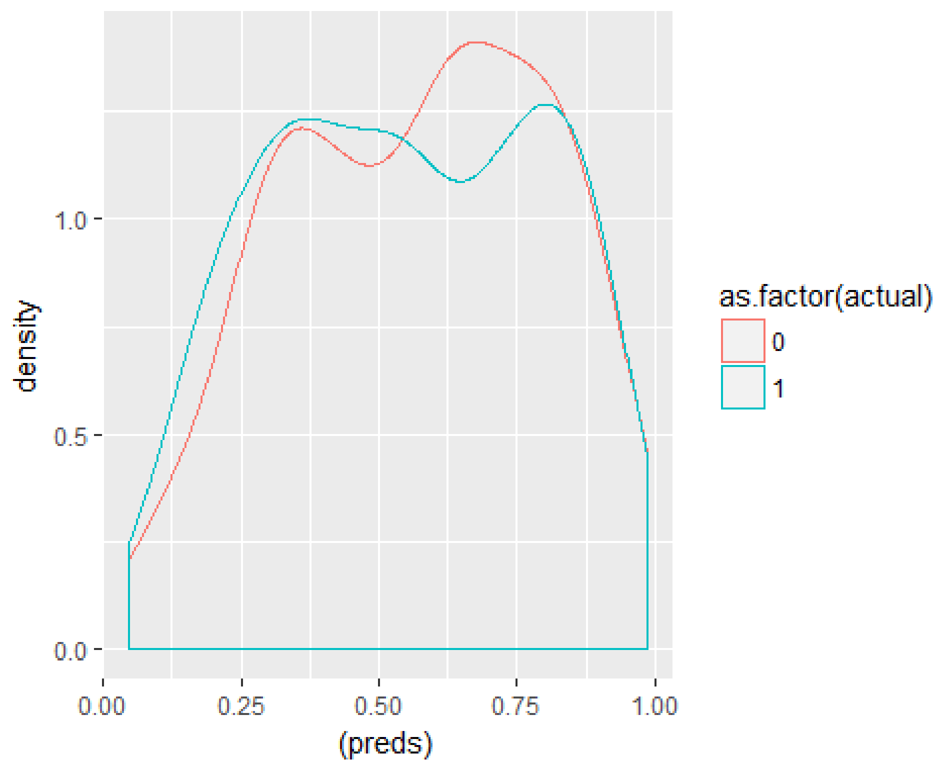


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5662242
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```

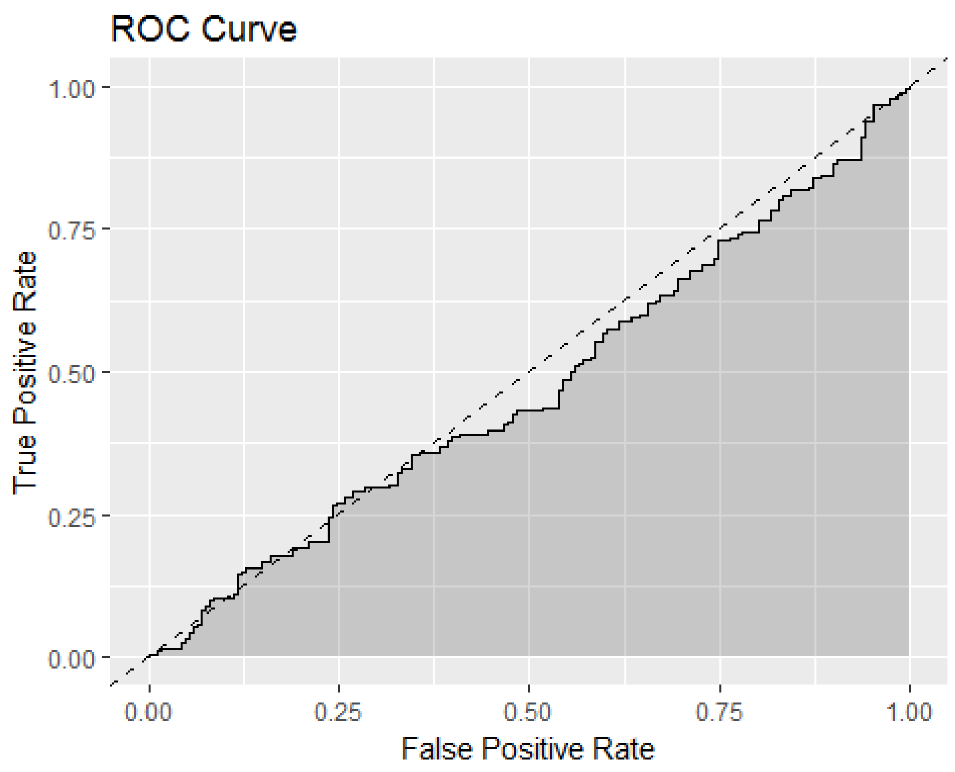


XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  74  83
##       1 112 109
#Accuracy
mean(preds_y==ytest)
## [1] 0.484127
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.515873
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



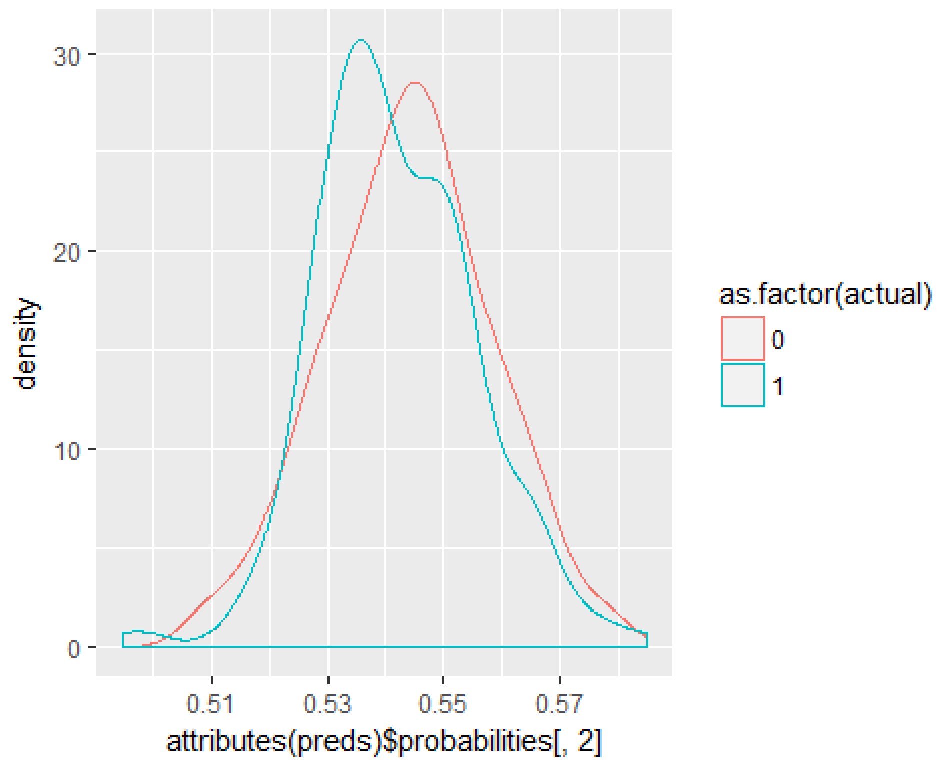
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4740423
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



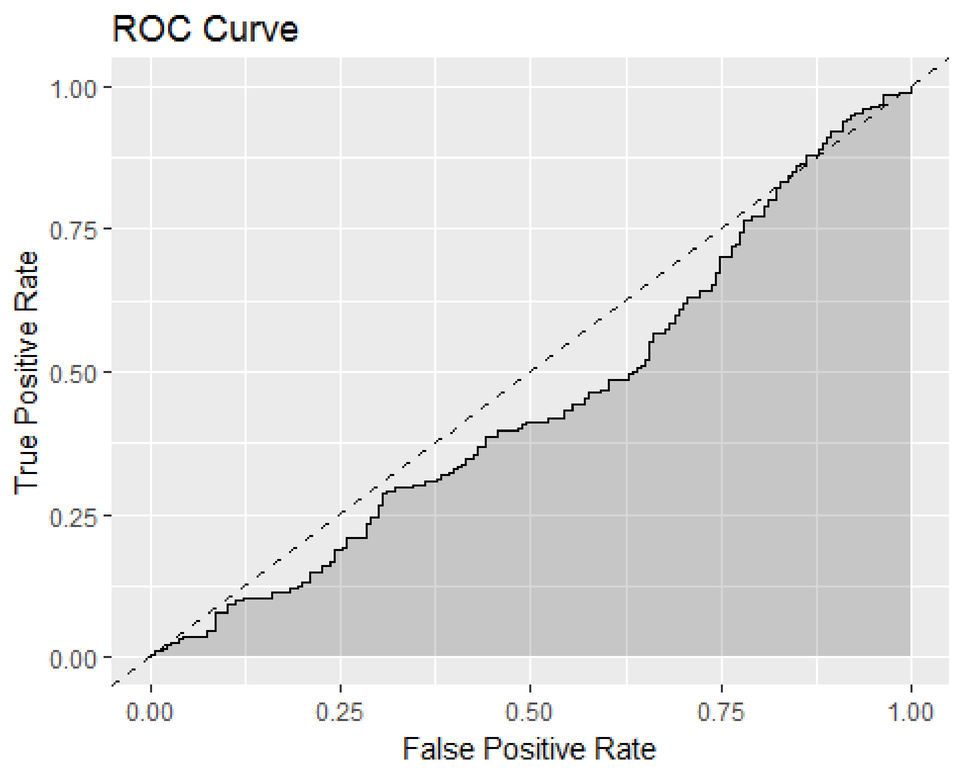
RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'cecil' and 'charli.hebdo' and 'corbyn' and 'hebdo'
## and 'jeremi.corbyn' and 'migrant.crisi' and 'palmyra' and 'panama.paper'
## and 'ramadi' and 'russian.air' and 'saudil' and 'zika' and 'zika.virus'
## constant. Cannot scale data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  67  95
##       1 119  97
#Accuracy
mean(preds_y==ytest)
## [1] 0.4338624
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5661376
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
```

```
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

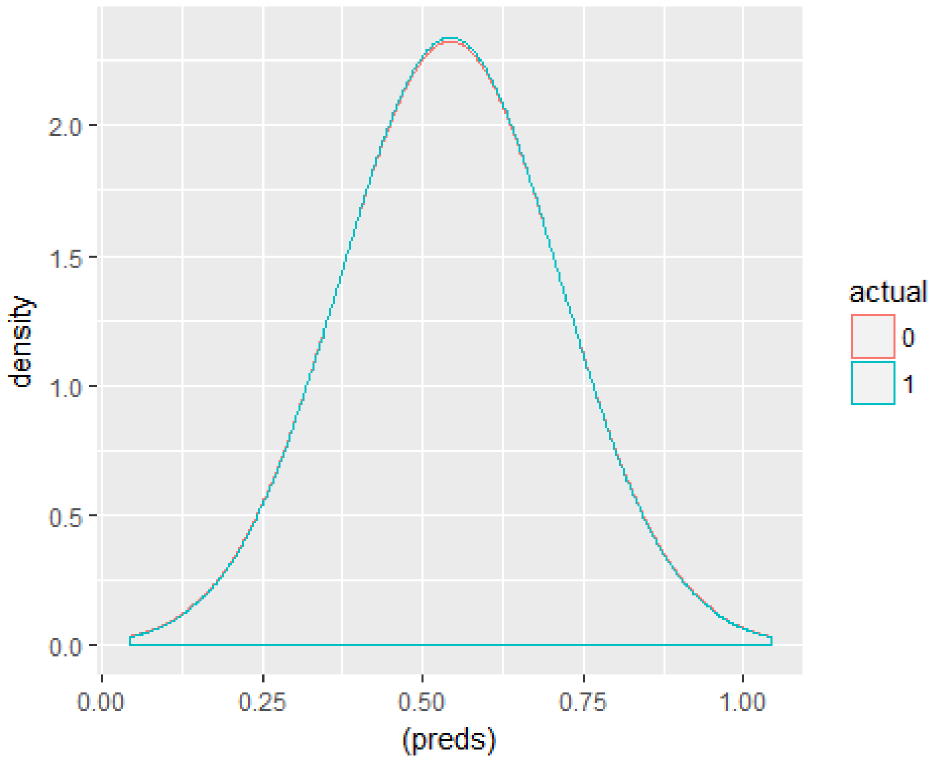


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4481407
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```

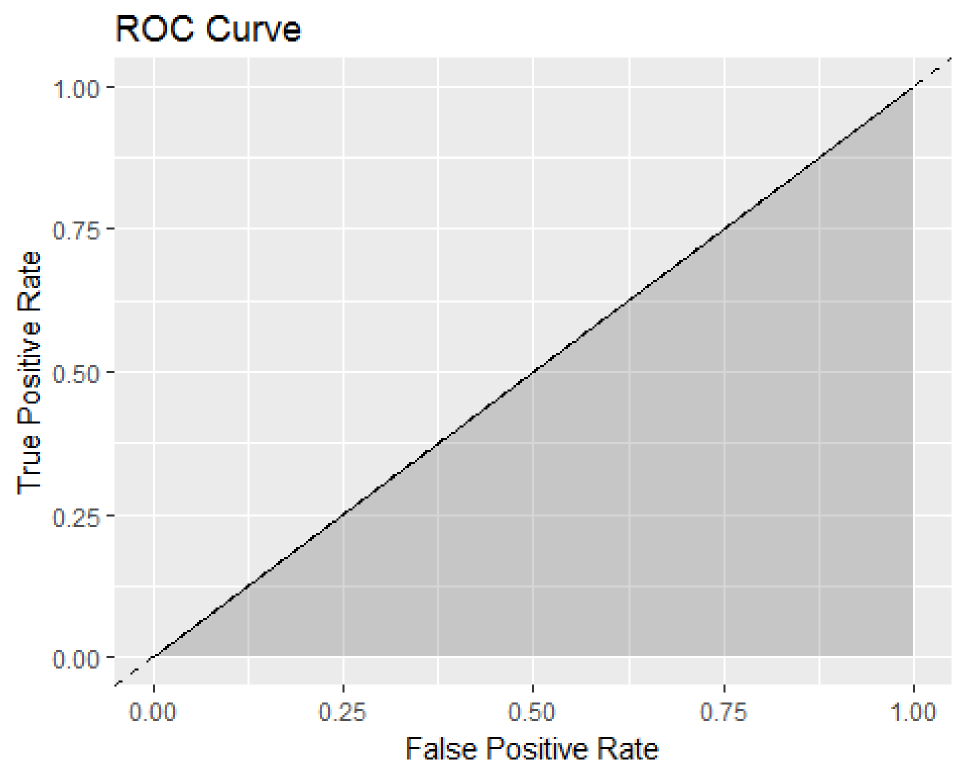


N3G1
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



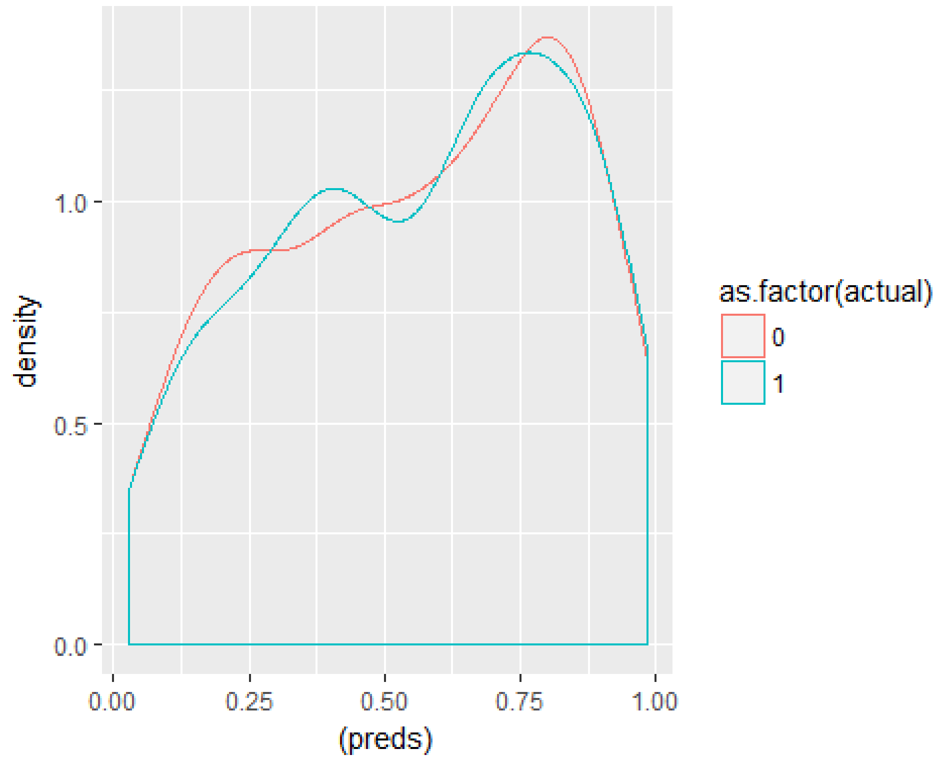
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



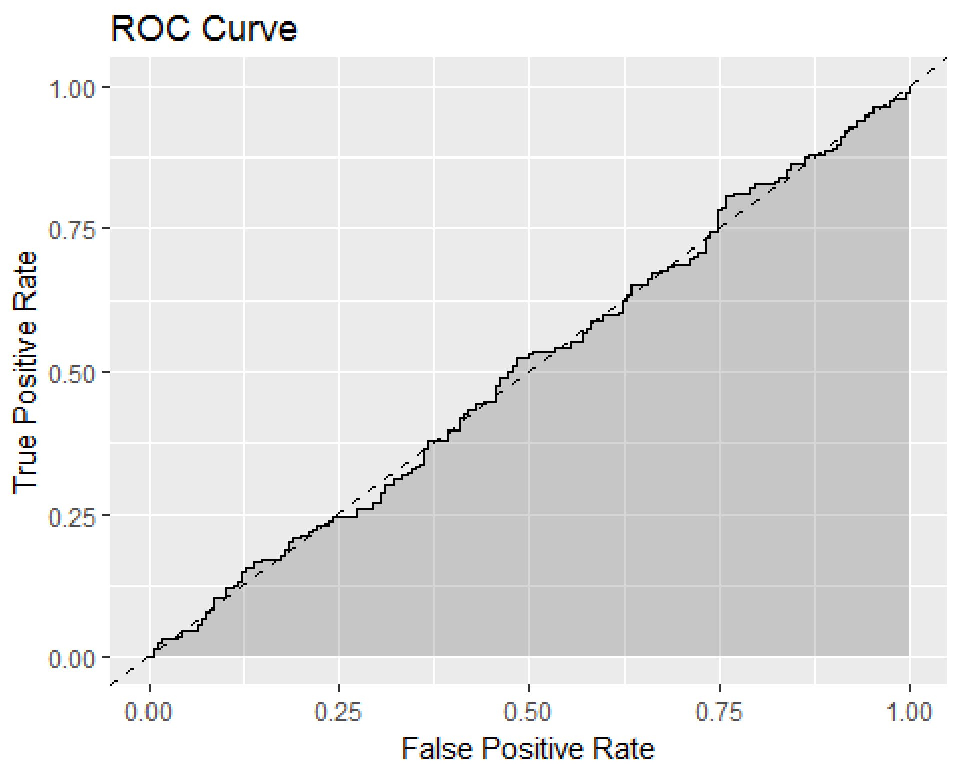
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  78  80
##       1 108 112
#Accuracy
mean(preds_y==ytest)
## [1] 0.5026455
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4973545
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

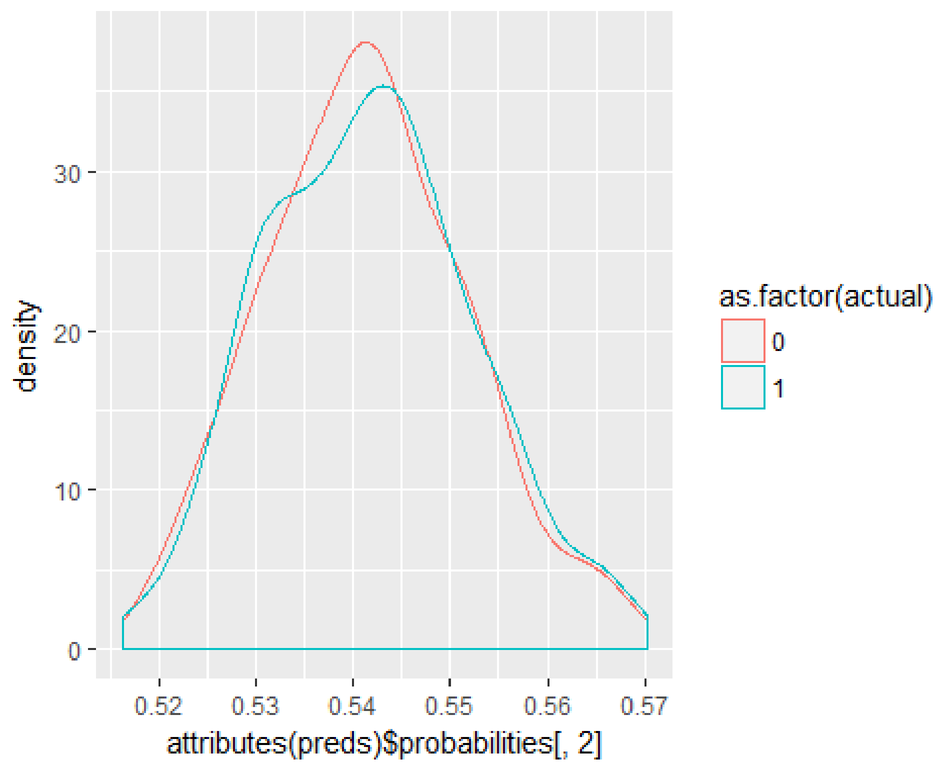


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5029962
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

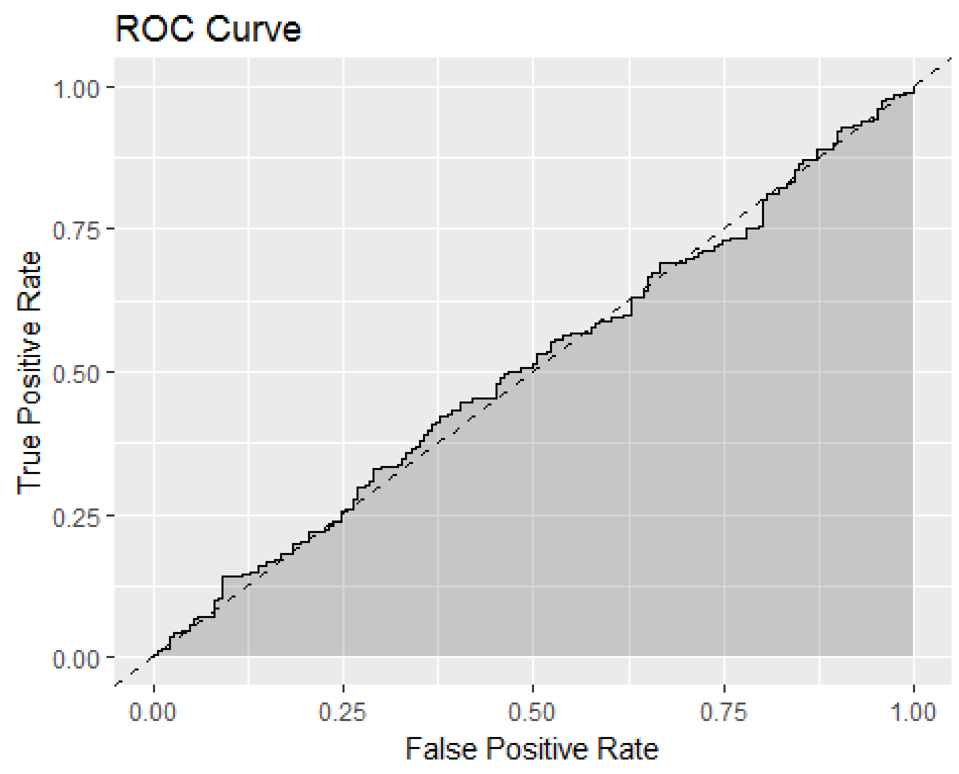


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  75  86
##       1 111 106
#Accuracy
mean(preds_y==ytest)
## [1] 0.478836
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.521164
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



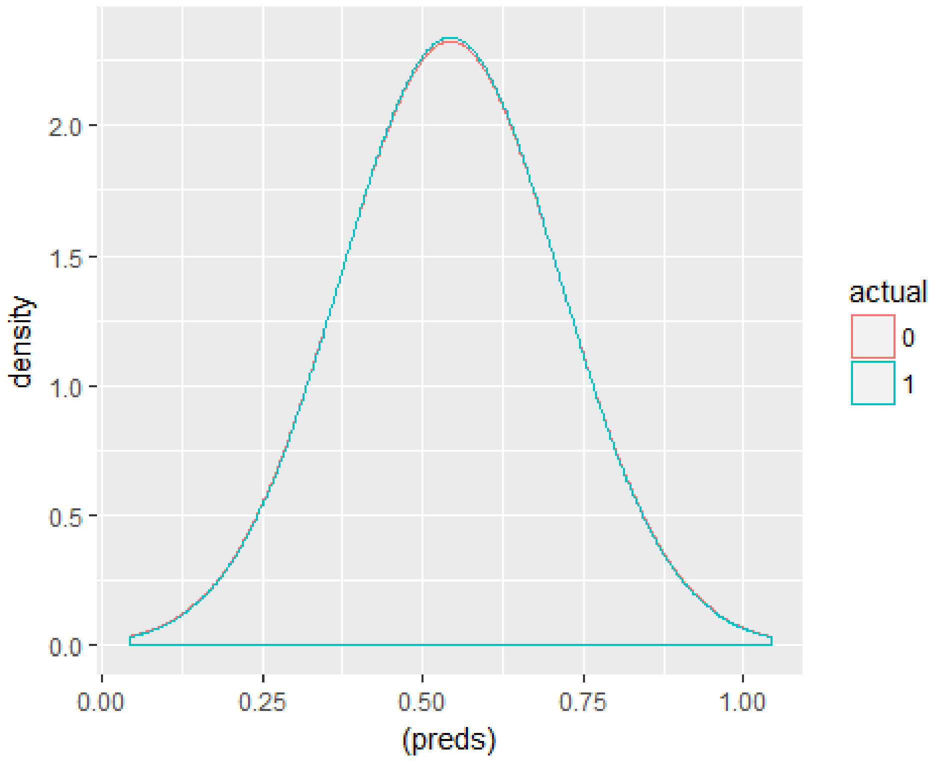
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5079245
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```



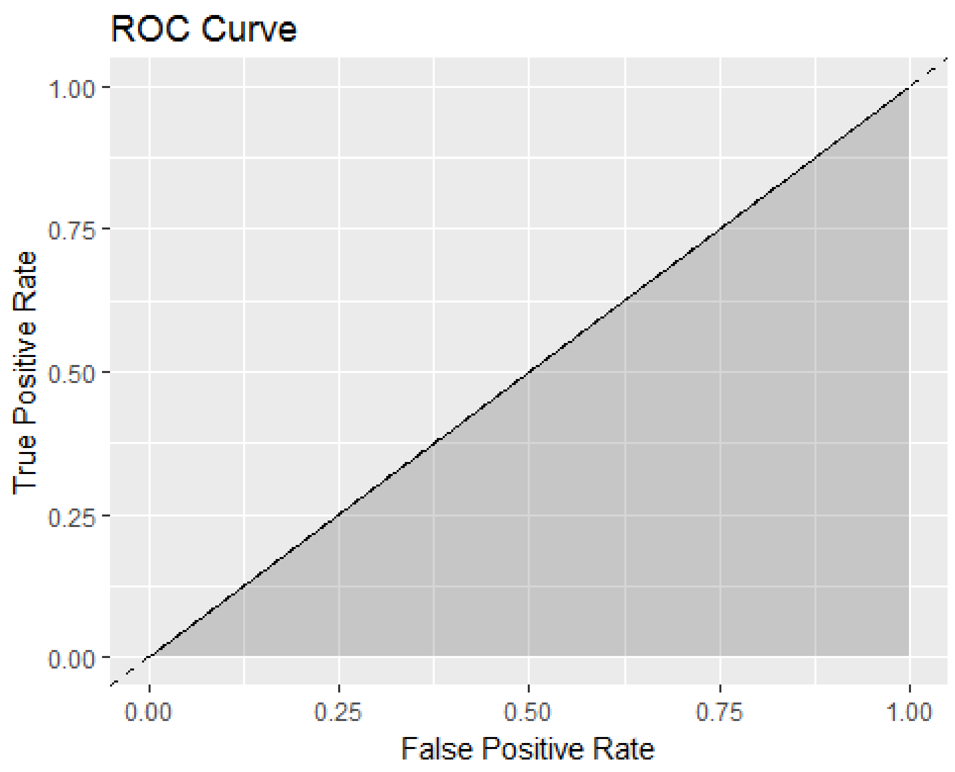
N3G2

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



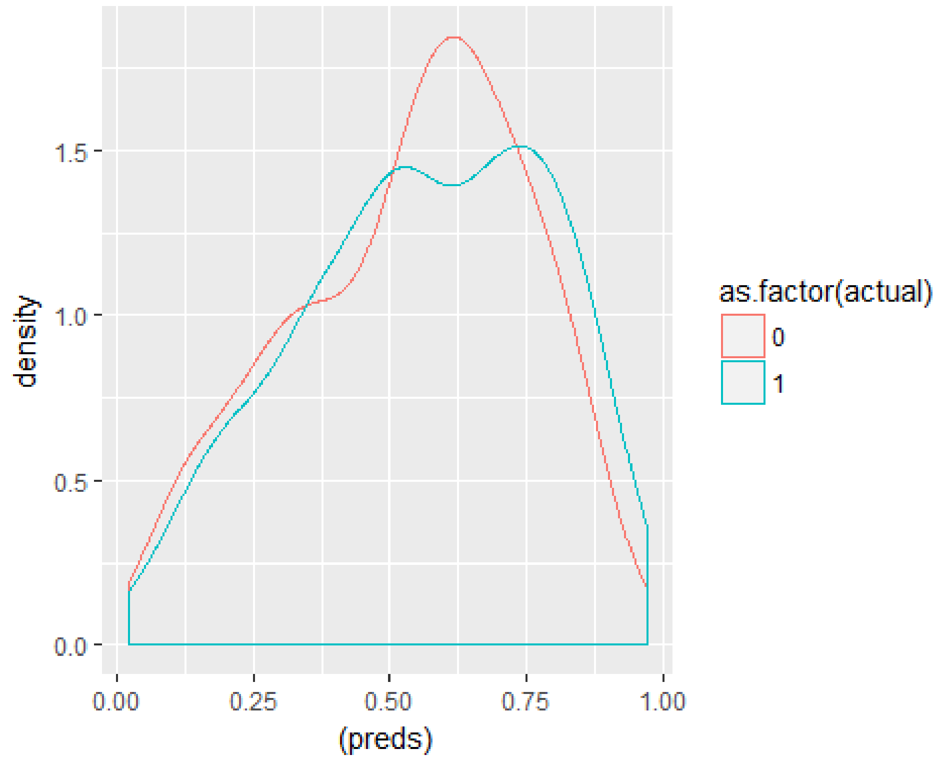
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



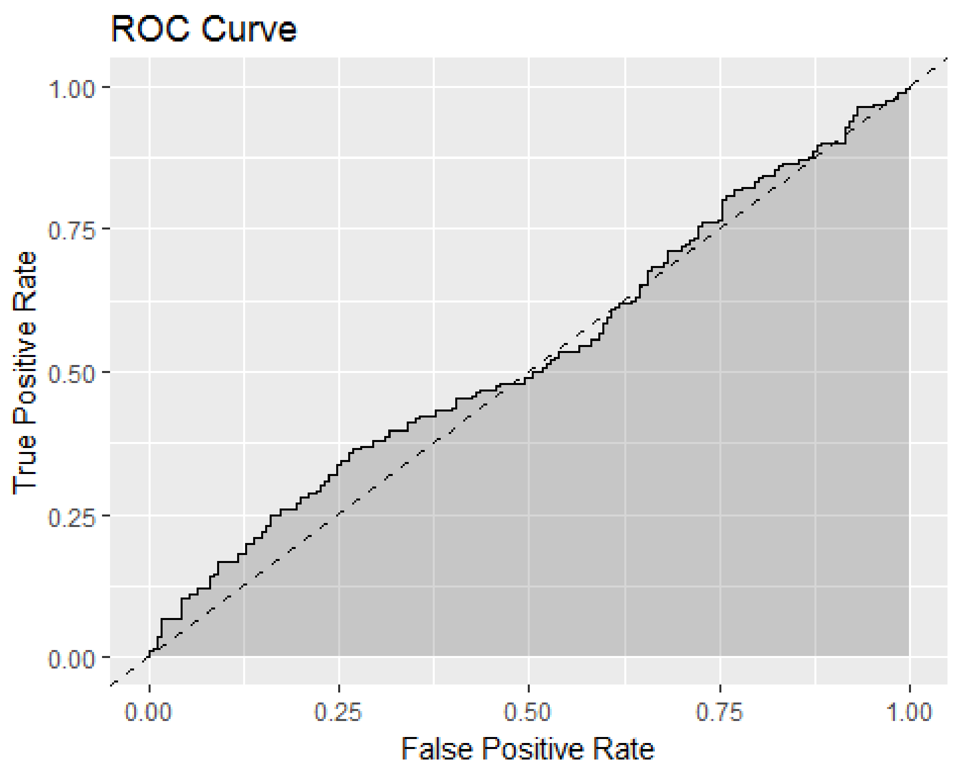
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  72  74
##       1 114 118
#Accuracy
mean(preds_y==ytest)
## [1] 0.5026455
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4973545
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.531558
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

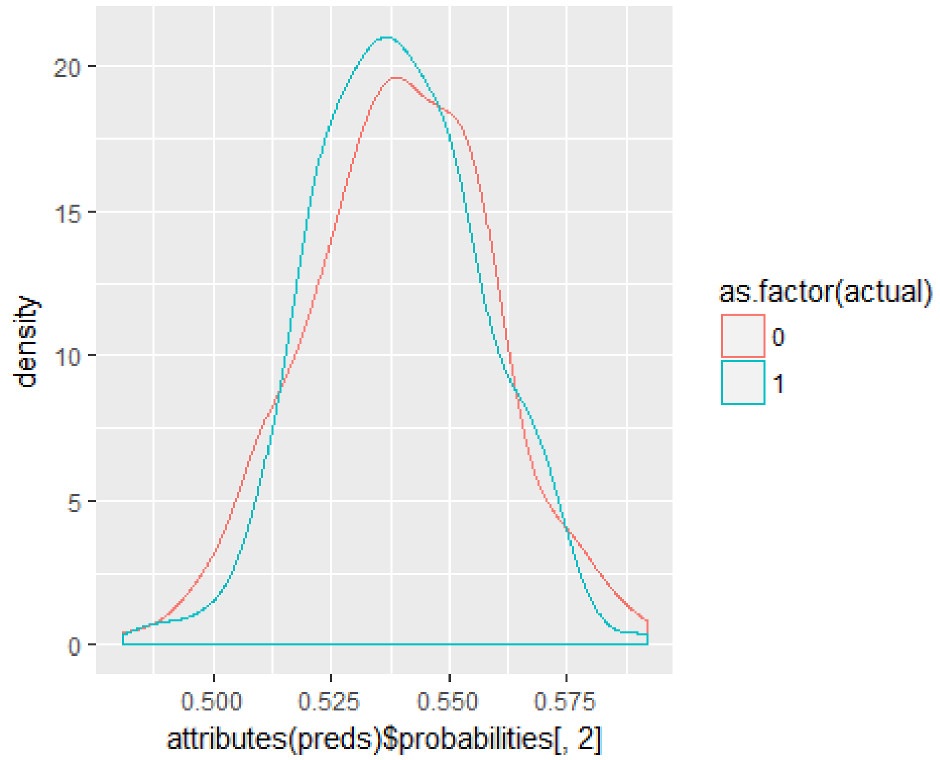


RandomForest Model

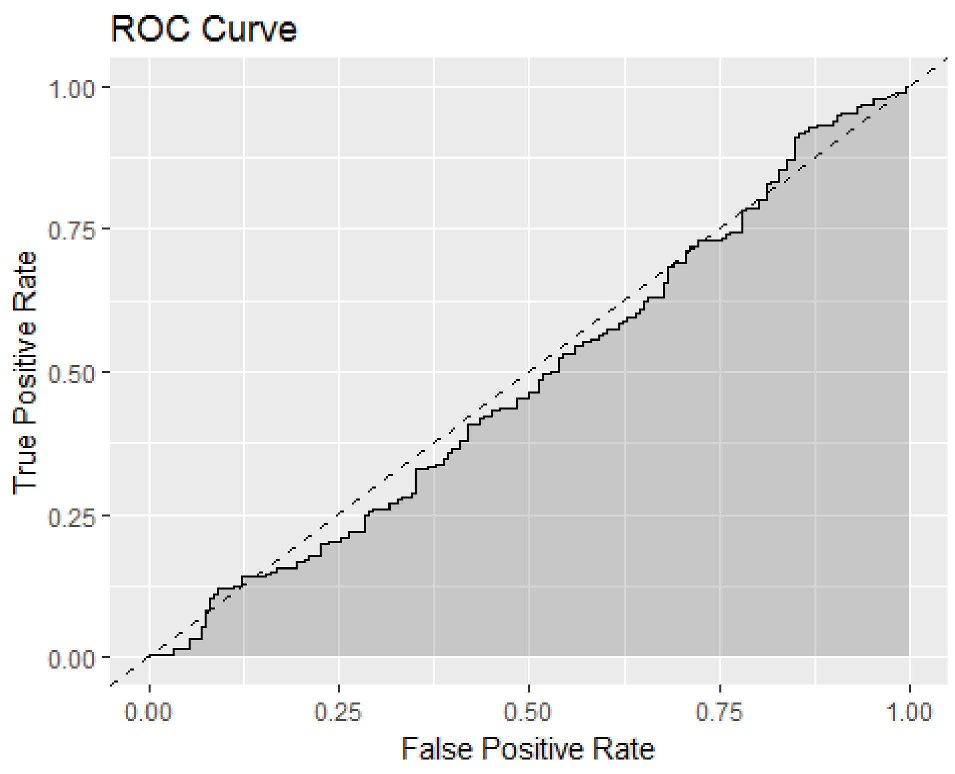
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'attack.pari' and 'charli.hebdo' and 'china.stock' and
## 'jeremi.corbyn' and 'leav.eu' and 'migrant.crisi' and 'panama.paper' and
## 'pari.climat' and 'russian.air' and 'saudil.coalit' and 'snooper.charter'
## and 'state.syria' and 'zika.virus' constant. Cannot scale data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  76  73
##       1 110 119
#Accuracy
mean(preds_y==ytest)
## [1] 0.515873
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.484127
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
```

```
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



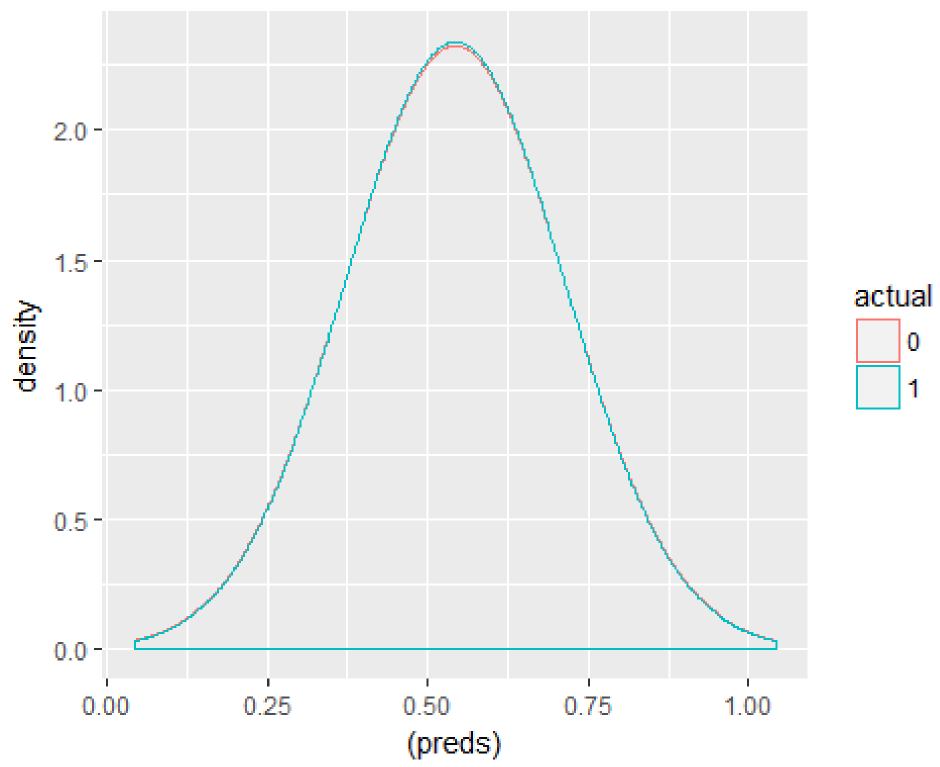
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.485075
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



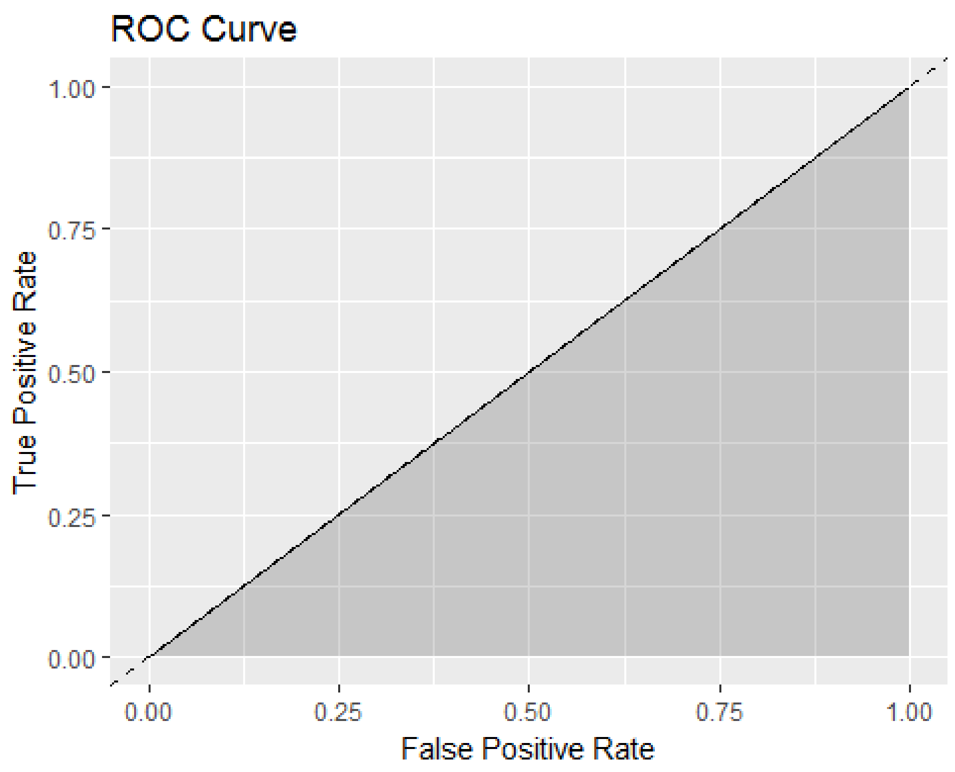
N3G3

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

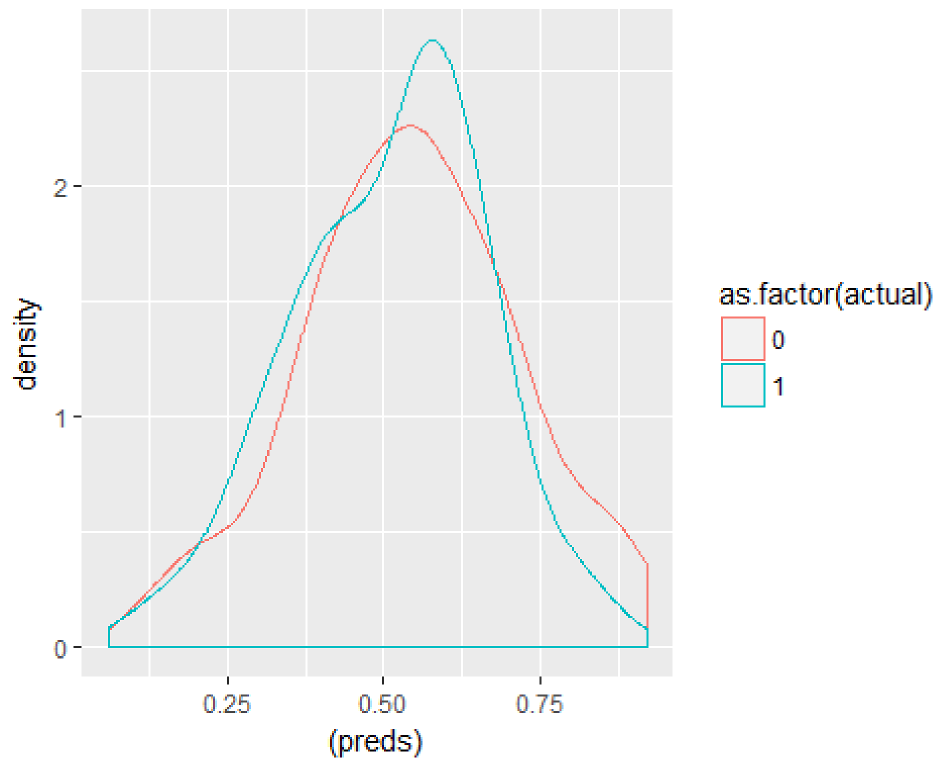


XGBoost Model

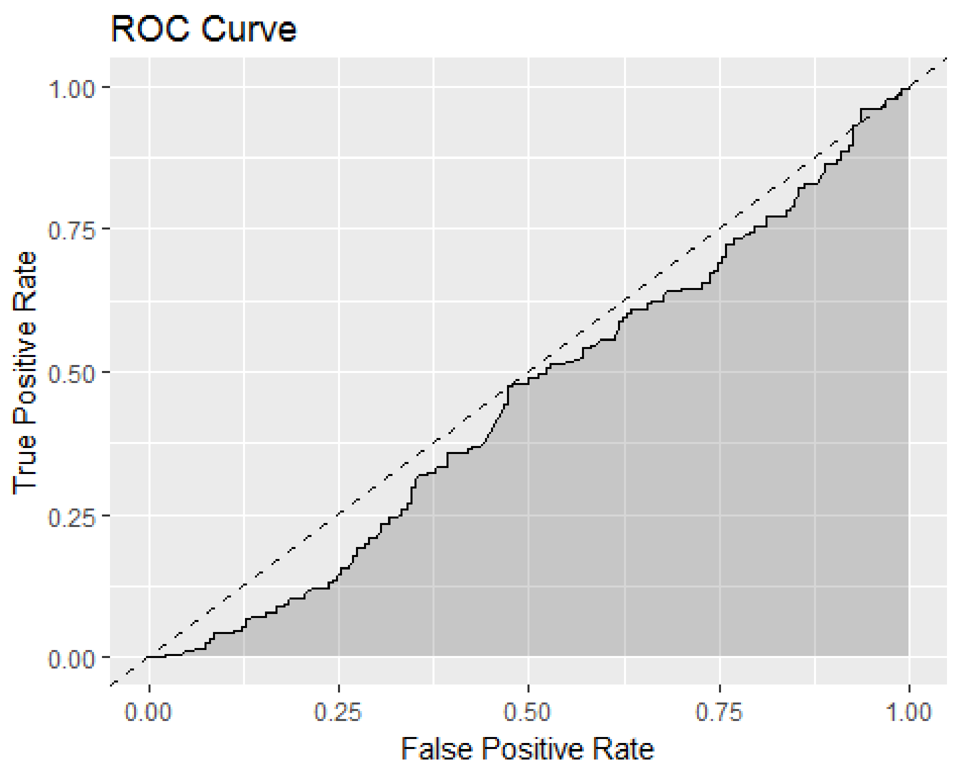
```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)
```

```
preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  71  81
##       1 115 111
#Accuracy
mean(preds_y==ytest)
## [1] 0.4814815
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5185185
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



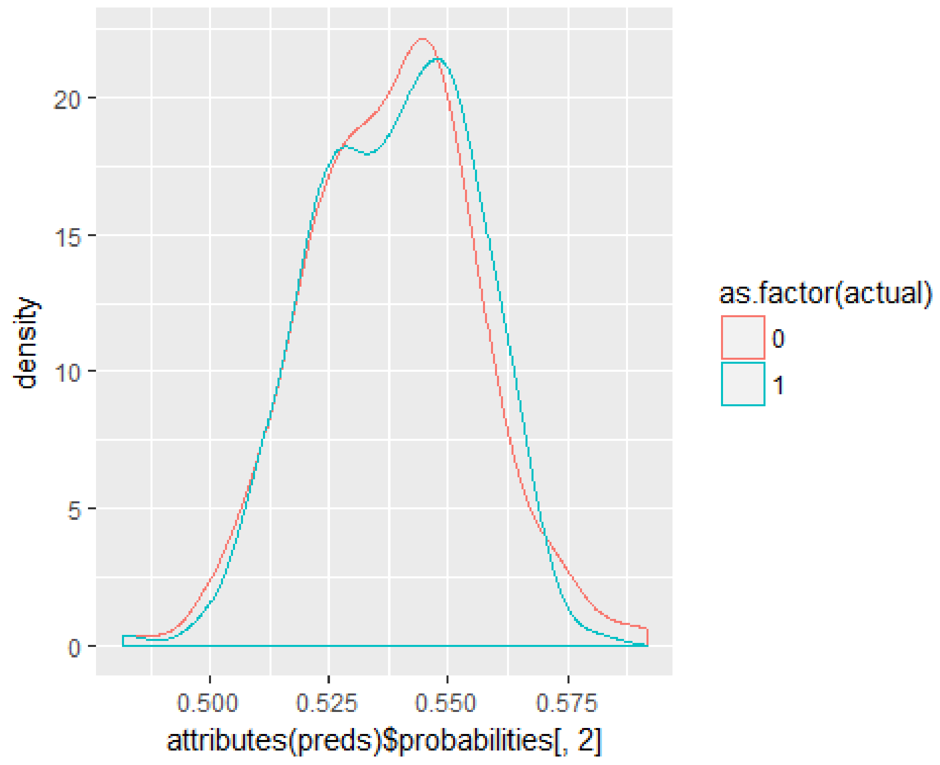
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4510249
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



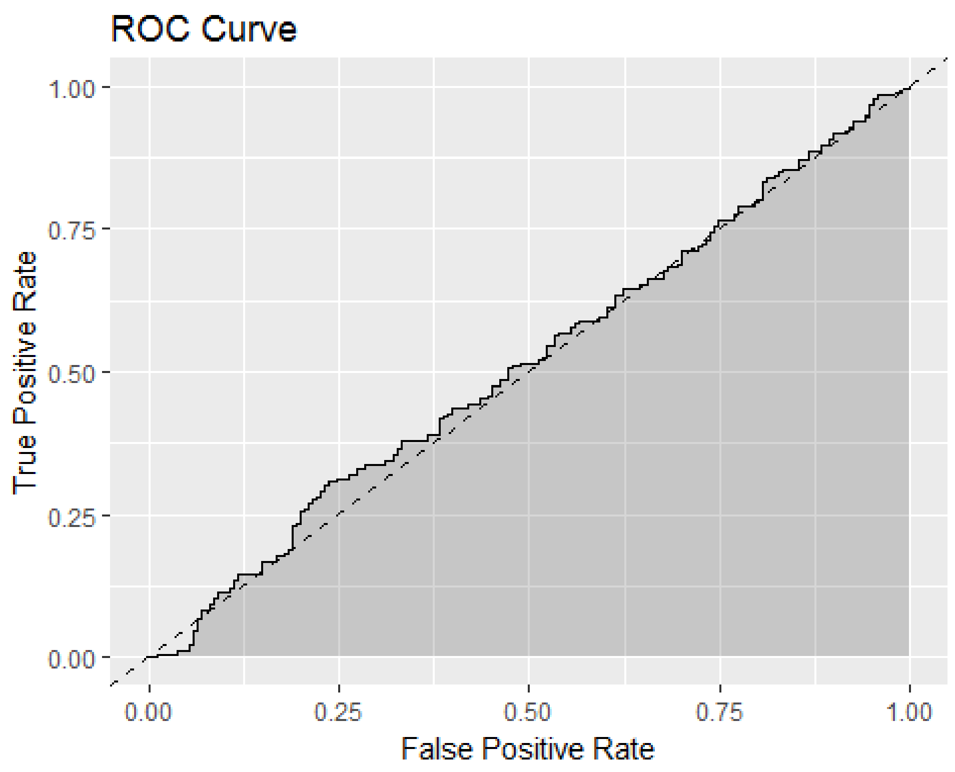
RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'billion.year.ago' and 'blogger.raif.badawi' and
## 'charli.hebdo.attack' and 'charli.hebdo.cartoon' and 'china.stock.plung'
## and 'coal.oil.gas' and 'doctor.border.hospit' and 'down.russian.jet' and
## 'hack.death.bangladesh' and 'hong.kong.booksel' and 'islam.state.syria'
## and 'lee.kuan.yew' and 'panama.paper.leak' and 'panama.paper.reveal'
## and 'pari.terror.attack' and 'russian.air.forc' and 'russian.air.strike'
## and 'sale.saudi.arabia' and 'saudi.arabia.accus' and 'sea.level.rise'
## and 'tpp.trade.deal' and 'turkey.down.russian' and 'zika.virus.spread'
## constant. Cannot scale data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  78  84
##       1 108 108
#Accuracy
mean(preds_y==ytest)
## [1] 0.4920635
#Misclassification rate
mean(preds_y!=ytest)
```

```
## [1] 0.5079365
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

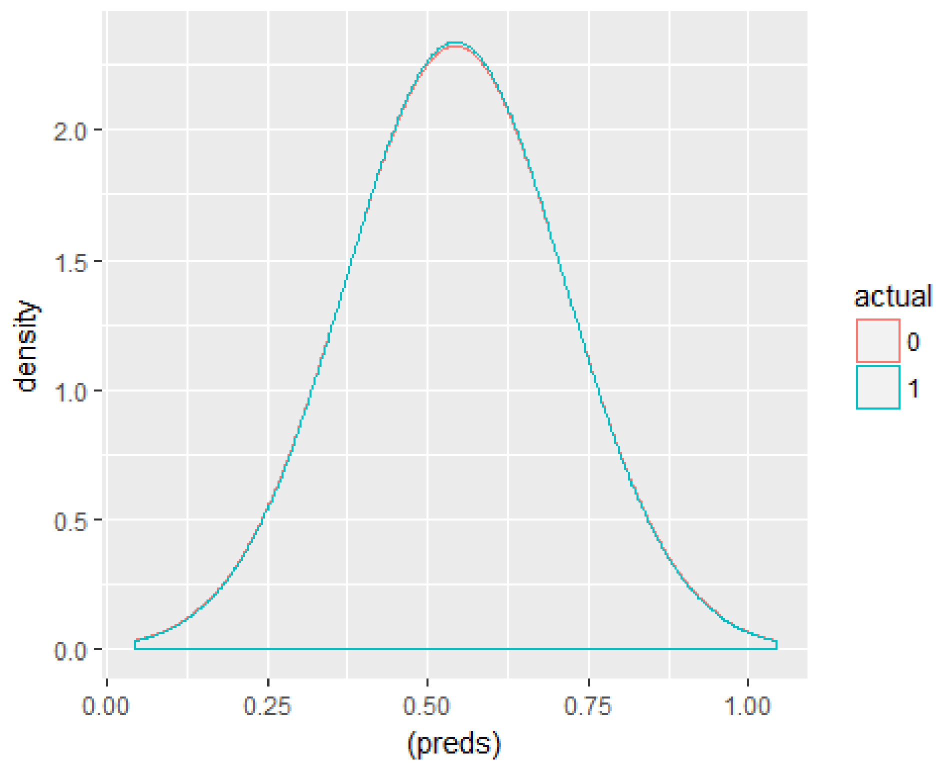


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.514925
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

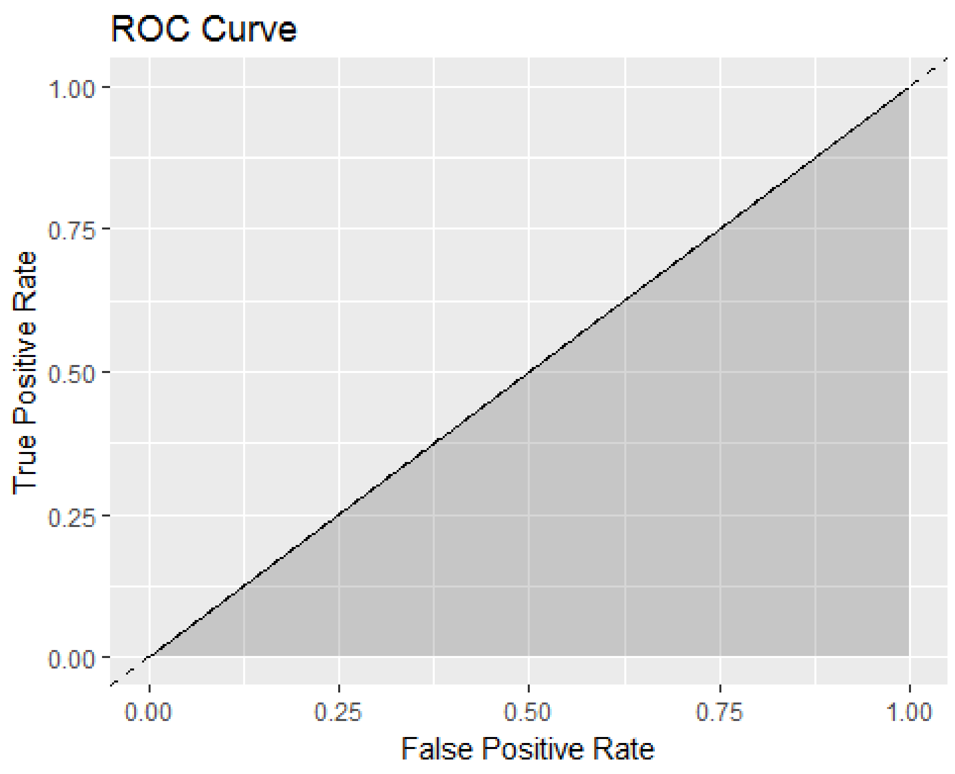


N3G4
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

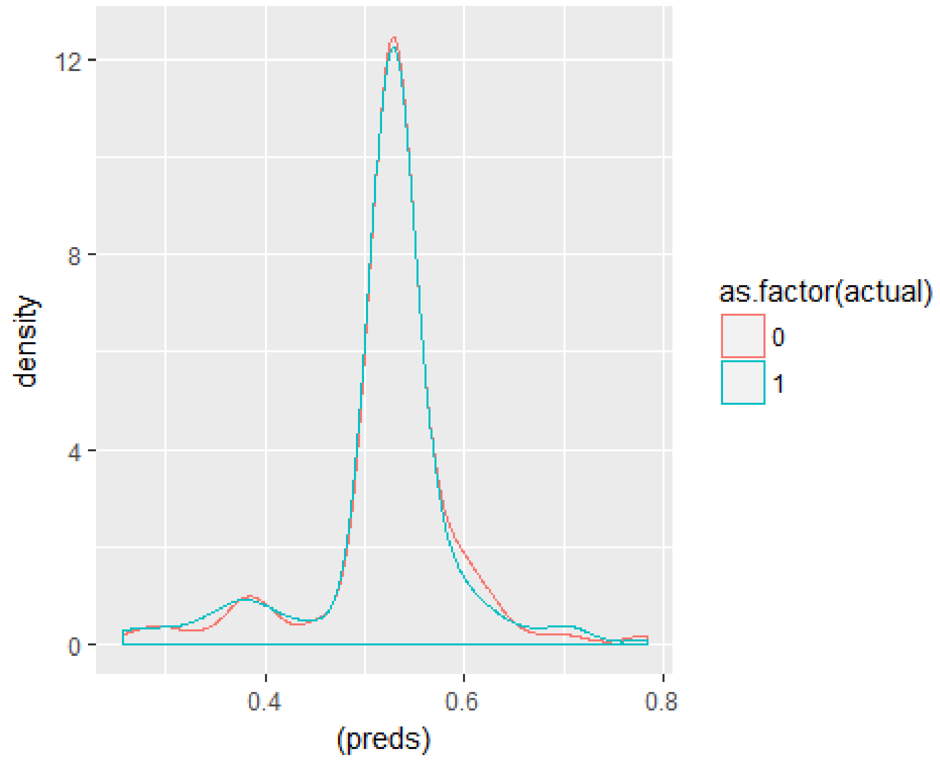


XGBoost Model

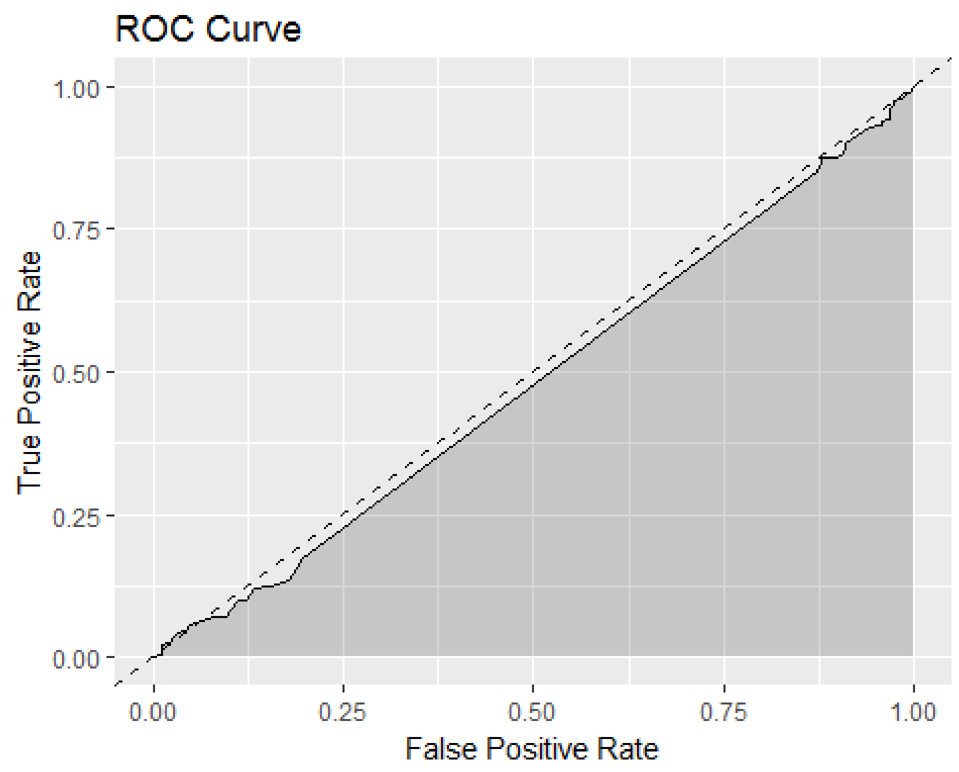
```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)
```

```
preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  24  28
##       1 162 164
#Accuracy
mean(preds_y==ytest)
## [1] 0.4973545
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5026455
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

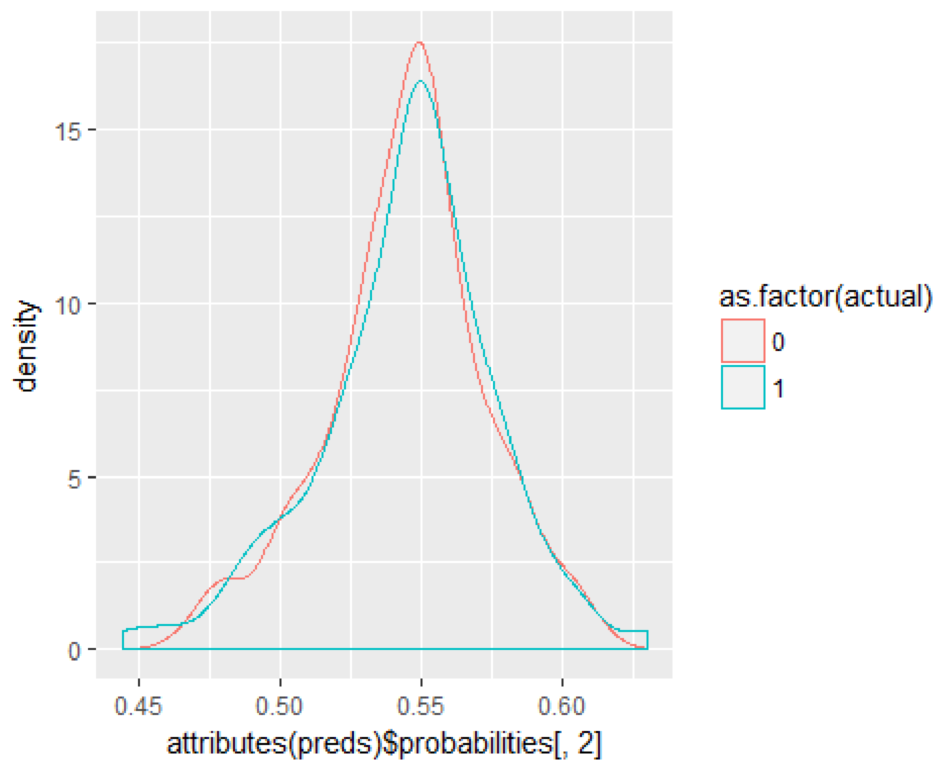


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4810708
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

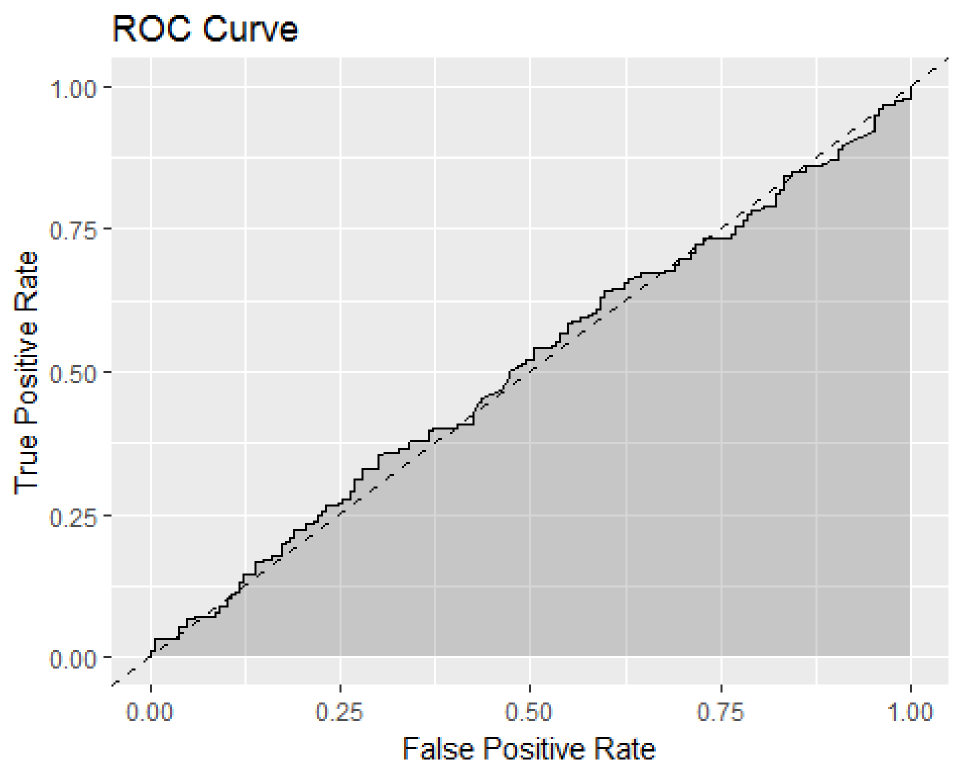


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'X129.billion.arm.sale' and 'X135.tv.film.radio'
## and 'X15billion.saudi.arm.deal' and 'X2015.hottest.year.record'
## and 'X2018.world.cup.russia' and 'X260000.count.accessori.murder'
## and 'X5.year.ahead.schedul' and 'X76.politician.43.music' and
## 'accus.coverup.heavili.redact' and 'accus.wildlif.traffick.anim'
## and 'adolf.hitler.mein.kampf' and 'ahead.pari.climat.talk'
## and 'aid.north.korea.worst' and 'air.strike.syria.kill' and
## 'ali.moham.alnimr.crucifixion' and 'annual.militari.exercis.south'
## and 'approv.129.billion.arm' and 'arabia.human.right.council' and
## 'argentina.presid.cristina.fernandez' and 'arm.sale.saudi.arabia'
## and 'ayatollah.ali.khamenei.wednesday' and 'ban.smoke.car.children'
## and 'bbc.news.migrant.crisi' and 'big.tobacco.accus.coverup' and
## 'blogger.hack.death.bangladesh' and 'blogger.raif.badawi.sentenc'
## and 'bodi.immun.system.attack' and 'boko.haram.attack.baga' and
## 'boko.haram.attack.kill' and 'bomb.doctor.border.hospit' and
## 'boycott.2018.world.cup' and 'brazil.descend.chao.olymp' and
## 'brazil.judg.order.whatsapp' and 'break.global.temperatur.record'
## and 'buddhist.templ.accus.wildlif' and 'canada.electron.spi.agenc'
## and 'cartel.take.tortilla.busi' and 'charg.260000.count.accessori'
## and 'chines.tourist.travel.abroad' and 'cia.chief.hack.email' and
## 'client.fossil.fuel.invest' and 'coalit.fight.islam.state' and
## 'commiss.big.tobacco.accus' and 'coral.bleach.event.underway' and
## 'corrupt.global.oil.industri' and 'coverup.heavili.redact.document'
## and 'cuba.state.sponsor.terror' and 'death.threat.german.mps' and
## 'debri.found.reunion.island' and 'deploy.nuclear.weapon.crimea' and
## 'descend.chao.olymp.loom' and 'doctor.border.hospit.afghanistan'
## and 'doesnt.call.punish.homosexu' and 'drug.cartel.take.tortilla'
## and 'end.fossil.fuel.subsidi' and 'european.commiss.big.tobacco'
## and 'exxon.knew.climat.chang' and 'fast.food.giant.mcdonald'
## and 'femal.suicid.bomber.kill' and 'fifa.presid.sepp.blatter'
## and 'fight.islam.state.milit' and 'fight.islam.state.syria' and
## 'foreign.minist.juli.bishop' and 'fossil.fuel.compani.econom' and
## 'found.dead.sign.tortur' and 'freedom.declin.era.propaganda' and
## 'fuel.compani.econom.nonviabl' and 'gay.imam.quran.doesnt' and
## 'gay.iranian.poet.seek' and 'german.antiislam.movement.pegida' and
## 'global.sea.level.rise' and 'global.supermarket.sell.shrimp' and
## 'greec.recogn.palestinian.state' and 'guilti.crimin.charg.manipul'
## and 'guinea.tsunami.warn.issu' and 'hezbollah.list.terror.threat'
## and 'honduran.activist.berta.ccere' and 'hong.kong.booksel.miss' and
## 'human.right.palestinian.territori' and 'imam.quran.doesnt.call' and
## 'impeach.presid.dilma.rousseff' and 'increas.number.asylum.seeker' and
## 'indian.space.research.organis' and 'insult.presid.recep.tayyip' and
## 'integr.minist.sylvi.listhaug' and 'introduc.temporari.border.control'
## and 'iran.hezbollah.list.terror' and 'iraqi.presid.saddam.hussein'
## and 'irish.samesex.marriag.referendum' and 'isi.commit.genocid.iraq'
## and 'islam.state.execut.chines' and 'islam.state.mustard.gas'
## and 'islamist.group.boko.haram' and 'japanes.offici.charg.tpp'
## and 'join.intern.crimin.court' and 'kidnap.murder.13.young'
## and 'kill.200.isi.milit' and 'kill.drone.strike.afghanistan'
## and 'kim.jongun.snub.china' and 'knew.osama.bin.laden' and
## 'korea.fire.missil.sea' and 'korea.miniatur.nuclear.warhead'
## and 'kurd.invit.antiisi.confer' and 'kurdistan.worker.parti.pkk'
## and 'law.firm.mossack.fonseca' and 'limit.global.warm.2c' and
## 'manipul.foreign.exchang.rate' and 'microwav.gun.disabl.drone'
## and 'militari.drill.north.korea' and 'minist.najib.razak.person'
## and 'miss.hong.kong.booksel' and 'montreal.dump.8.billion' and
## 'mosqu.eastern.saudi.arabia' and 'murder.13.young.peopl' and
## 'murder.honduran.activist.berta' and 'muslim.gay.imam.quran'
## and 'muslim.protest.charli.hebdo' and 'najib.razak.person.bank'
## and 'nation.declar.state.emerg' and 'nation.human.right.commiss'
## and 'navi.fire.warn.shot' and 'north.korea.miniatur.nuclear' and
## 'north.korea.plant.landmin' and 'north.korea.restart.plutonium'
## and 'north.korea.worst.drought' and 'nuclear.deal.world.power'
## and 'opposit.leader.bori.nemtsov' and 'osama.bin.laden.hide'
## and 'panama.paper.law.firm' and 'papua.guinea.tsunami.warn' and
## 'pari.attack.identifi.french' and 'peopl.kill.bomb.blast' and
## 'plan.establish.militari.base' and 'plead.guilti.crimin.charg'
## and 'presid.barack.obama.dismiss' and 'press.freedom.declin.era'
## and 'prime.minist.justin.trudeau' and 'prime.minist.najib.razak'
## and 'quran.doesnt.call.punish' and 'razak.person.bank.account'
## and 'relat.kill.drone.strike' and 'remov.cuba.state.sponsor' and
## 'remov.iran.hezbollah.list' and 'retak.mosul.islam.state' and
## 'run.complet.renew.energi' and 'russian.air.strike.syria' and
## 'russian.opposit.leader.bori' and 'saudi.arabia.behead.indonesian'
## and 'saudi.arabia.human.right' and 'saudi.blogger.raif.badawi' and
## 'saudi.comedian.death.threat' and 'sell.shrimp.peel.slave' and 'sentenc.
## 10.year.prison' and 'sentenc.men.520.year' and 'sewag.st.lawrenc.river'
## and 'show.solidar.charli.hebdo' and 'slaughter.15.million.armenian'
## and 'space.research.organis.isro' and 'spacex.falcon.9.rocket'
## and 'state.iraq.syria.isi' and 'state.sponsor.terror.list' and
## 'supermarket.give.unsold.food' and 'supermarket.sell.shrimp.peel'
## and 'svalbard.global.seed.vault' and 'take.tortilla.busi.mexico'
## and 'templ.accus.wildlif.traffick' and 'thai.buddhist.templ.accus'
## and 'tiger.cub.bodi.found' and 'tobacco.accus.coverup.heavili'
## and 'top.japanes.offici.charg' and 'trade.invest.partnership.ttip'
## and 'transatlant.trade.invest.partnership' and
## 'transpacif.partnership.tpp.trade' and 'transpacif.partnership.trade.deal'
## and 'ttip.free.trade.deal' and 'tune.100.billion.euro' and
## 'turkey.down.russian.jet' and 'turkish.polic.tear.gas' and
## 'uk.chancellor.georg.osborn' and 'uk.corpor.tax.2014' and
## 'uk.home.secretari.theresa' and 'uk.vote.leav.european' and
## 'vote.leav.european.union' and 'watchdog.press.freedom.declin'
## and 'wildlif.traffick.anim.abus' and 'wind.farm.produc.electr' and
## 'work.protect.sea.turtl' and 'world.largest.radio.telescop' and
## 'yangtz.giant.softshel.turtl' and 'yulin.dog.meat.festiv' constant. Cannot
## scale data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0 108 116
##       1  78  76
#Accuracy
mean(preds_y==ytest)
## [1] 0.4867725
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5132275
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

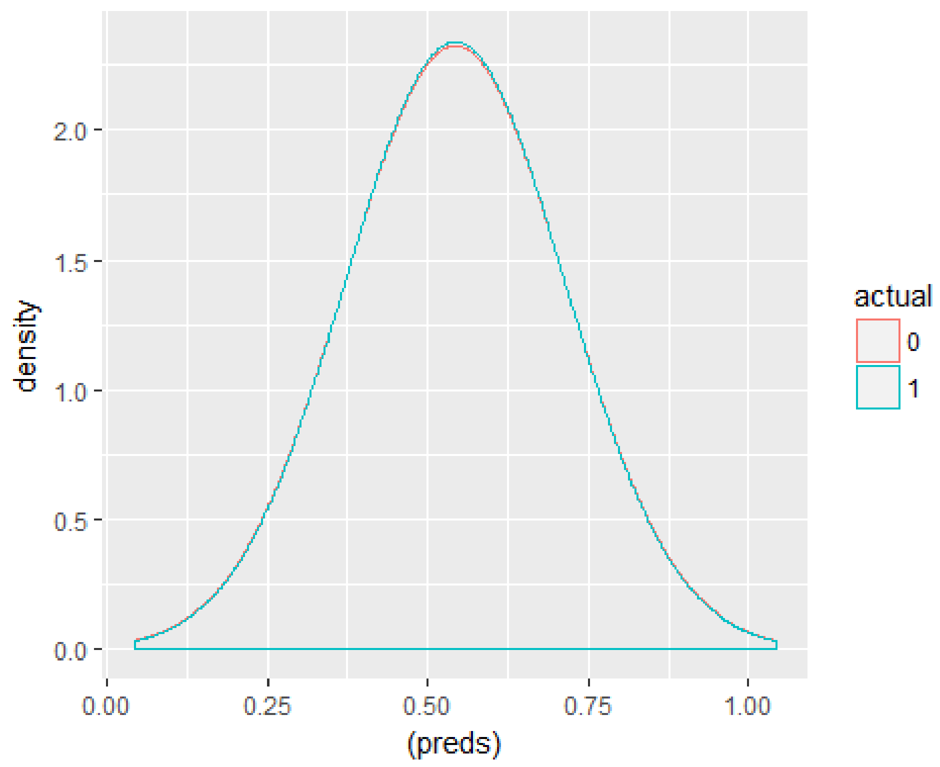


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5090026
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

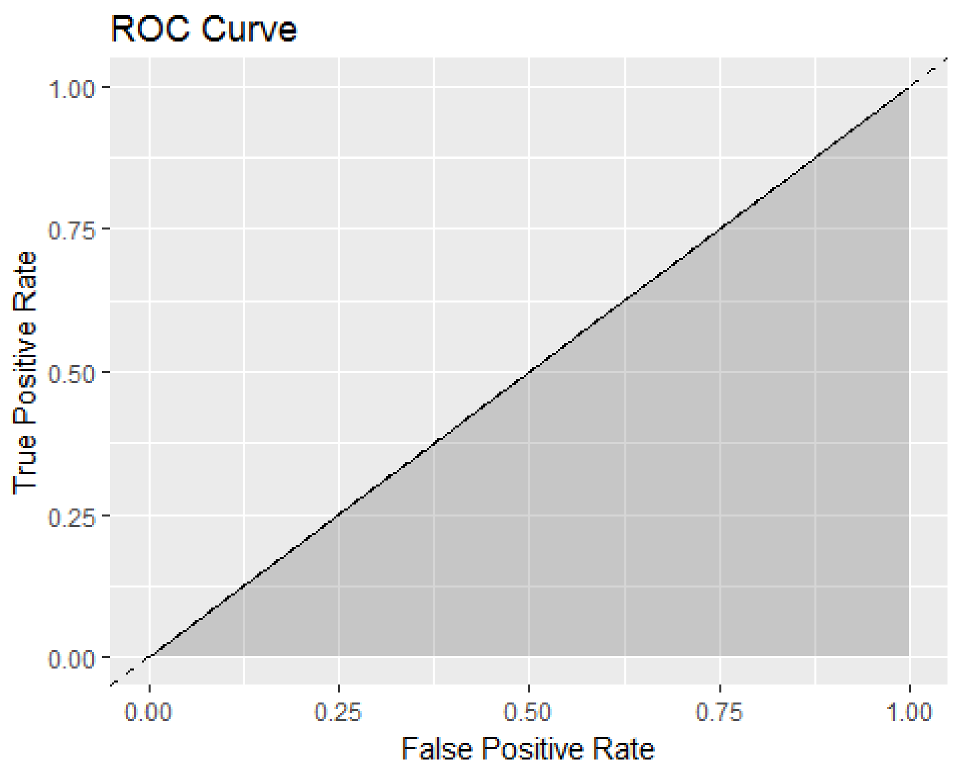


N3G23
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



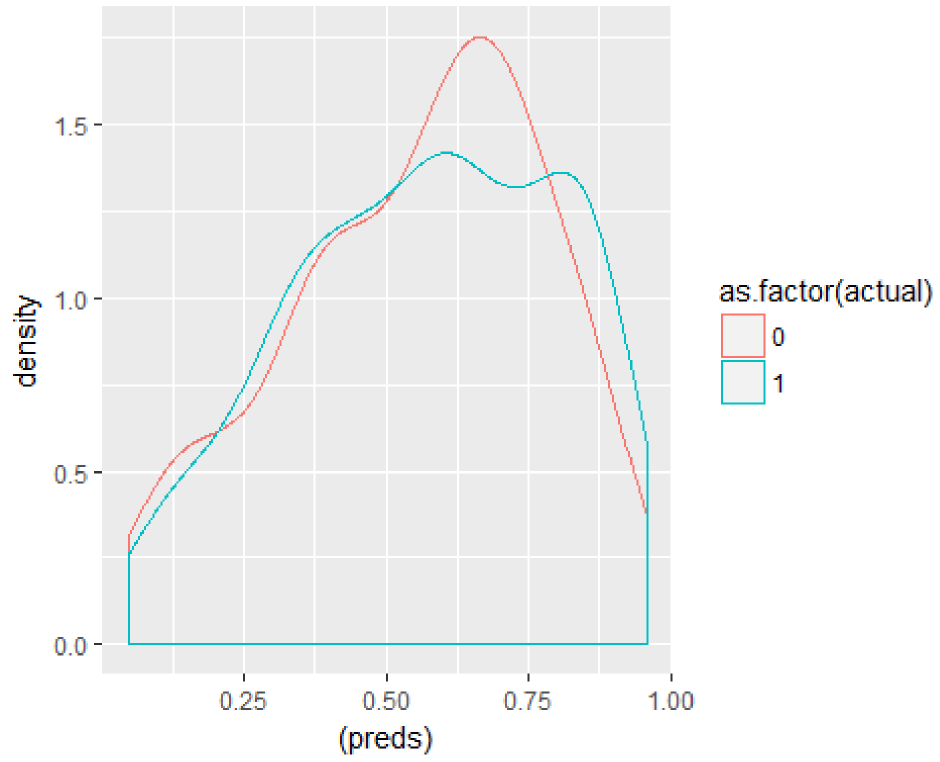
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



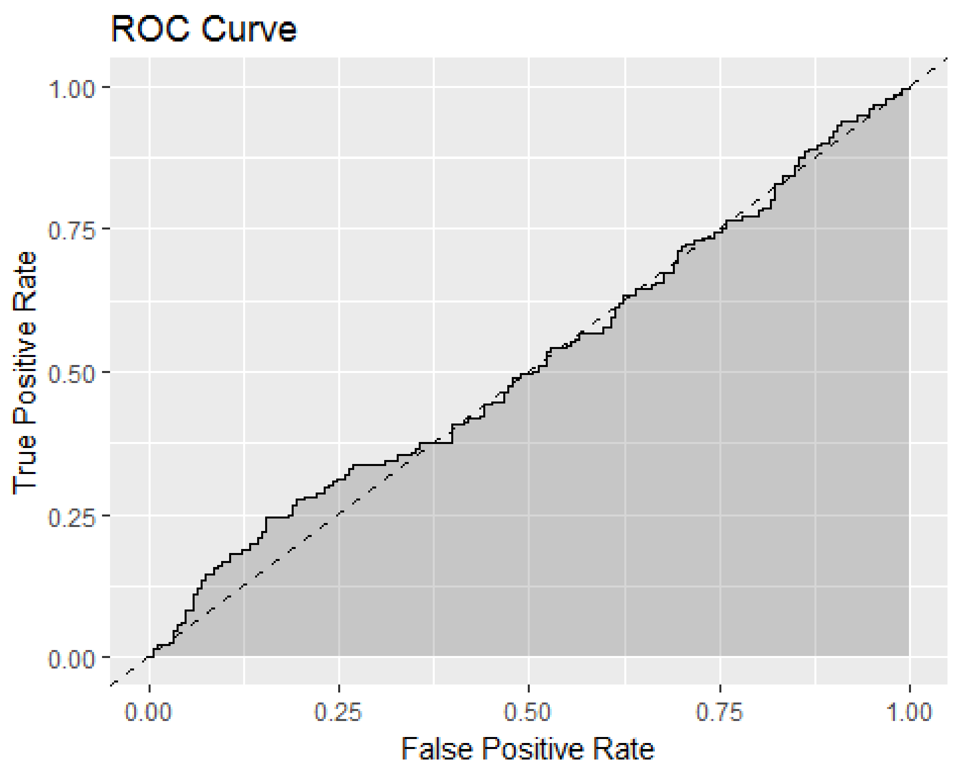
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  70  73
##       1 116 119
#Accuracy
mean(preds_y==ytest)
## [1] 0.5
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

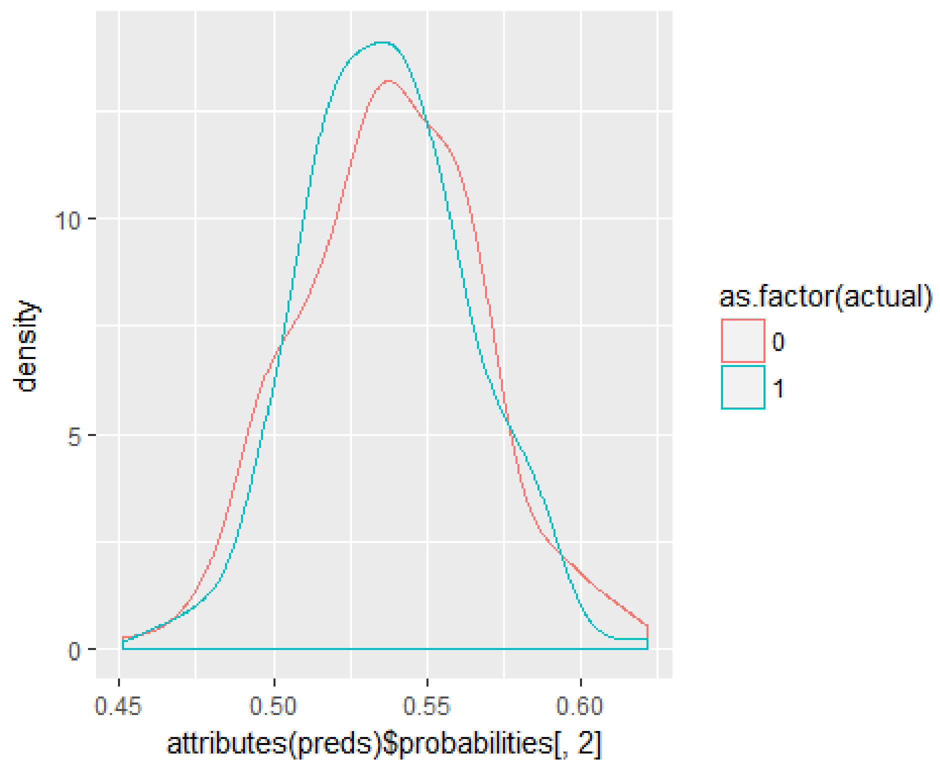


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5166891
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

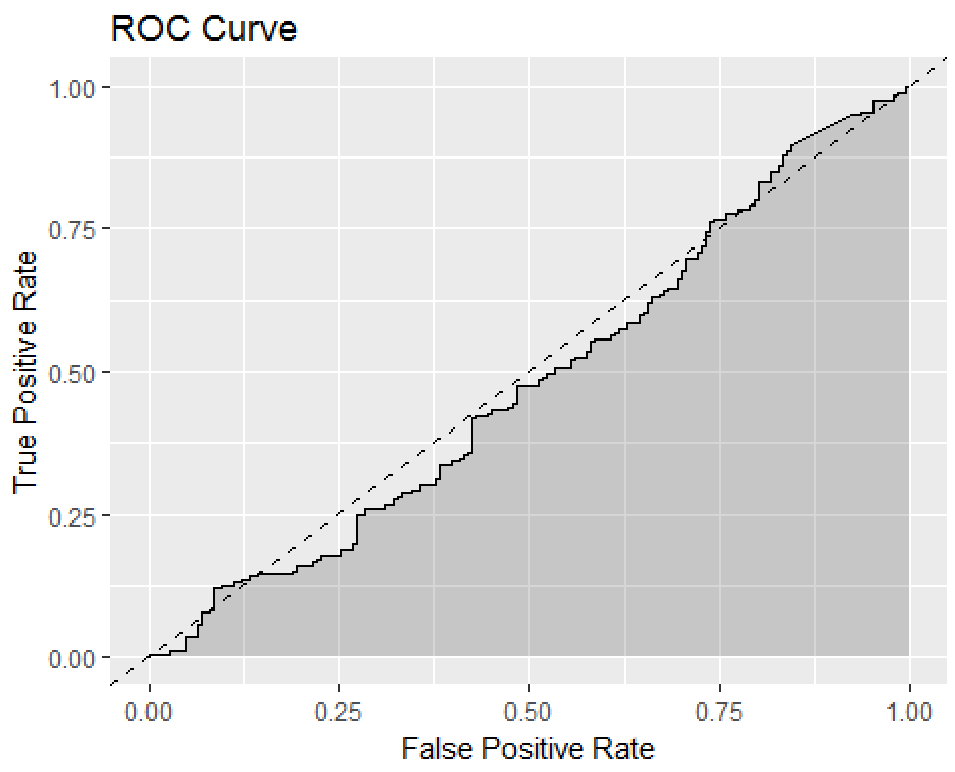


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  79  70
##       1 107 122
#Accuracy
mean(preds_y==ytest)
## [1] 0.531746
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.468254
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

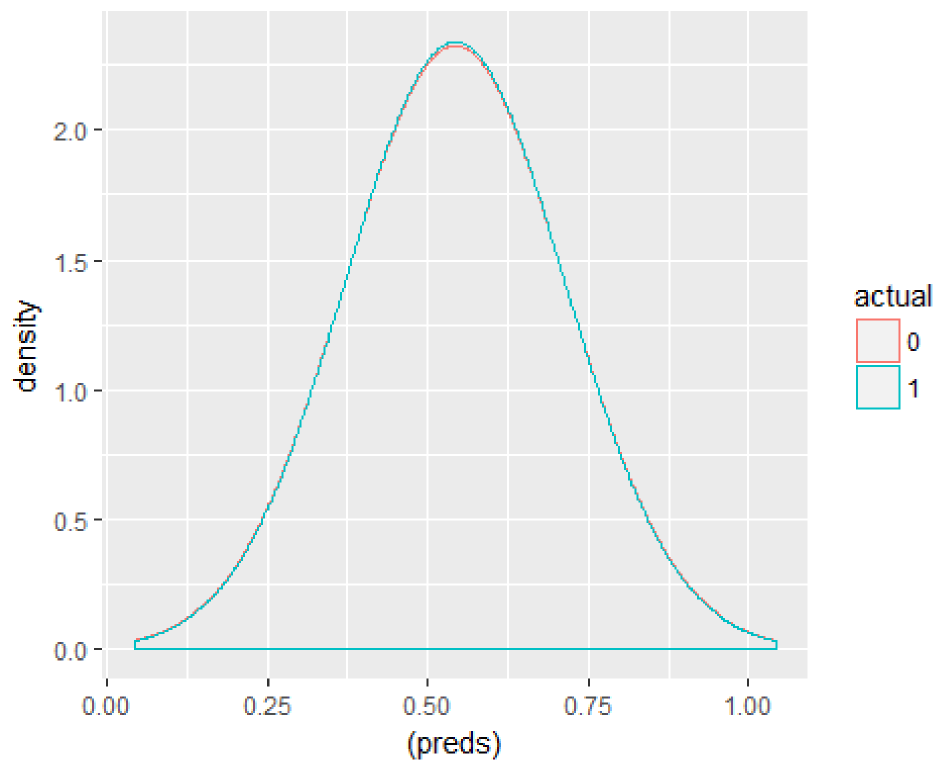


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4812668
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```

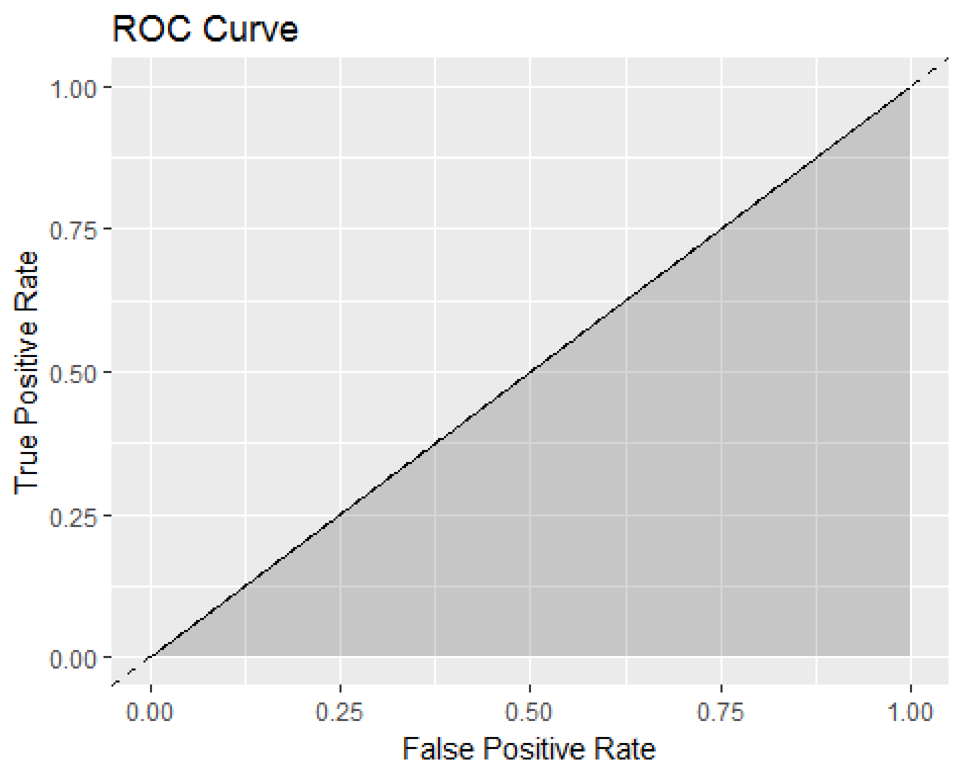


N3G123
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



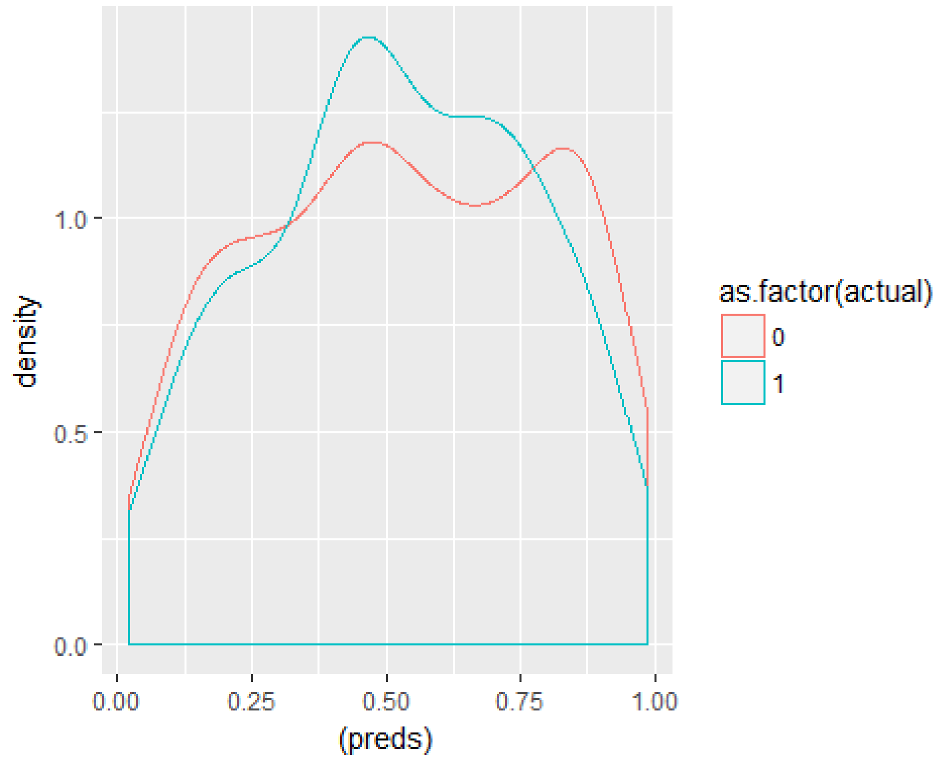
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



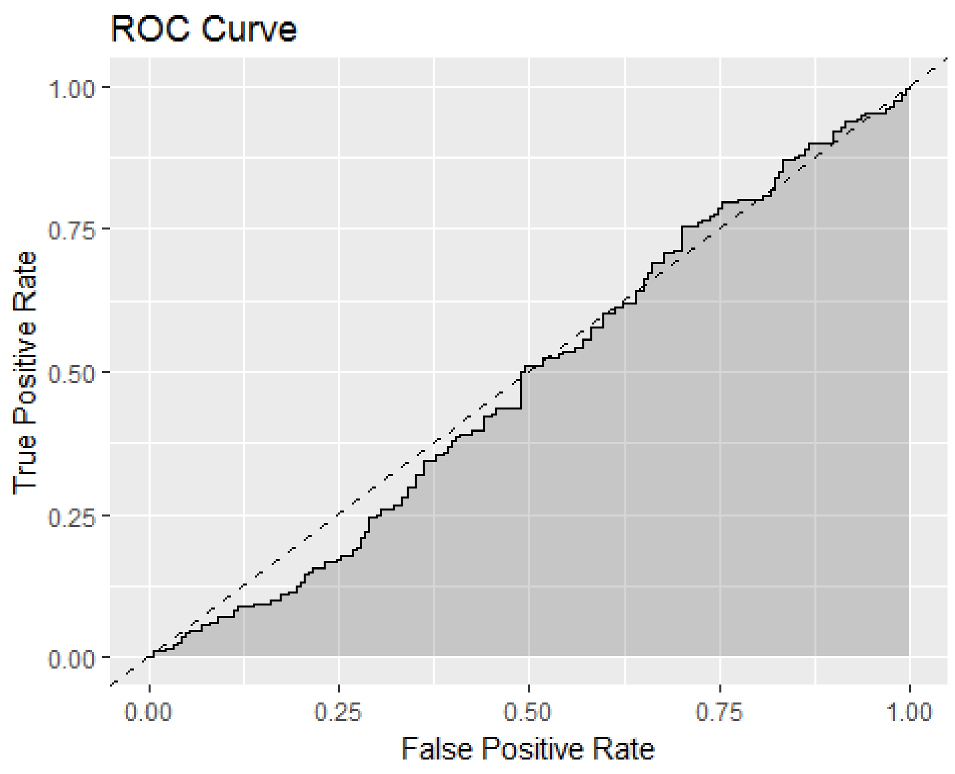
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  89  91
##       1  97 101
#Accuracy
mean(preds_y==ytest)
## [1] 0.5026455
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4973545
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

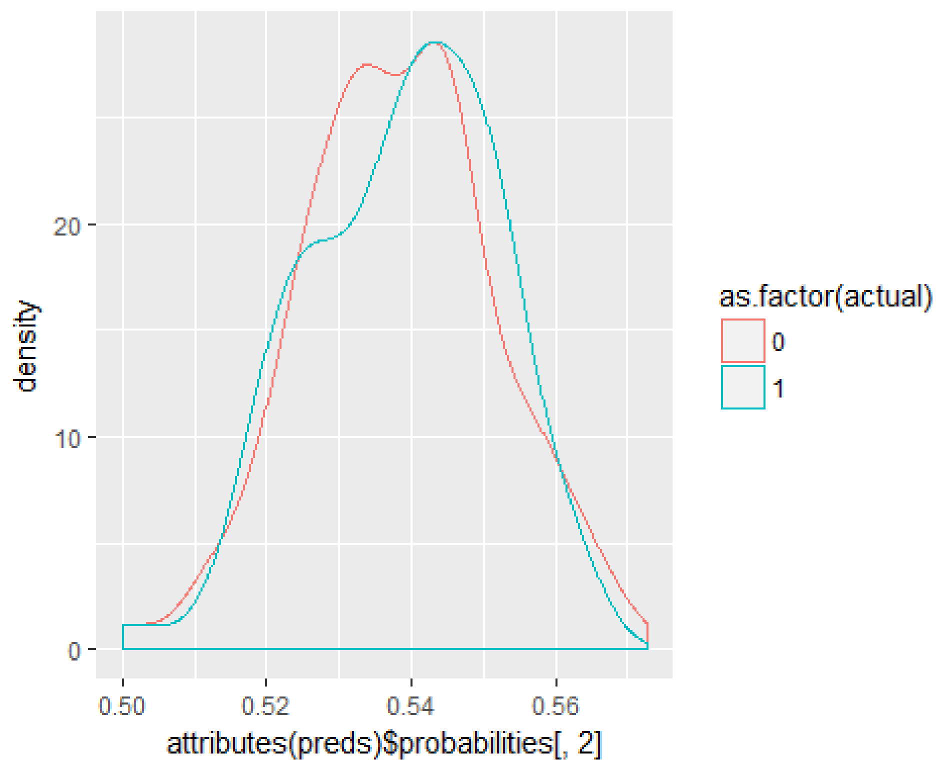


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.484263
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

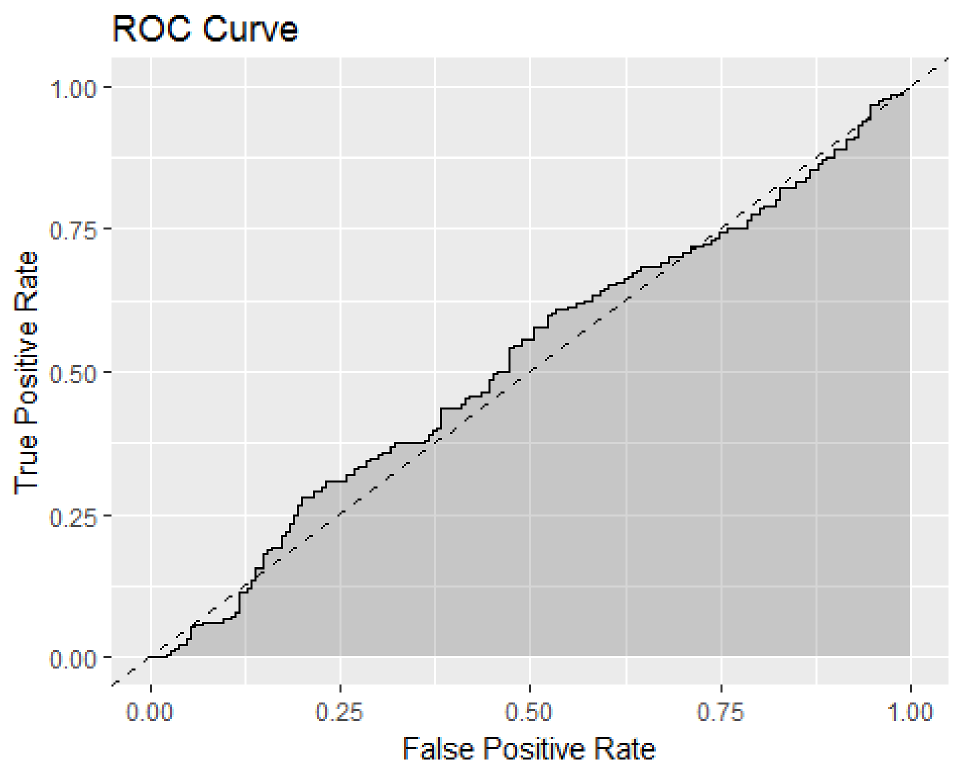


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05,
## probability = T): Variable(s) 'X41.billion' and 'abdeslam' and 'adblock'
## and 'alberto.nisman' and 'alnimr' and 'amo.yee' and 'antidepress'
## and 'arabia.accus' and 'assyrian.christian' and 'astronom.discov' and
## 'attack.pari' and 'badawi' and 'bataclan' and 'berta' and 'berta.ccere'
## and 'bill.c51' and 'billion.year.ago' and 'bleach.event' and 'blogger.raif'
## and 'blogger.raif.badawi' and 'booksel' and 'border.hospit' and
## 'bori.nemtsov' and 'brexit' and 'c51' and 'calai.jungl' and 'call.isi'
## and 'ccere' and 'cecil' and 'cecil.lion' and 'cere' and 'charli.hebdo'
## and 'charli.hebdo.attack' and 'charli.hebdo.cartoon' and 'china.stock'
## and 'china.stock.plung' and 'citi.palmyra' and 'coal.oil' and
## 'coal.oil.gas' and 'coral.bleach' and 'corbyn' and 'crucifixion' and
## 'daraprim' and 'death.bangladesh' and 'deir' and 'doctor.border.hospit'
## and 'down.russian.jet' and 'dutert' and 'egyptair' and 'el.nino' and
## 'emiss.cheat' and 'emiss.scandal' and 'euro.2016' and 'europ.news'
## and 'falcon.9' and 'fernandez' and 'fifa.offici' and 'fonseca' and
## 'forc.fight' and 'fort.mcmurray' and 'germani.stop' and 'germanw' and
## 'hack.death.bangladesh' and 'hebdo' and 'hebdo.attack' and 'hebdo.cartoon'
## and 'honduran.activist' and 'hong.kong.booksel' and 'hunt.trophi' and
## 'impeach.vote' and 'indian.space' and 'insult.erdogan' and 'isi.kill' and
## 'isi.suspect' and 'isisheld' and 'islam.scholar' and 'islam.state.syria'
## and 'jeremi.corbyn' and 'jihadi.john' and 'king.salman' and 'kong.booksel'
## and 'kuan' and 'kuan.yew' and 'leav.eu' and 'lee.kuan' and 'lee.kuan.yew'
## and 'lion.hunt' and 'mcmurray' and 'microcephali' and 'migrant.back' and
## 'migrant.crisi' and 'migrant.europ' and 'millionyearold' and 'month.record'
## and 'mossack' and 'mossack.fonseca' and 'nation.geograph' and 'nauru' and
## 'nemtsov' and 'nepal.earthquak' and 'nineveh' and 'nisman' and 'nonus' and
## 'pacif.trade' and 'palmyra' and 'panama.paper' and 'panama.paper.leak' and
## 'panama.paper.reveal' and 'paper.leak' and 'pari.climat' and 'pari.terror'
## and 'pari.terror.attack' and 'pariti' and 'pegida' and 'plan.declar'
## and 'plastic.garbag' and 'pod' and 'radio.telescop' and 'raif' and
## 'raif.badawi' and 'ramadi' and 'refuge.arriv' and 'refuge.germani'
## and 'research.suggest' and 'reveal.saudi' and 'russian.air' and
## 'russian.air.forc' and 'russian.air.strike' and 'russian.airstrik' and
## 'salah.abdeslam' and 'sale.saudi.arabia' and 'saudi.arabia.accus' and
## 'saudi.blogger' and 'saudi.coalit' and 'saudi.govern' and 'saudil' and
## 'saudil.coalit' and 'sea.level.rise' and 'secular.blogger' and 'snooper'
## and 'snooper.charter' and 'stabber' and 'state.alli' and 'state.syria' and
## 'syria.isi' and 'syrian.kurdish' and 'tianjin' and 'tpp.trade.deal' and
## 'tppa' and 'train.ukrainian' and 'turkey.down' and 'turkey.down.russian'
## and 'turkey.erdogan' and 'unsold.food' and 'varoufaki' and 'vw.emiss'
## and 'war.isi' and 'warmer' and 'weapon.ukrain' and 'whatsapp' and
## 'worth.billion' and 'yee' and 'yew' and 'zika' and 'zika.virus' and
## 'zika.virus.spread' constant. Cannot scale data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  68  73
##       1 118 119
#Accuracy
mean(preds_y==ytest)
## [1] 0.494709
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.505291
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



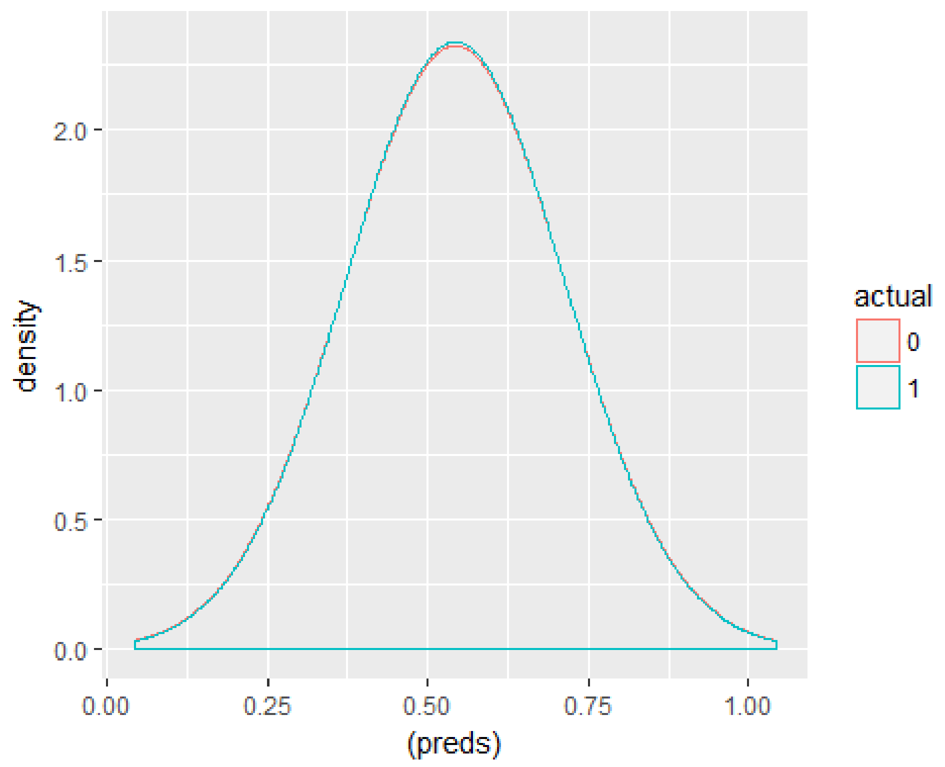
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5193492
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



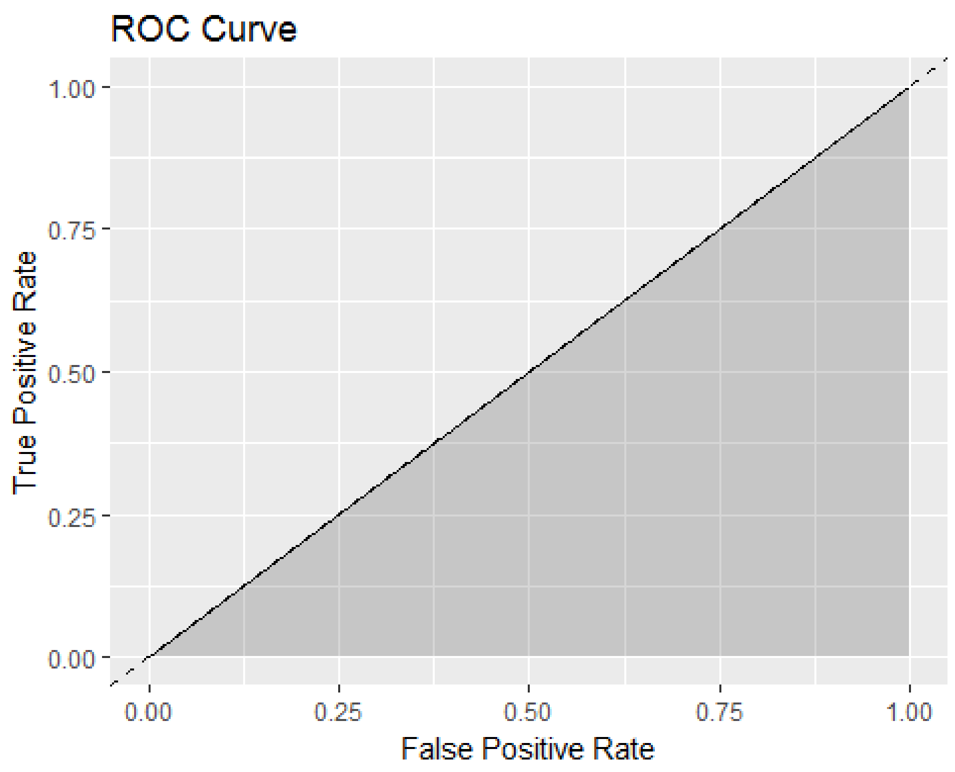
N5G1

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



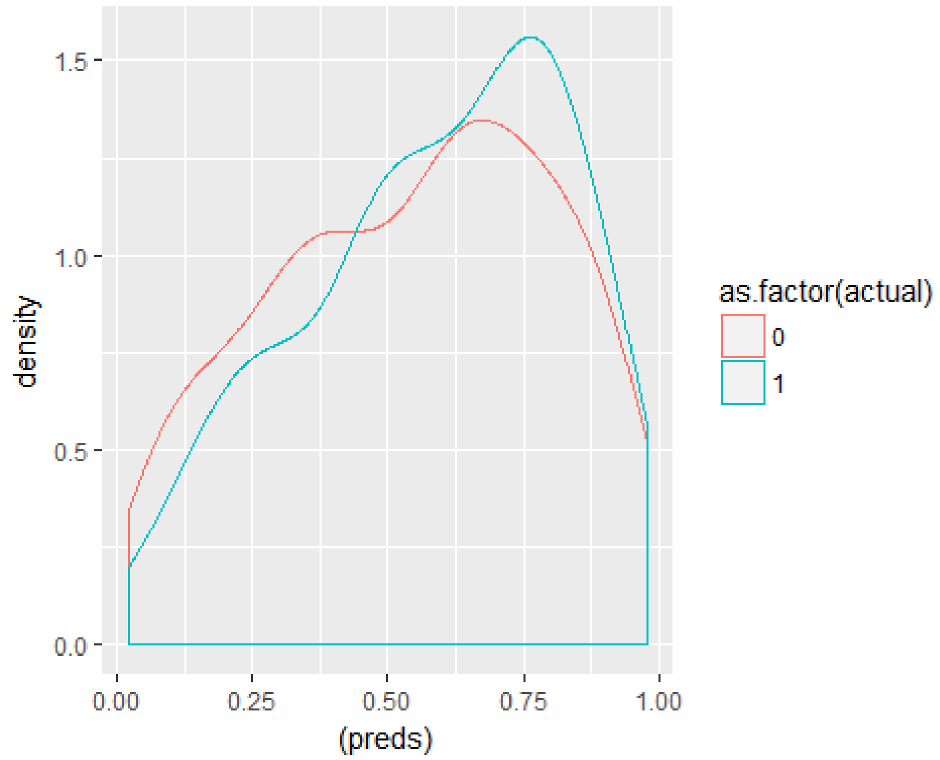
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```



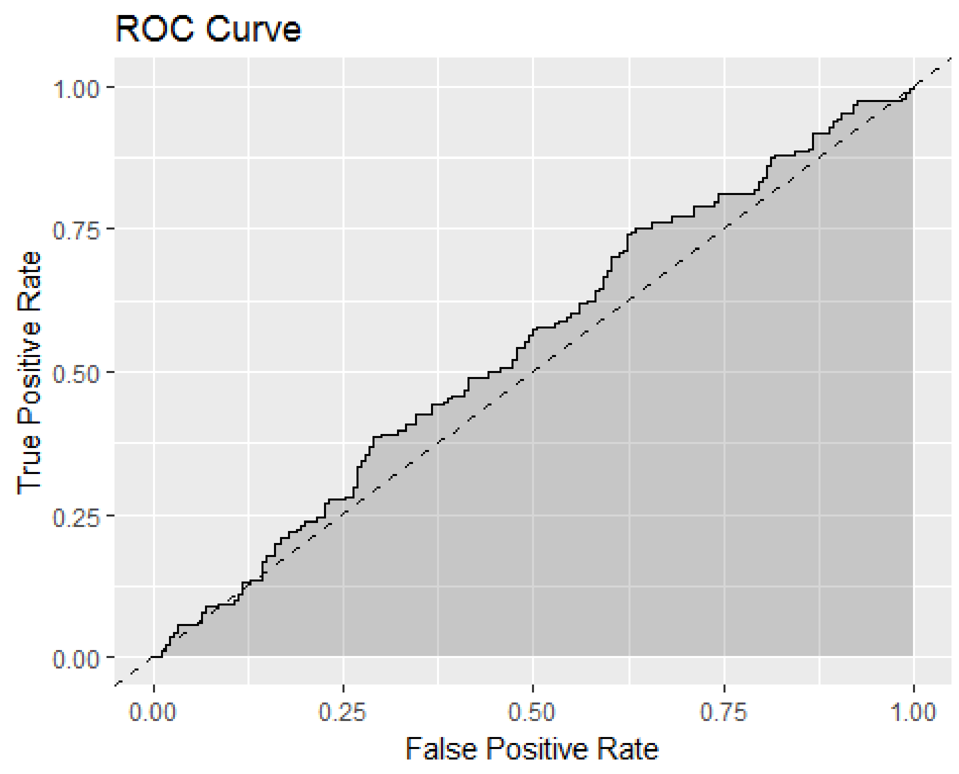
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  77  69
##       1 109 123
#Accuracy
mean(preds_y==ytest)
## [1] 0.5291005
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4708995
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```r
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

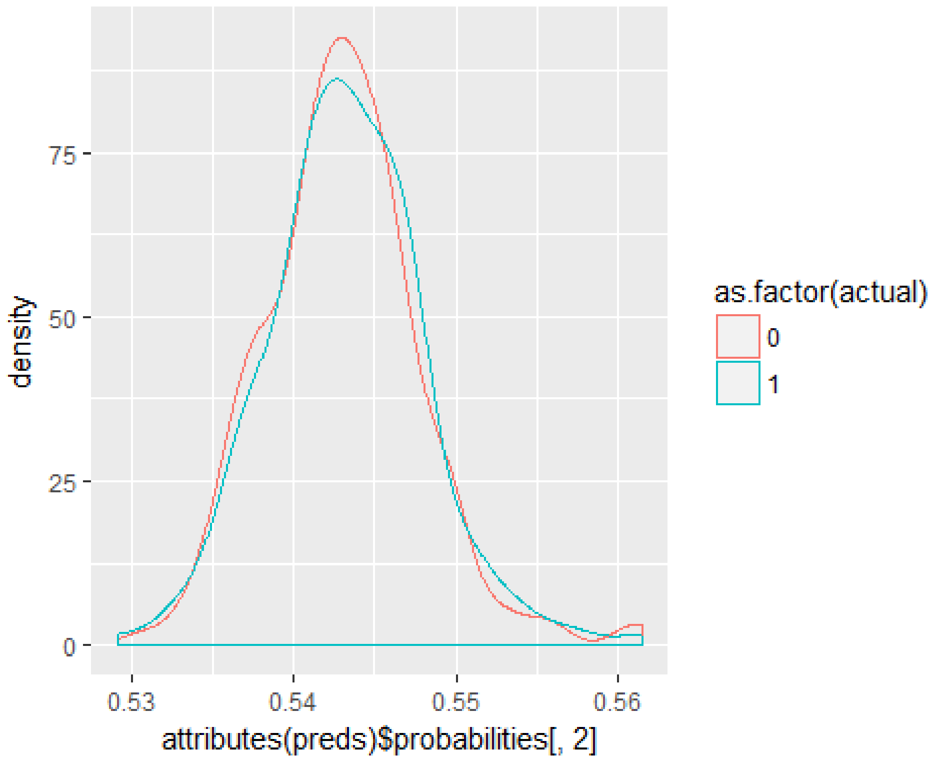


```r
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.546119
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

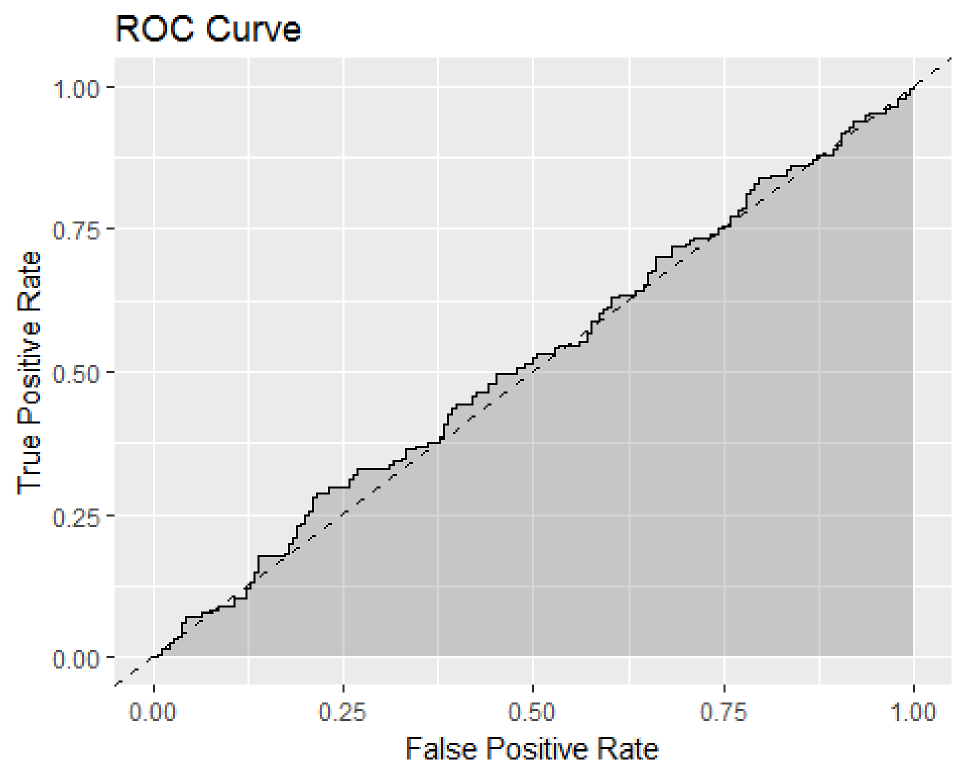


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0 104 105
##       1  82  87
#Accuracy
mean(preds_y==ytest)
## [1] 0.505291
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.494709
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



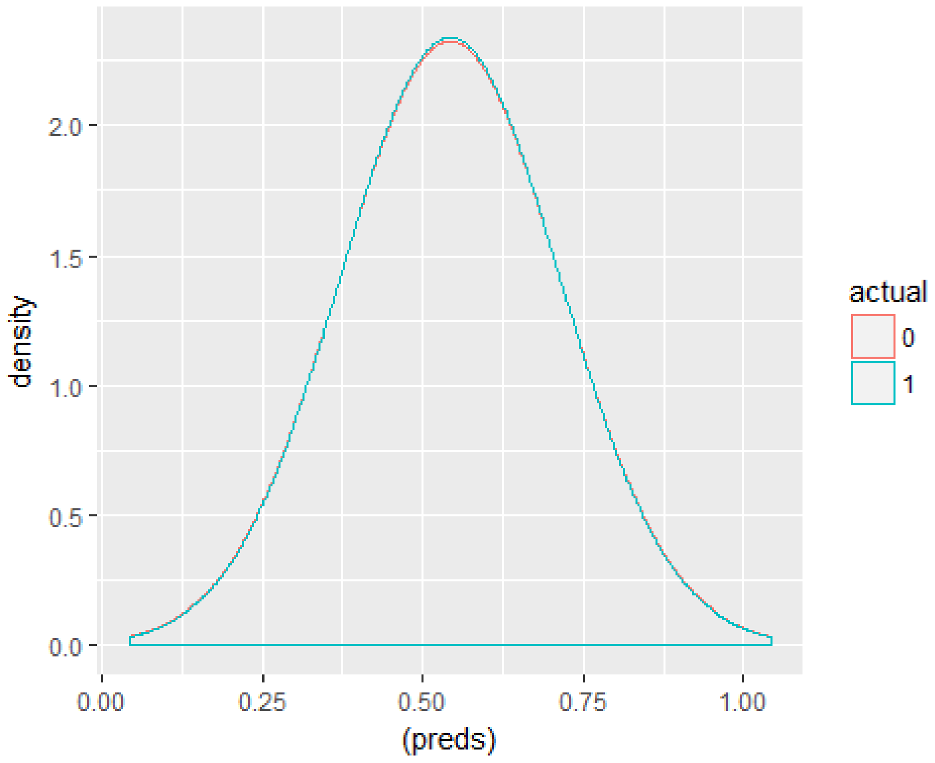
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5172211
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



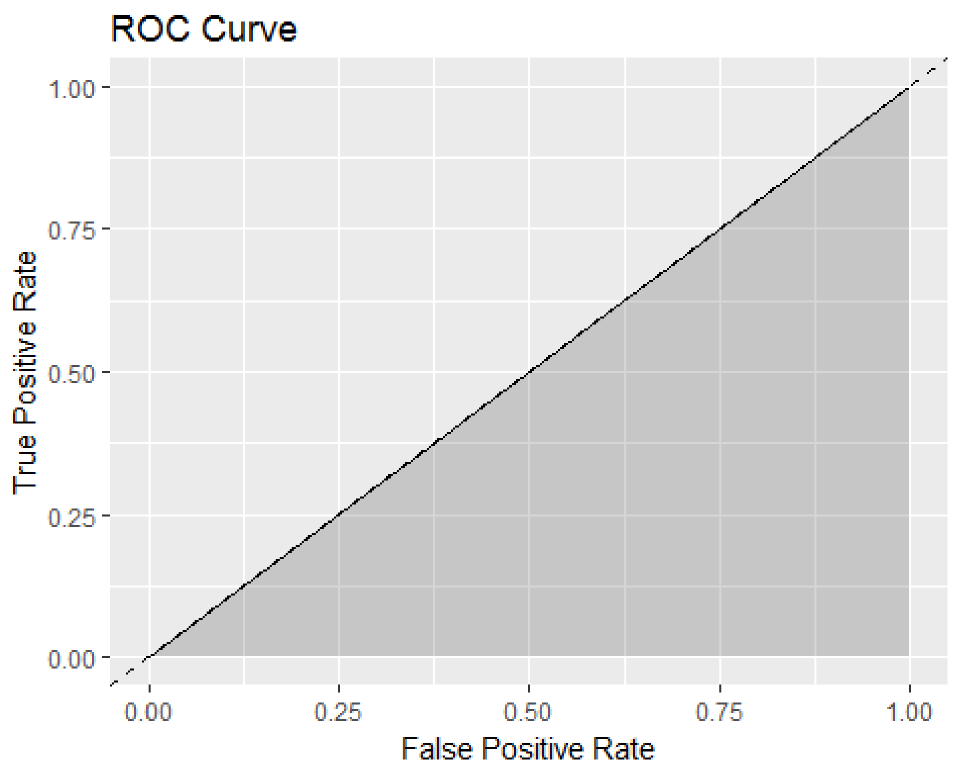
N5G2

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



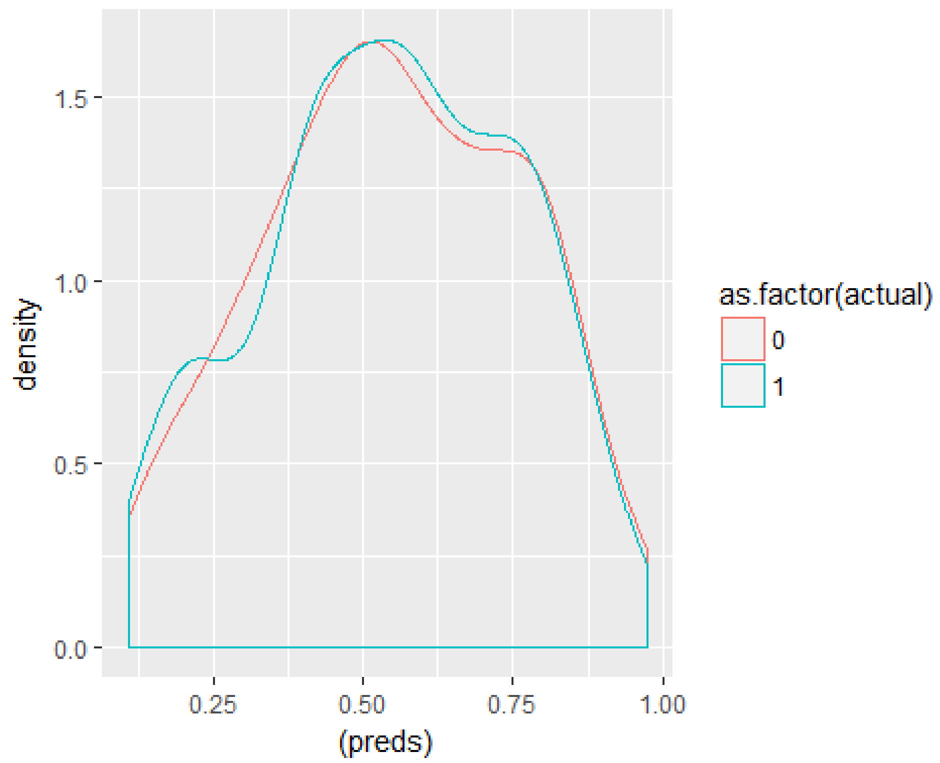
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



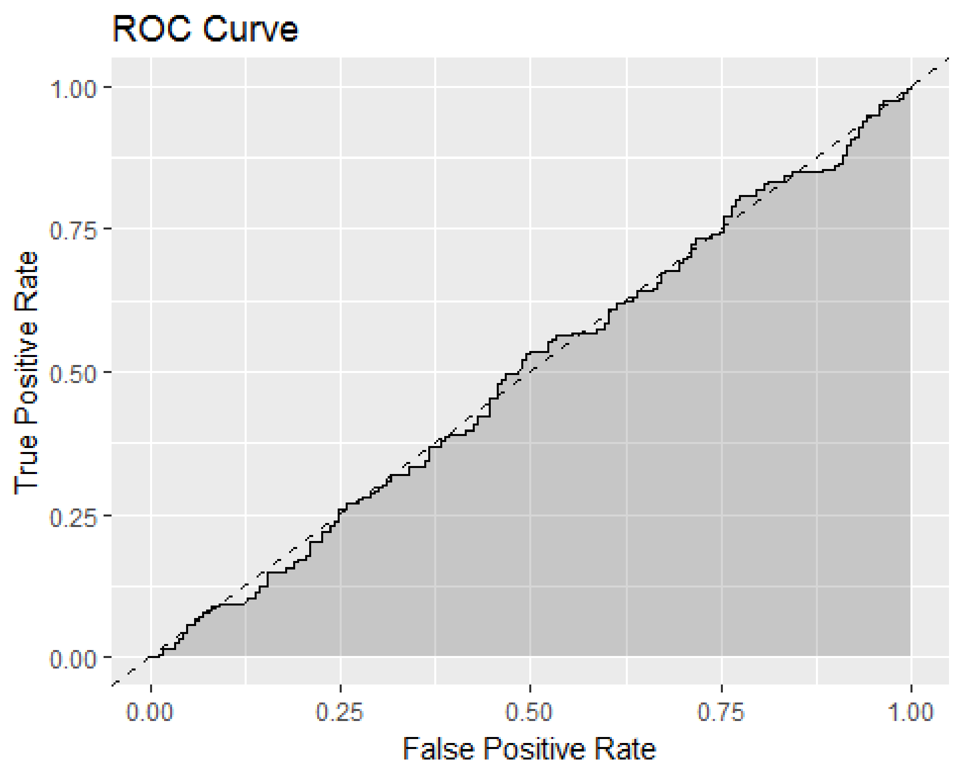
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  74  80
##       1 112 112
#Accuracy
mean(preds_y==ytest)
## [1] 0.4920635
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5079365
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



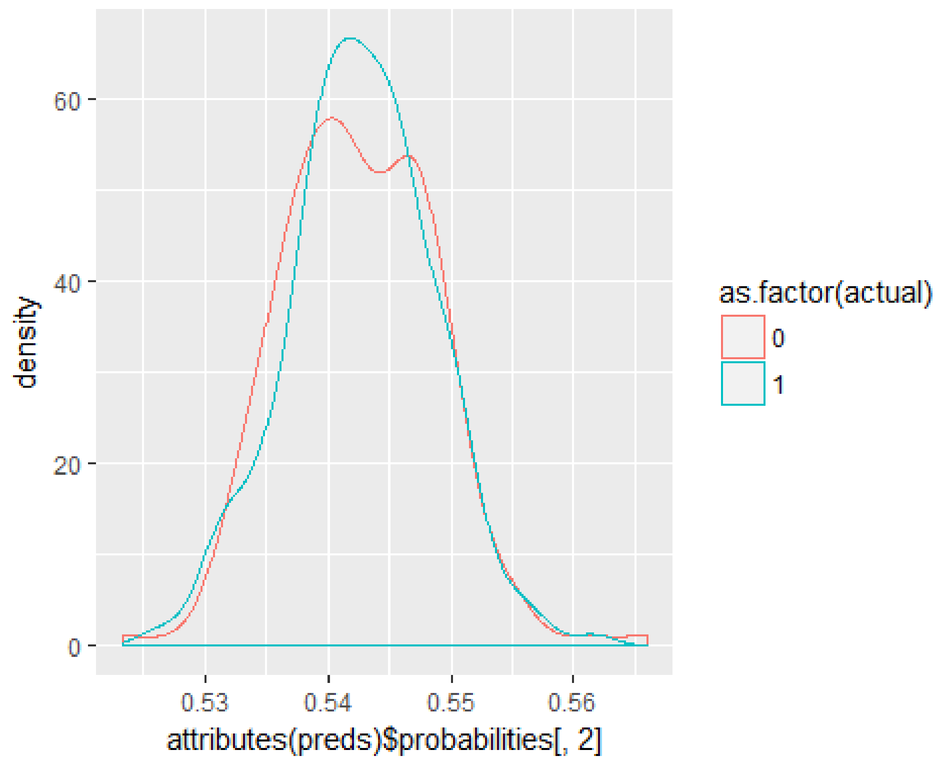
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4959397
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



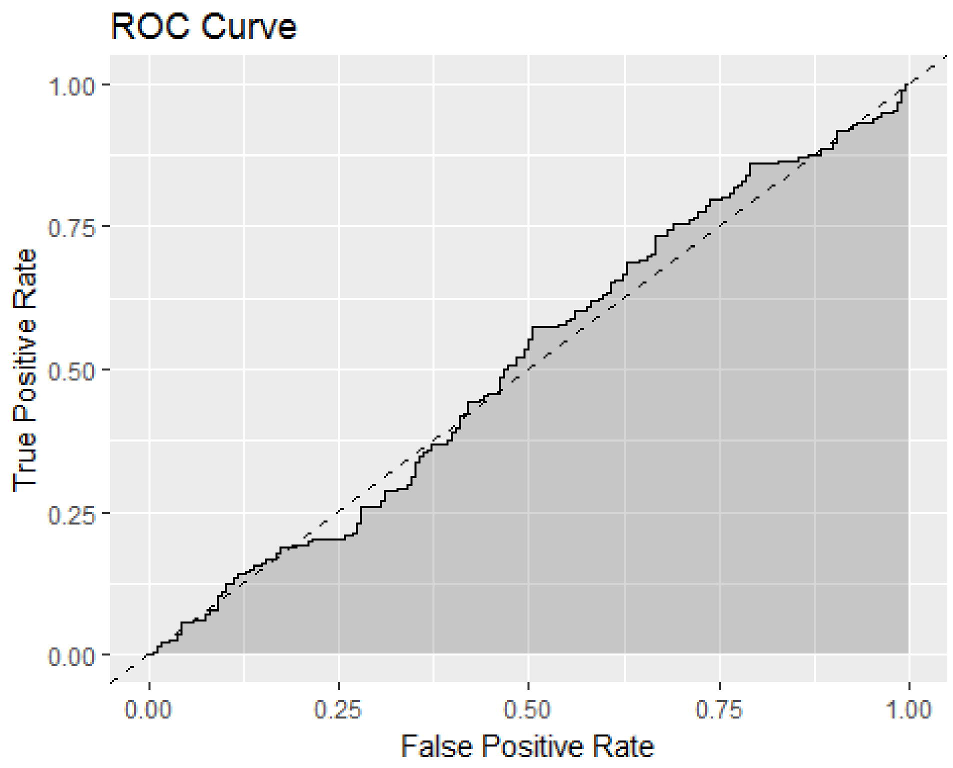
RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'X41.billion' and 'alberto.nisman' and 'amo.yee'
## and 'arabia.accus' and 'assyrian.christian' and 'astronom.discov' and
## 'attack.pari' and 'berta.ccere' and 'bill.c51' and 'bleach.event' and
## 'blogger.raif' and 'border.hospit' and 'call.isi' and 'charli.hebdo' and
## 'china.stock' and 'citi.palmyra' and 'coal.oil' and 'coral.bleach' and
## 'emiss.cheat' and 'emiss.scandal' and 'europ.news' and 'falcon.9' and
## 'fifa.offici' and 'forc.fight' and 'germani.stop' and 'hebdo.cartoon'
## and 'hunt.trophi' and 'indian.space' and 'insult.erdogan' and
## 'isi.kill' and 'isi.suspect' and 'islam.scholar' and 'jeremi.corbyn'
## and 'jihadi.john' and 'kong.booksel' and 'kuan.yew' and 'leav.eu' and
## 'lee.kuan' and 'migrant.back' and 'migrant.crisi' and 'migrant.europ'
## and 'month.record' and 'mossack.fonseca' and 'panama.paper' and
## 'pari.climat' and 'pari.terror' and 'plastic.garbag' and 'raif.badawi'
## and 'refuge.arriv' and 'refuge.germani' and 'research.suggest' and
## 'russian.air' and 'russian.airstrik' and 'saudi.blogger' and 'saudi.coalit'
## and 'saudi.govern' and 'saudil.coalit' and 'secular.blogger' and
## 'snooper.charter' and 'state.alli' and 'state.syria' and 'syria.isi' and
## 'syrian.kurdish' and 'turkey.down' and 'turkey.erdogan' and 'war.isi' and
## 'weapon.ukrain' and 'worth.billion' and 'zika.virus' constant. Cannot scale
## data.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)
```

```
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y  0  1
##       0 90 97
##       1 96 95
#Accuracy
mean(preds_y==ytest)
## [1] 0.489418
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.510582
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5113127
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2) +
  geom_line(aes(y=tpr)) +
  geom_abline(slope=1, intercept=0, linetype='dashed') +
  ggtitle("ROC Curve") +
  ylab('True Positive Rate') +
  xlab('False Positive Rate')
```

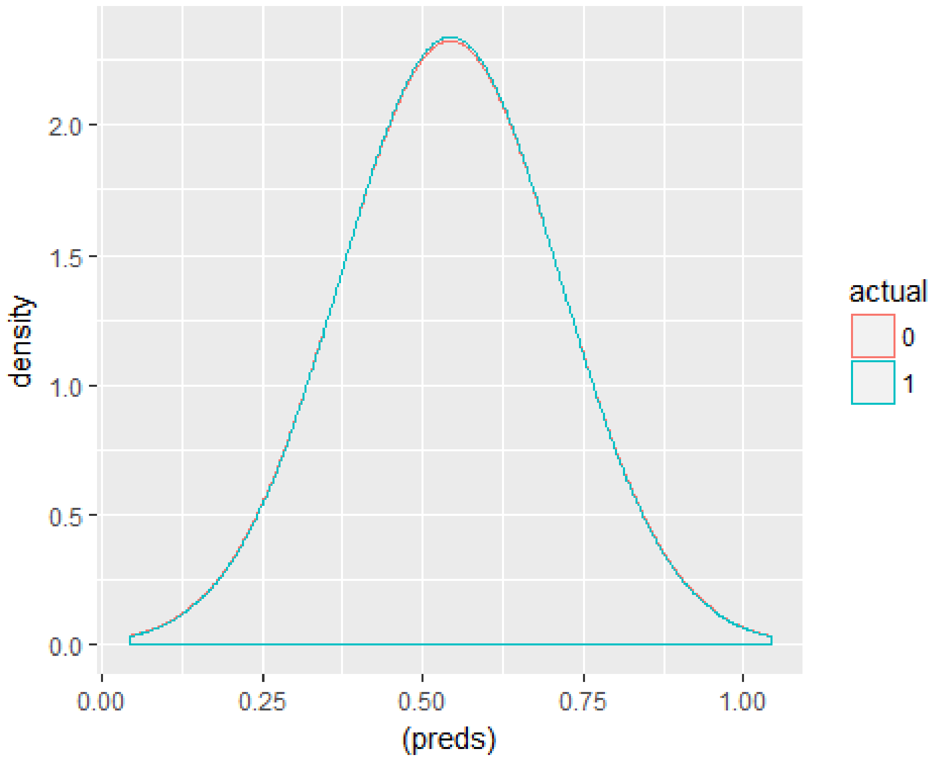


N5G3

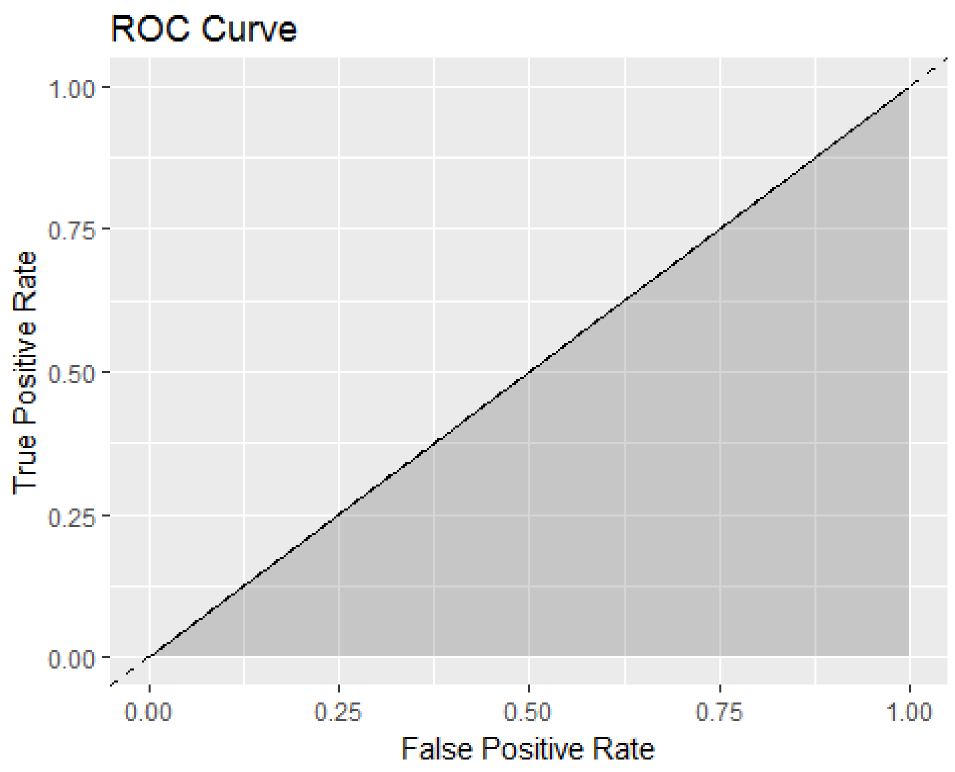
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
```

```
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



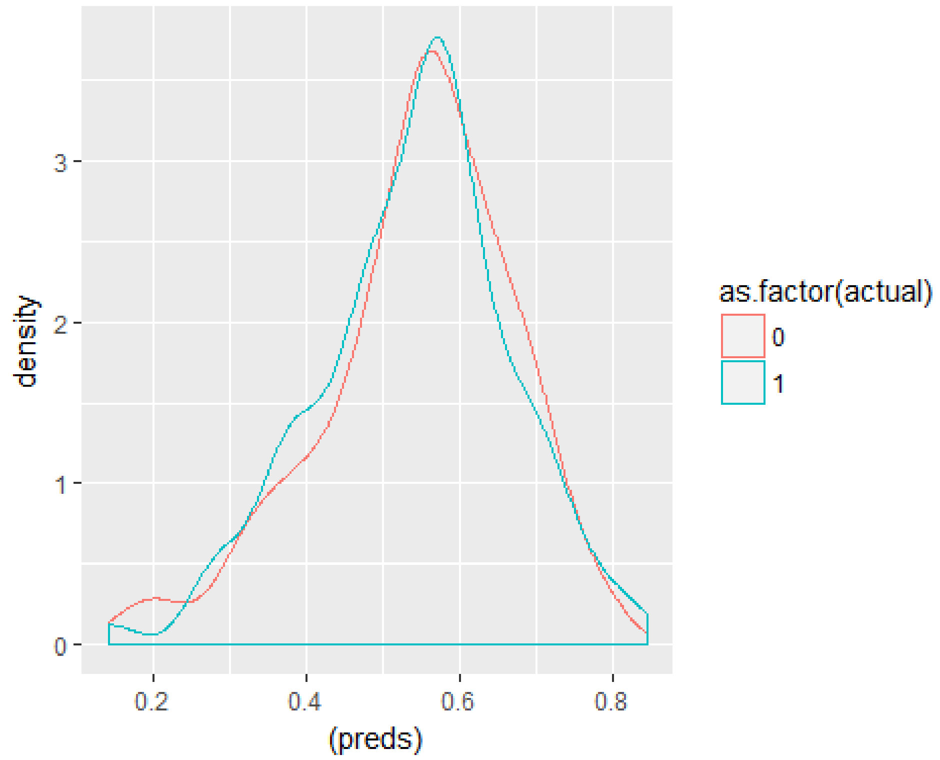
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



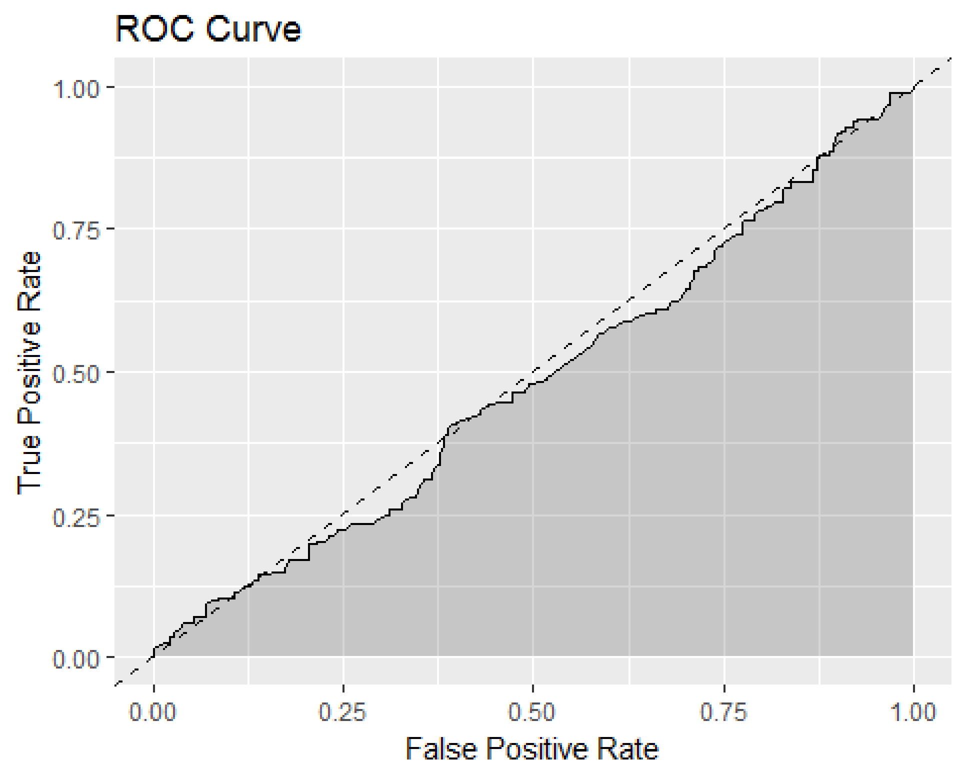
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  54  65
##       1 132 127
#Accuracy
mean(preds_y==ytest)
## [1] 0.478836
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.521164
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

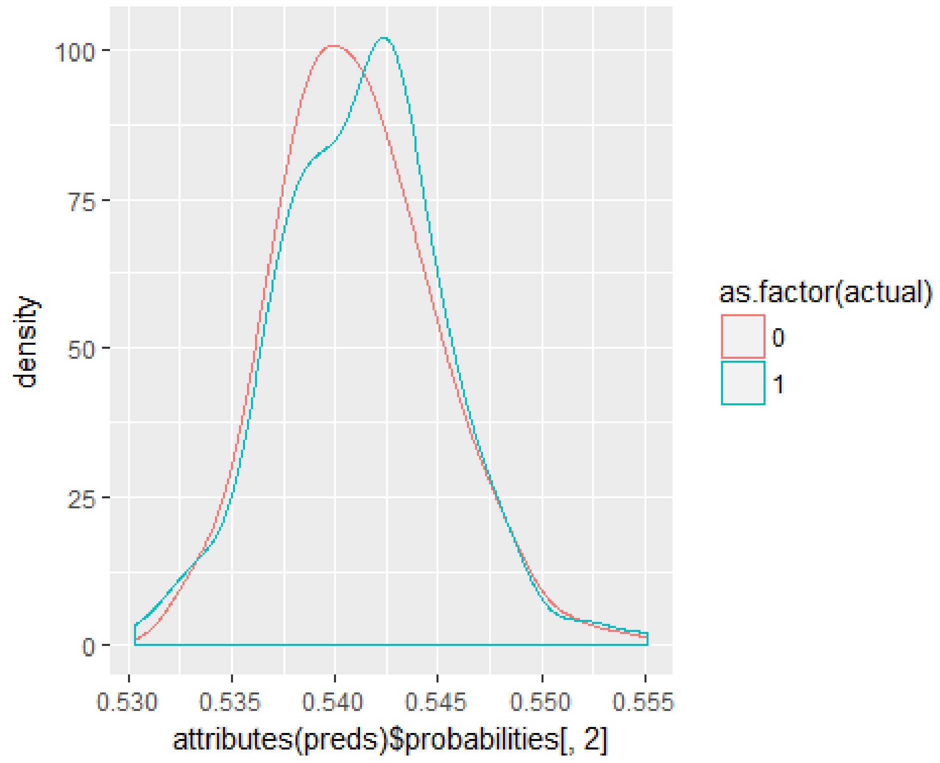


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4820929
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```

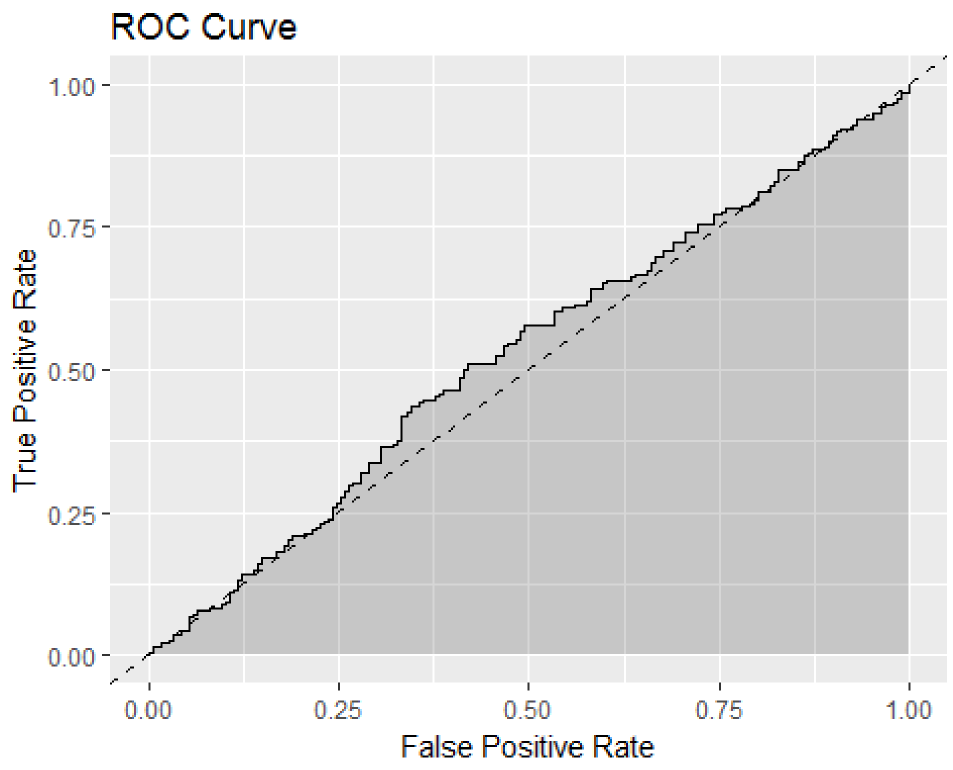


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e
## +05, probability = T): Variable(s) 'X1.billion.euro' and
## 'X100.million.poor' and 'X100.million.user' and 'X10000.syrian.refuge'
## and 'X129.billion.arm' and 'X13.young.peopl' and 'X15billion.saudi.arm'
## and 'X18.million.peopl' and 'X18.month.ago' and 'X2.degre.celsius'
## and 'X200.isi.milit' and 'X2010.plane.crash' and 'X2015.coal.consumpt'
## and 'X2015.hottest.year' and 'X2016.olymp.game' and 'X333.mink.whale'
## and 'X5.year.ahead' and 'X60.million.year' and 'X75.cent.time' and
## 'accus.coverup.heavili' and 'adolf.hitler.mein' and 'afghan.capit.kabul'
## and 'afghan.secur.forc' and 'ahead.pari.climat' and 'air.space.news' and
## 'air.strike.iraq' and 'air.strike.syrian' and 'airasia.flight.qz8501'
## and 'aircraft.carrier.battl' and 'al.qaeda.affili' and
## 'al.qaeda.syria' and 'ali.khamenei.wednesday' and 'ali.moham.alnimr'
## and 'american.petroleum.institut' and 'ananta.bijoy.das' and
## 'ancient.citi.palmyra' and 'ancient.syrian.citi' and 'anim.cruelti.regist'
## and 'annex.crimea.ukrain' and 'annual.militari.exercis'
## and 'antiislam.movement.pegida' and 'antiterror.bill.c51'
## and 'approv.129.billion' and 'arabia.behead.indonesian' and
## 'arabia.human.right' and 'arm.saudi.arabia' and 'arm.shipment.turkish'
## and 'arrest.honduran.activist' and 'astronaut.tim.peak' and
## 'attack.cancer.cell' and 'attack.found.dead' and 'attack.identifi.french'
## and 'attack.kurdish.forc' and 'attack.tourist.hotel' and
## 'australia.pass.law' and 'australian.border.forc' and
## 'australian.david.hick' and 'australian.stock.market' and 'ban.hunt.trophi'
## and 'ban.lion.hunt' and 'bank.help.client' and 'barack.obama.dismiss'
## and 'base.border.ukrain' and 'basic.incom.experi' and 'bbc.news.migrant'
## and 'belgian.polic.arrest' and 'belgian.prime.minist' and
## 'big.game.hunter' and 'big.tobacco.accus' and 'billion.climat.chang' and
## 'billion.fossil.fuel' and 'billion.light.year' and 'billion.peopl.face' and
## 'billion.renew.energi' and 'billion.year.ago' and 'black.market.american'
## and 'bleach.event.underway' and 'blogger.amo.yee' and 'blogger.hack.death'
## and 'blogger.raif.badawi' and 'bodi.immun.system' and 'boko.haram.appear'
## and 'boko.haram.camp' and 'boko.haram.fighter' and 'boko.haram.forc'
## and 'boko.haram.past' and 'boko.haram.stronghold' and 'bomb.boko.haram'
## and 'bomb.doctor.border' and 'bomb.sikh.templ' and 'border.control.face'
## and 'border.hospit.afghanistan' and 'boy.die.ebola' and 'boycott.
## 2018.world' and 'brazil.judg.order' and 'break.global.temperatur'
## and 'britain.leav.eu' and 'brussel.antiterror.raid' and
## 'bueno.air.zoo' and 'bust.child.traffick' and 'calai.refuge.camp' and
## 'call.punish.homosexu' and 'cameroon.armi.kill' and 'canada.electron.spi'
## and 'canada.justic.minist' and 'carbon.emiss.year' and 'case.zika.virus'
## and 'central.american.countri' and 'central.eastern.europ'
## and 'chancellor.georg.osborn' and 'charg.edward.snowden' and
## 'charg.electr.car' and 'charli.hebdo.attack' and 'charli.hebdo.cartoon' and
## 'charli.hebdo.issu' and 'chernobyl.nuclear.plant' and 'china.silk.road' and
## 'china.stock.plung' and 'chines.tourist.travel' and 'chip.implant.brain'
## and 'cia.black.site' and 'citi.demand.presid' and 'class.action.lawsuit'
## and 'client.fossil.fuel' and 'climat.chang.40' and 'climat.chang.found' and
## 'climat.chang.scientist' and 'climat.chang.world' and 'closest.live.relat'
## and 'coal.oil.gas' and 'coalfir.power.plant' and 'coalit.fight.islam' and
## 'comedian.death.threat' and 'commiss.big.tobacco' and 'commit.genocid.iraq'
## and 'compani.econom.nonviabl' and 'complet.renew.energi'
## and 'condemn.saudi.arabia' and 'conduct.airstrik.syria' and
## 'confirm.jihadi.john' and 'controversi.bill.c51' and 'coral.bleach.event'
## and 'corpor.tax.2014' and 'count.accessori.murder' and
## 'coverup.heavili.redact' and 'crimin.charg.manipul' and 'crimin.code.creat'
## and 'cuba.state.sponsor' and 'cuba.terror.list' and 'cut.7800.job' and
## 'cut.water.suppli' and 'cut.welfar.benefit' and 'david.cameron.reveal'
## and 'dead.sign.tortur' and 'deal.saudi.arabia' and 'debri.found.reunion'
## and 'declar.global.emerg' and 'declar.war.isi' and 'deni.armenian.genocid'
## and 'dentist.kill.zimbabw' and 'deploy.nuclear.weapon' and
## 'deploy.thousand.troop' and 'diagnos.thyroid.cancer' and 'diesel.car.sale'
## and 'disappear.15.billion' and 'disappear.43.mexican' and
## 'discuss.gay.right' and 'divest.fossil.fuel' and 'doctor.border.hospit'
## and 'document.reveal.canada' and 'document.transpacif.partnership' and
## 'doesnt.call.punish' and 'dog.meat.festiv' and 'down.russian.jet' and
## 'drill.north.korea' and 'drive.climat.chang' and 'drone.strike.afghanistan'
## and 'drought.middl.east' and 'due.climat.chang' and 'dump.raw.sewag' and
## 'dwarf.planet.cere' and 'eagl.death.metal' and 'earthlik.planet.discov' and
## 'eastern.ukrain.start' and 'el.chapo.prison' and 'electron.spi.agenc' and
## 'endang.siberian.tiger' and 'establish.militari.base' and 'eu.border.agenc'
## and 'eu.court.advis' and 'eu.leader.agre' and 'european.citi.includ'
## and 'european.commiss.big' and 'european.parliament.vote' and
## 'european.union.drop' and 'eus.refuge.quota' and 'euus.trade.deal' and
## 'extend.state.emerg' and 'exxon.knew.climat' and 'face.treason.charg'
## and 'fake.peer.review' and 'falcon.9.rocket' and 'fall.35.barrel'
## and 'fallout.panama.paper' and 'fals.greek.passport' and
## 'farm.produc.electr' and 'fast.food.giant' and 'fernndez.de.kirchner' and
## 'fifa.presid.sepp' and 'fire.missil.sea' and 'firm.mossack.fonseca' and
## 'flown.english.channel' and 'food.giant.mcdonald' and 'forc.fight.isi'
## and 'foreign.minist.juli' and 'fossil.fuel.invest' and 'fossil.fuel.role'
## and 'found.dead.sign' and 'found.guilti.plan' and 'found.reunion.island'
## and 'french.law.block' and 'french.polic.fire' and 'fuel.compani.econom'
## and 'full.text.tpp' and 'futur.climat.chang' and 'gay.imam.quran'
## and 'german.antiislam.movement' and 'german.justic.minist' and
## 'germani.reject.greek' and 'girl.islam.state' and 'give.unsold.food'
## and 'global.children.chariti' and 'global.seed.vault' and
## 'global.temperatur.danger' and 'global.temperatur.record' and 'global.warm.
## 2c' and 'global.warm.stop' and 'goldman.environment.prize' and
## 'govern.intellig.agenc' and 'govern.pass.bill' and 'greec.financ.minist'
## and 'greec.recogn.palestinian' and 'gross.domest.product' and
## 'ground.oper.yemen' and 'group.claim.respons' and 'guilti.crimin.charg'
## and 'guinea.tsunami.warn' and 'hack.death.bangladesh' and
## 'hadron.collid.discov' and 'hama.founder.face' and 'haram.attack.kill' and
## 'haze.southeast.asia' and 'heat.wave.kill' and 'heavili.redact.document'
## and 'hit.middl.east' and 'hit.saudi.coalit' and 'holi.citi.mecca' and
## 'home.alien.life' and 'home.news.news' and 'hong.kong.booksel' and
## 'hong.kong.politician' and 'hospit.war.crime' and 'hottest.month.record'
## and 'hungari.seal.border' and 'imam.quran.doesnt' and
## 'immigr.detent.centr' and 'immun.system.attack' and 'impact.climat.chang'
## and 'increas.number.asylum' and 'indian.consul.afghanistan'
## and 'indian.govern.plan' and 'indian.space.research' and
## 'insult.presid.erdogan' and 'insult.presid.recep' and 'integr.minist.sylvi'
## and 'intend.syrian.rebel' and 'intern.aid.agenc' and 'intern.team.research'
## and 'introduc.temporari.border' and 'invest.partnership.ttip' and
## 'invit.antiisi.confer' and 'iran.recogn.israel' and 'iraq.syria.isi'
## and 'iraqi.citi.mosul' and 'iraqi.presid.saddam' and 'ireland.pass.law'
## and 'irish.samesex.marriag' and 'isi.burn.aliv' and 'isi.buy.oil' and
## 'isi.claim.respons' and 'isi.commit.genocid' and 'isi.leader.baghdadi' and
## 'isi.twitter.account' and 'isil.mustard.gas' and 'islam.state.chief' and
## 'islam.state.destroy' and 'islam.state.mustard' and 'islam.state.returne'
## and 'islam.state.syria' and 'islam.state.video' and 'islamist.group.boko'
## and 'isra.govt.approv' and 'isra.settlement.occupi' and 'jail.6.year'
## and 'japan.apolog.wwii' and 'japanes.court.rule' and 'je.sui.charli'
## and 'johannesburg.south.africa' and 'join.chinaback.aiib' and
## 'join.intern.crimin' and 'jongun.snub.china' and 'judg.order.whatsapp'
## and 'justin.trudeau.promis' and 'khaama.press.kp' and 'kidnap.murder.
## 13' and 'kill.2.million' and 'kill.200.isi' and 'kill.bomb.blast' and
## 'kill.civilian.syria' and 'kim.jong.order' and 'kim.jong.visit' and
## 'kim.jongun.snub' and 'knew.climat.chang' and 'knew.osama.bin' and
## 'kong.booksel.miss' and 'kore
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##      0  62  77
##      1 124 115
#Accuracy
mean(preds_y==ytest)
## [1] 0.468254
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.531746
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```

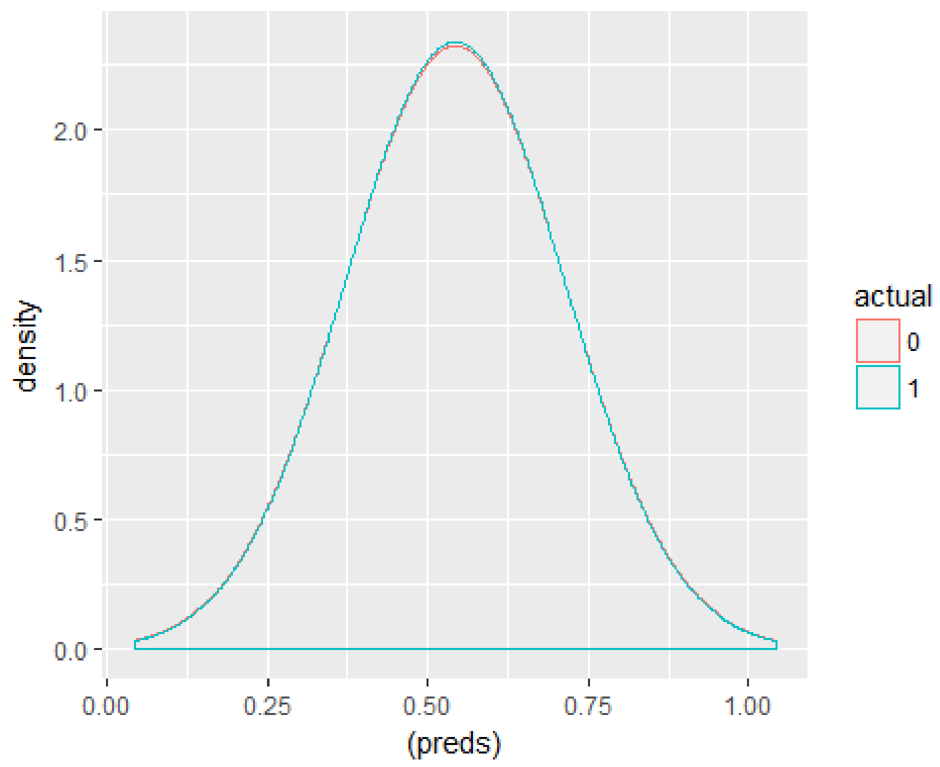


```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5269657
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

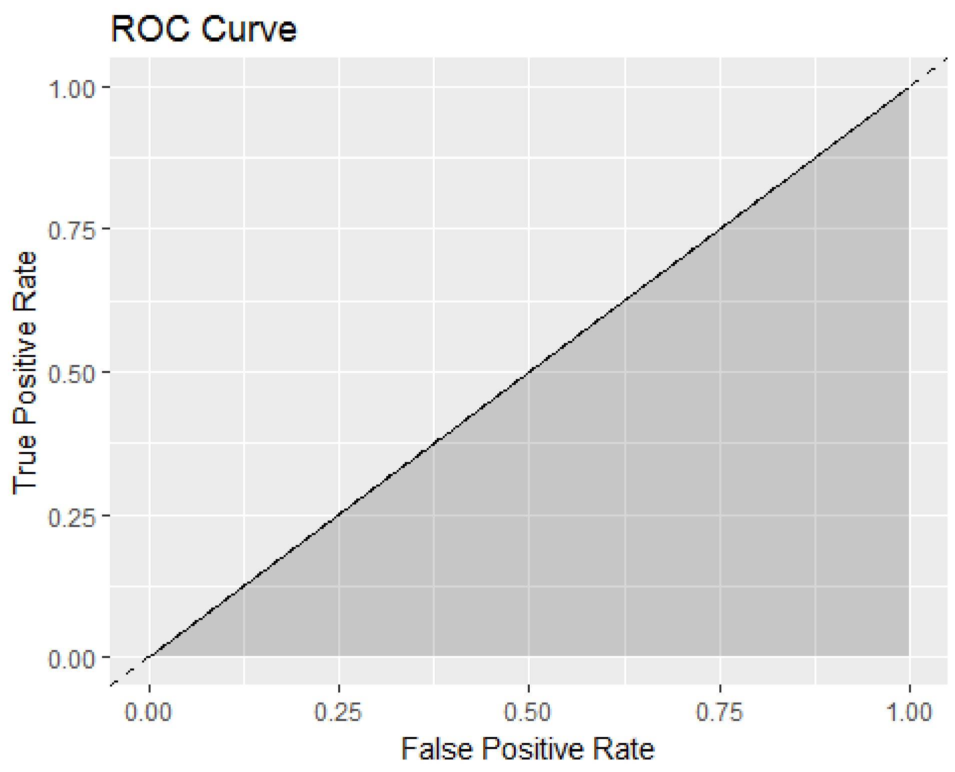


N5G23
Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



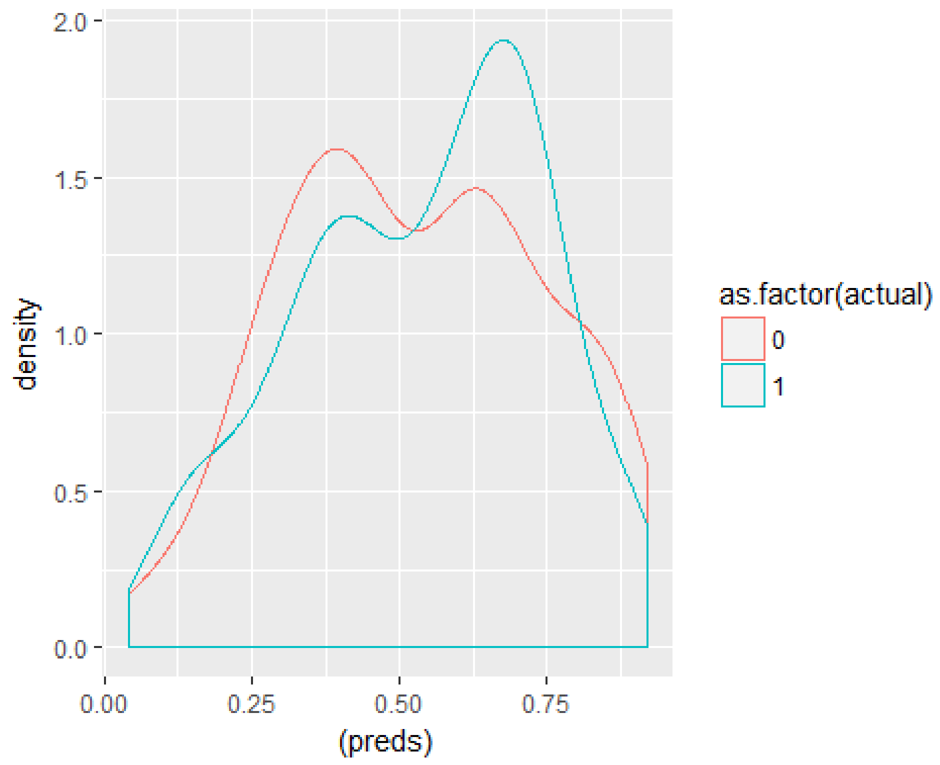
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



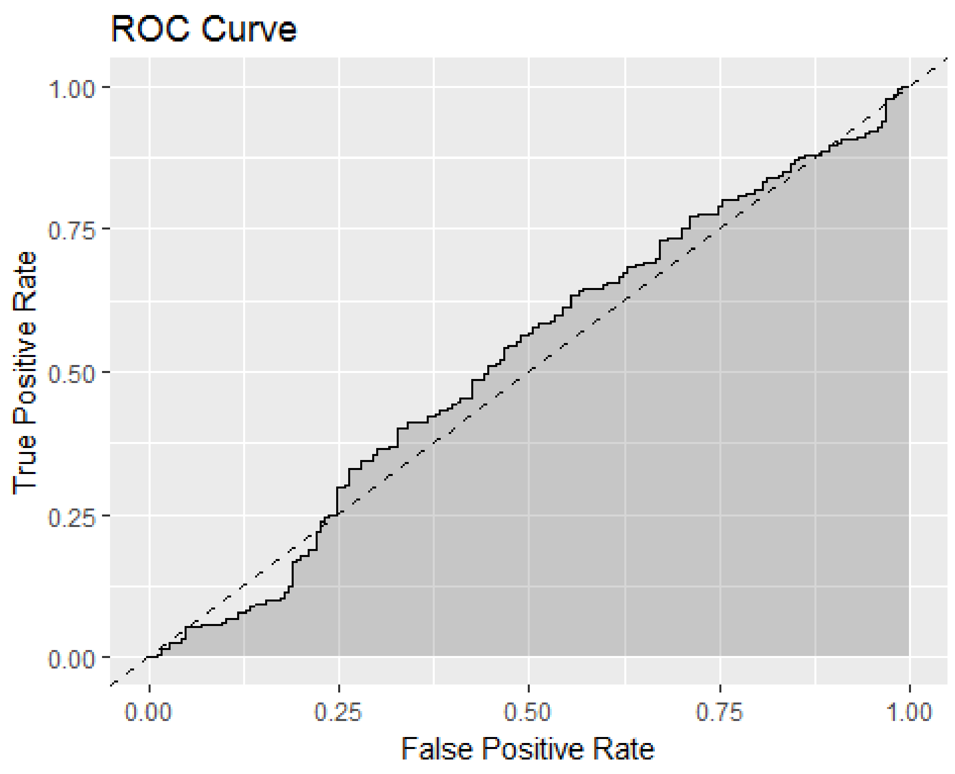
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  88  80
##       1  98 112
#Accuracy
mean(preds_y==ytest)
## [1] 0.5291005
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4708995
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```

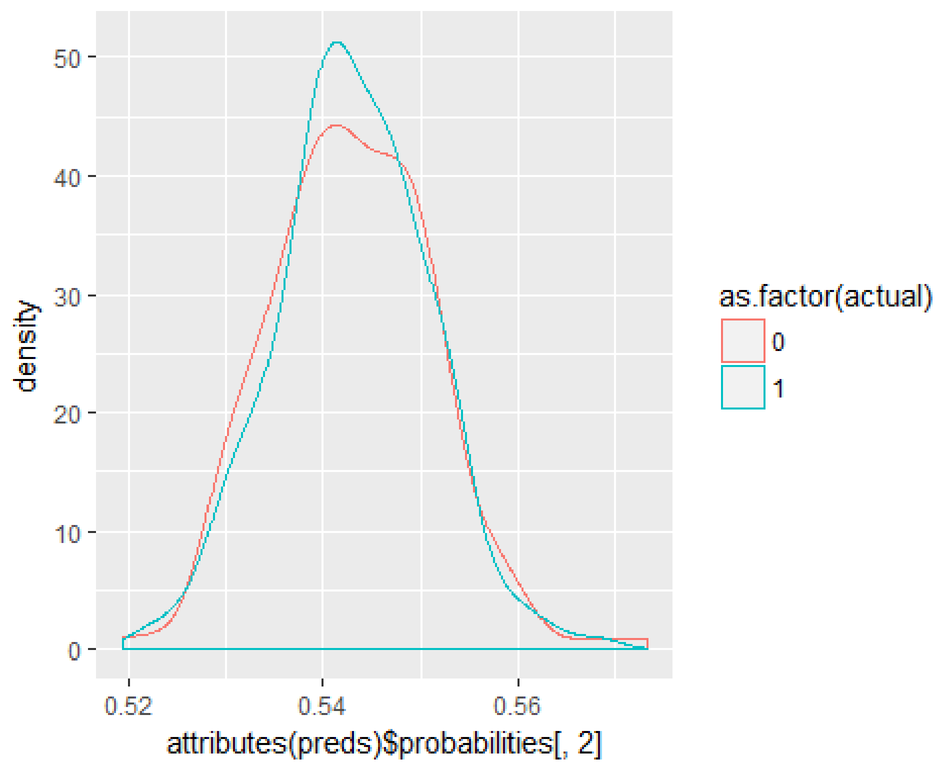


```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5215054
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

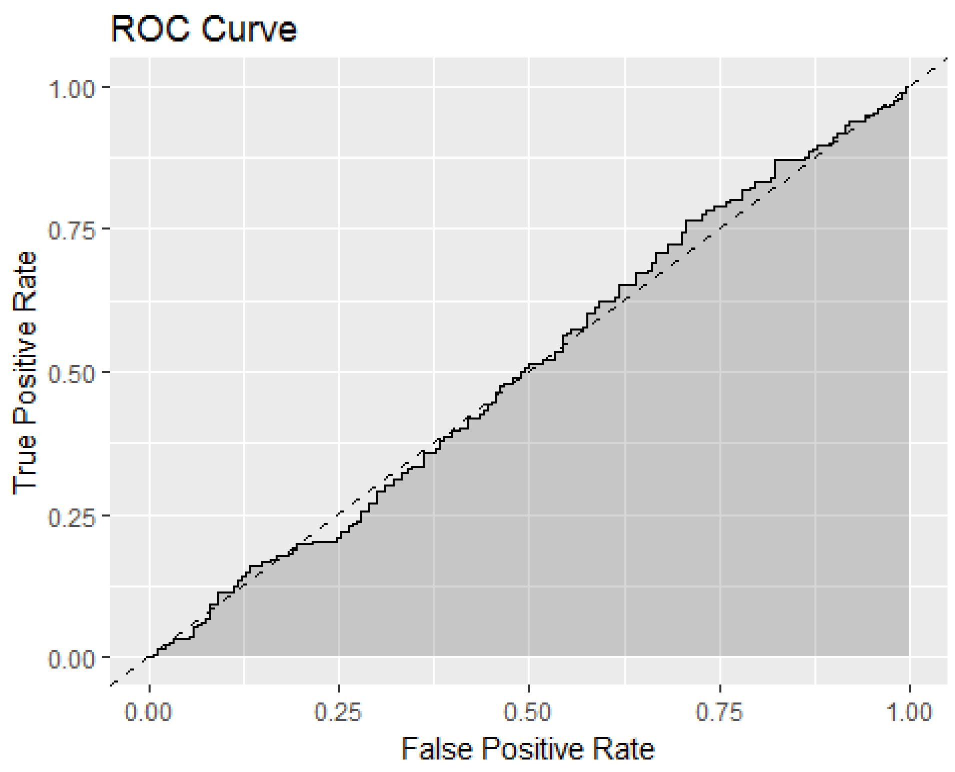


RandomForest Model
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y  0  1
##       0 90 94
##       1 96 98
#Accuracy
mean(preds_y==ytest)
## [1] 0.4973545
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5026455
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



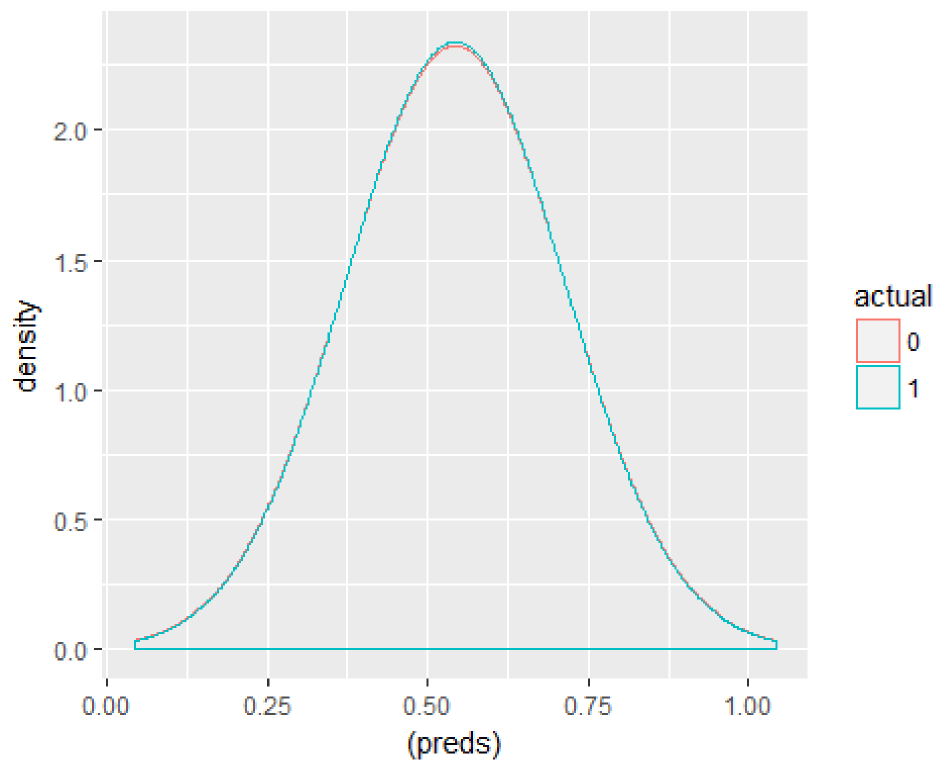
```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5060204
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate')
```



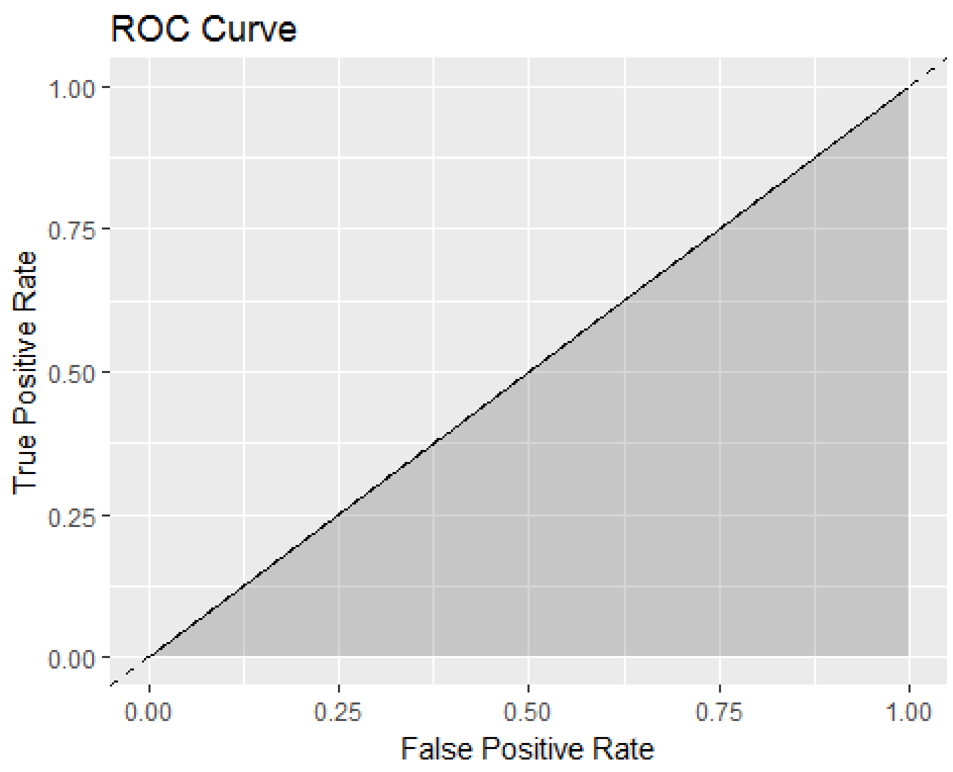
N5G123

Ridge Model

```
knitr::opts_chunk$set(echo = TRUE)
# Train the model
set.seed(50)
glmnet.fit <- cv.glmnet(x=xtrain, y=ytrain, family='binomial', alpha=0)

# Generate predictions
preds <- predict(glmnet.fit, newx=xtest, type='response', s="lambda.min")

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       1 186 192
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)

ggplot(results, aes(x=(preds), color=actual)) + geom_density()
```



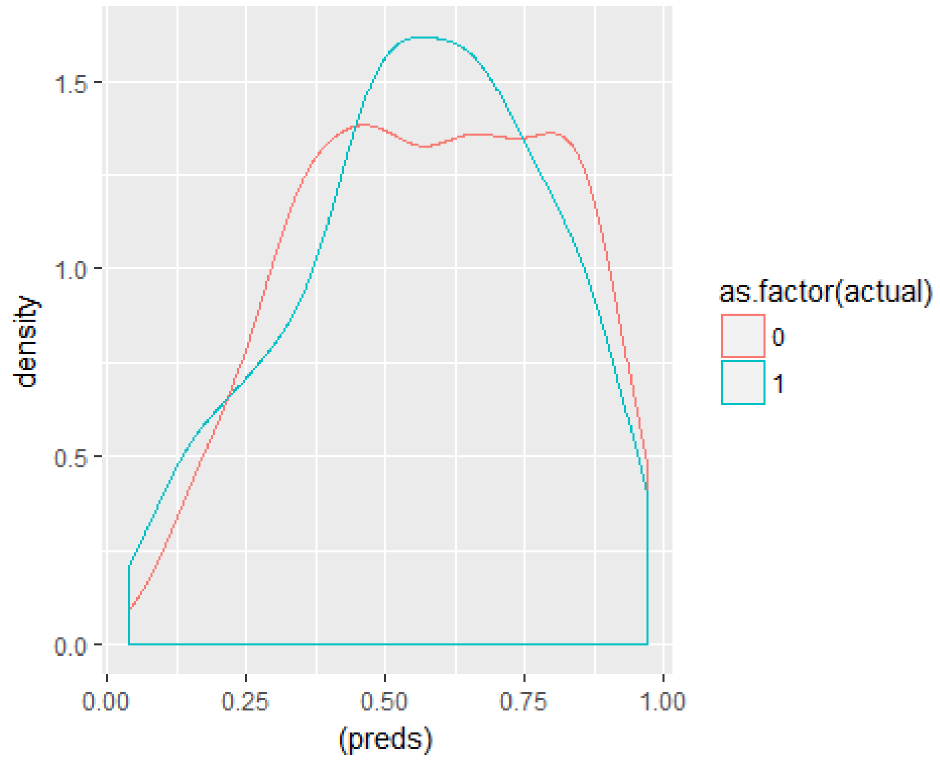
```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```



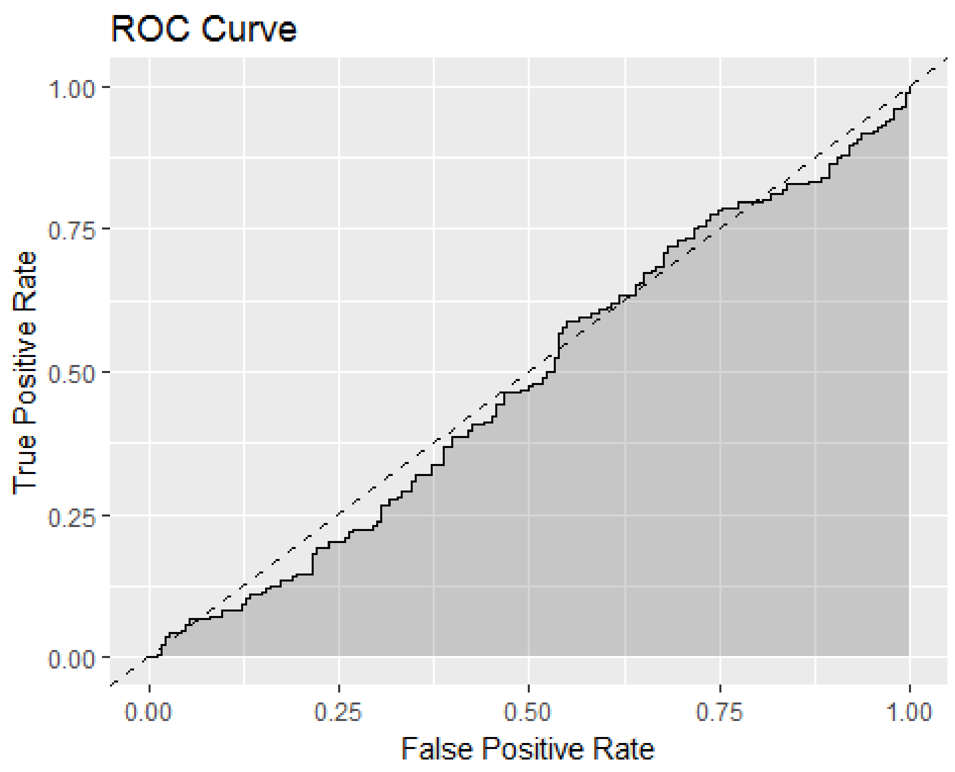
XGBoost Model

```
knitr::opts_chunk$set(echo = TRUE)
xgb_mod <- xgboost(xtrain,
                   label=ytrain,
                   nrounds=100,
                   objective="binary:logistic",
                   verbose=0)



preds <- predict(xgb_mod, xtest)

preds_y <- rep(0,length(ytest))
preds_y[preds>0.5] <- 1
#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  76  76
##       1 110 116
#Accuracy
mean(preds_y==ytest)
## [1] 0.5079365
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.4920635
# Put results into dataframe for plotting.
results <- data.frame(pred=(preds), actual=ytest)
```

```
# Plot dual densities
ggplot(results, aes(x=(preds), color=as.factor(actual))) + geom_density()
```



```
prediction <- prediction((preds), ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.4833109
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```

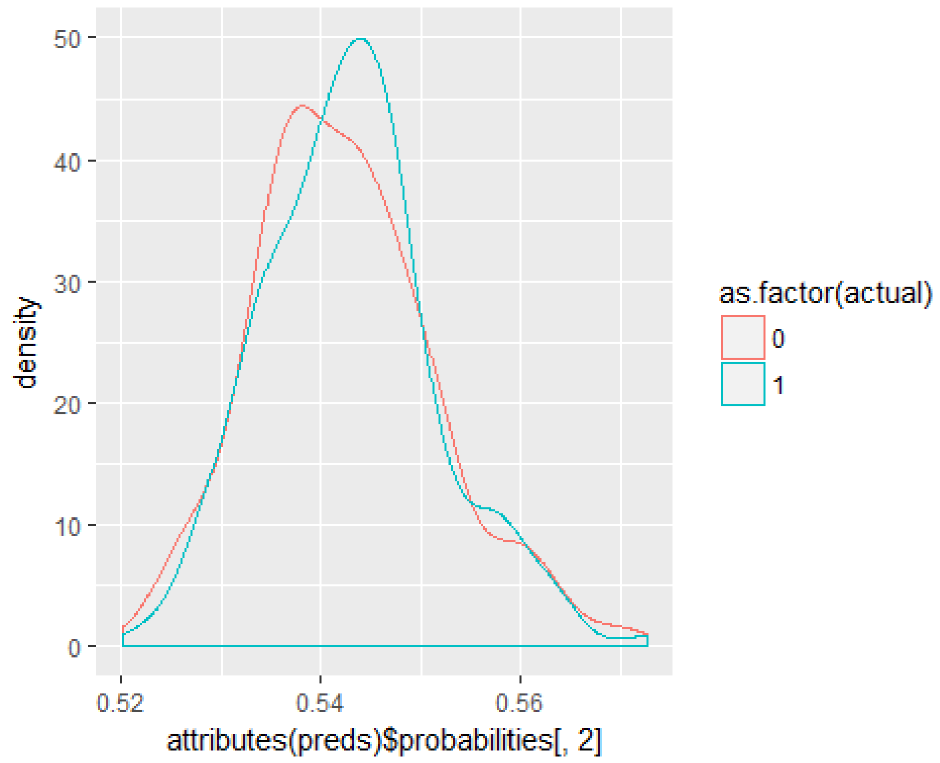


RandomForest Model

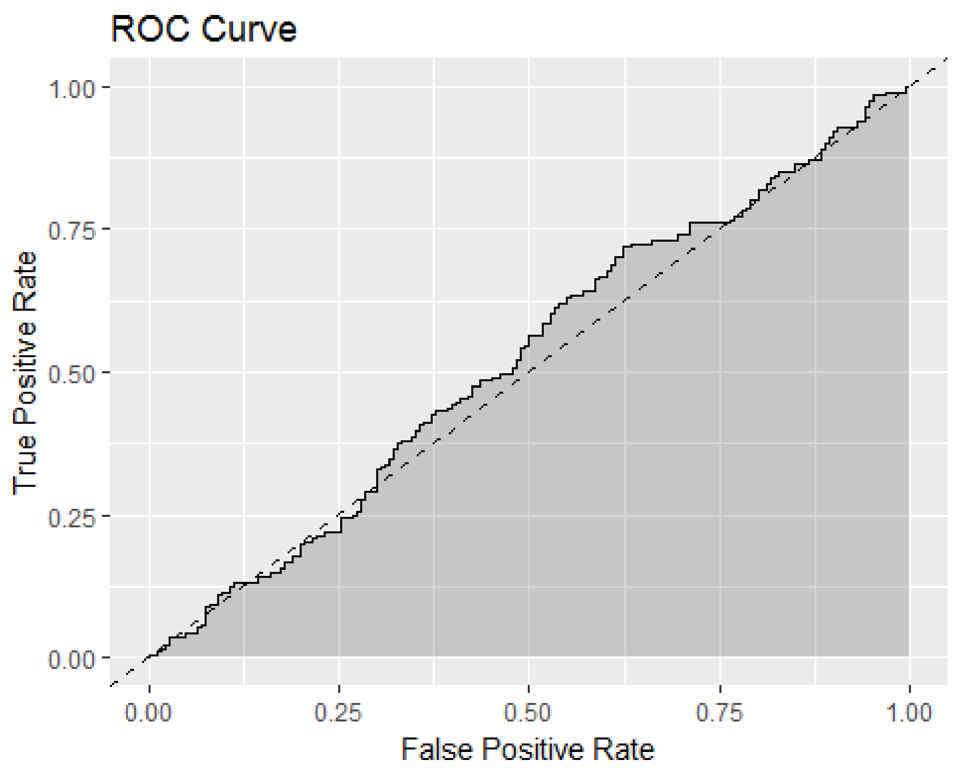
SVM Model

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
svm_model<-svm(xtrain, as.factor(ytrain), cost = 1e+05, probability = T)
## Warning in svm.default(xtrain, as.factor(ytrain), cost = 1e+05, probability
## = T): Variable(s) 'X1.billion.euro' and 'X1.month' and 'X1.tv' and
## 'X10.cent' and 'X10.syrian' and 'X100.drug' and 'X100.kurdish' and
## 'X100.million.poor' and 'X100.million.user' and 'X10000.syrian.refuge' and
## 'X1000km' and 'X100200' and 'X10fold' and 'X11year.low' and 'X125.million'
## and 'X129.billion' and 'X129.billion.arm' and 'X13.young.peopl' and
## 'X131' and 'X135.year' and 'X136' and 'X15.min' and 'X153.chines' and
## 'X15billion.saudi' and 'X15billion.saudi.arm' and 'X15yo' and 'X16.countri'
## and 'X17th.centuri' and 'X18.million.peopl' and 'X18.month.ago' and
## 'X1915.armenian' and 'X1951' and 'X2.degre' and 'X2.degre.celsius'
## and 'X20.barrel' and 'X20.pay' and 'X200.isi' and 'X200.isi.milit'
## and 'X200.lion' and 'X2009.2013' and 'X2010.bp' and 'X2010.plane' and
## 'X2010.plane.crash' and 'X2012.delhi' and 'X2015.budget' and 'X2015.coal'
## and 'X2015.coal.consumpt' and 'X2015.hottest' and 'X2015.hottest.year'
## and 'X2016.olymp.game' and 'X2016.rio' and 'X208' and 'X208.billion'
## and 'X220.million' and 'X23000.peopl' and 'X241' and 'X250000.euro' and
## 'X250yearold' and 'X262' and 'X28bn' and 'X3.indian' and 'X300.girl'
## and 'X300.troop' and 'X329' and 'X333.mink' and 'X333.mink.whale' and
## 'X35.barrel' and 'X3500yrold' and 'X4.femal' and 'X40.civilian' and
## 'X400000.year' and 'X41.billion' and 'X42195' and 'X42195.kilomet'
## and 'X43.year' and 'X4327' and 'X45.kill' and 'X46.million' and 'X483'
## and 'X483.million' and 'X490m' and 'X5.hurt' and 'X5.year.ahead'
## and 'X50.co2' and 'X500000.migrant' and 'X520' and 'X520.year' and
## 'X60.million.year' and 'X60.yrs' and 'X610' and 'X681m' and 'X68magnitud'
## and 'X75.cent.time' and 'X7800' and 'X7800.job' and 'X8500' and
## 'X88.million' and 'X89.million' and 'X900.year' and 'X911.lawsuit' and
## 'X940' and 'X9525' and 'X997' and 'a10' and 'a10.warthog' and 'abba.reject'
## and 'abdeslam' and 'abduct.tortur' and 'abid.intern' and 'abstin' and
## 'abu.sayyaf' and 'abus.peacekeep' and 'abus.survivor' and 'accept.payment'
## and 'access.secur' and 'access.twitter' and 'accident.reveal' and
## 'account.report' and 'account.scandal' and 'accus.corrupt' and
## 'accus.coverup' and 'accus.coverup.heavili' and 'accus.turkey' and
## 'ad.member' and 'adblock' and 'address.root' and 'admit.fighter' and
## 'adolf.hitler.mein' and 'advanc.weapon' and 'advers.effect' and 'aegean'
## and 'affili.gaza' and 'afghan.capit.kabul' and 'afghan.reconstruct' and
## 'afghan.secur.forc' and 'afghan.syrian' and 'afghanistan.offici' and
## 'africa.confirm' and 'africa.europ' and 'age.found' and 'agenc.isro'
## and 'agenc.studi' and 'agre.ban' and 'agre.give' and 'agre.monday' and
## 'agre.move' and 'agreement.reach' and 'ahead.pari' and 'ahead.pari.climat'
## and 'ahead.special' and 'aiib' and 'aim.ban' and 'air.space.news'
## and 'air.strike.iraq' and 'air.strike.syrian' and 'air.zoo' and
## 'airasia.flight.qz8501' and 'aircraft.advanc' and 'aircraft.bomb'
## and 'aircraft.carrier.battl' and 'aircraft.syria' and 'airport.evacu'
## and 'airstrik.hit' and 'al.qaeda.affili' and 'al.qaeda.syria'
## and 'alabadi' and 'alberto.nisman' and 'alexandria.ancient' and
## 'ali.khamenei.wednesday' and 'ali.moham' and 'ali.moham.alnimr' and
## 'alleg.cover' and 'allout.assault' and 'alnimr' and 'alnimr.crucifixion'
## and 'alongsid.kurdish' and 'american.dentist' and 'american.petroleum'
## and 'american.petroleum.institut' and 'amid.terror' and 'amo.yee' and
## 'amp.govt' and 'amp.ineffect' and 'amp.intern' and 'amp.rape' and 'amun'
## and 'analysi.climat' and 'ananta' and 'ananta.bijoy' and 'ananta.bijoy.das'
## and 'anbar' and 'ancient.citi.palmyra' and 'ancient.inscript' and
## 'ancient.shipwreck' and 'ancient.syrian' and 'ancient.syrian.citi'
## and 'anim.cruelti.regist' and 'anim.fur' and 'anim.wipe' and
## 'annex.crimea.ukrain' and 'annual.militari' and 'annual.militari.exercis'
## and 'anomali' and 'antelop' and 'antidepress' and 'antiisi.confer' and
## 'antiislam.movement' and 'antiislam.movement.pegida' and 'antimigr'
## and 'antirefuge' and 'antisemit.attack' and 'antiterror.bill.c51'
## and 'antitrust.case' and 'apolog.wwii' and 'app.store' and 'appl.fbi'
## and 'approv.129' and 'approv.129.billion' and 'approv.secur' and
## 'arab.muslim' and 'arabi' and 'arabia.accus' and 'arabia.behead.indonesian'
## and 'arabia.build' and 'arabia.host' and 'arabia.human' and
## 'arabia.human.right' and 'arabia.kill' and 'arabia.made' and
## 'arabia.militari' and 'arabia.rais' and 'arabia.readi' and 'arabia.reject'
## and 'arabia.top' and 'arbitrarili' and 'arctic.water' and 'area.hit' and
## 'argentin.prosecutor' and 'arm.man' and 'arm.saudi' and 'arm.saudi.arabia'
## and 'arm.shipment.turkish' and 'armenian.kill' and 'armi.child' and
## 'armi.gen' and 'armi.mass' and 'armi.posit' and 'armi.recaptur' and
## 'arrest.brussel' and 'arrest.honduran' and 'arrest.honduran.activist' and
## 'arrest.jewish' and 'arrow.wound' and 'arsenal.year' and 'ask.explain' and
## 'assad.crucial' and 'assang.embassi' and 'assembl.show' and 'asset.manag'
## and 'assyrian.christian' and 'astronaut.tim' and 'astronaut.tim.peak'
## and 'astronom.discov' and 'astronom.observ' and 'asylum.centr' and
## 'asylum.polici' and 'attack.20' and 'attack.arrest' and 'attack.calai'
## and 'attack.cancer' and 'attack.cancer.cell' and 'attack.face'
## and 'attack.found' and 'attack.found.dead' and 'attack.freedom'
## and 'attack.fugit' and 'attack.heroic' and 'attack.identifi' and
## 'attack.identifi.french' and 'attack.kurdish.forc' and 'attack.locat'
## and 'attack.pari' and 'attack.refuge' and 'attack.ringlead' and
## 'attack.tourist' and 'attack.tourist.hotel' and 'attempt.bring'
## and 'auditorgener' and 'australia.cut' and 'australia.join' and
## 'australia.pass.law' and 'australian.border' and 'australian.border.forc'
## and 'australian.david' and 'australian.david.hick' and 'australian.report'
## and 'australian.stock' and 'australian.stock.market' and 'author.money' and
## 'author.shut' and 'aviv.attack' and 'avoid.dead' and 'ayatollah.seyi' and
## 'back.kurd' and 'back.refuge' and 'back.ship' and 'bad.luck' and 'badawi'
## and 'badawi.sentenc' and 'ban.corpor' and 'ban.hunt.trophi' and 'ban.lion'
## and 'ban.lion.hunt' and 'ban.possess' and 'ban.reddit' and 'ban.whatsapp'
## and 'bangkok.bomb' and 'bangladesh.bank' and 'bank.help.client' and
## 'barack.obama.dismiss' and 'barrel.time' and 'base.border.ukrain' and
## 'base.island' and 'baselin' and 'basic.incom.experi' and 'bataclan'
## and 'bataclan.attack' and 'bbc.execut' and 'bbc.news.migrant' and
## 'bear.eat' and 'bear.imag' and 'begin.militari' and 'behead.corps'
## and 'behead.indonesian' and 'belgian.court' and 'belgian.polic.arrest'
## and 'belgian.prime' and 'belgian.prime.minist' and 'believ.taliban'
## and 'benckis' and 'berta' and 'berta.ccere' and 'bezo' and 'biaowiea'
## and 'biaowiea.forest' and 'bibl.passag' and 'big.data' and 'big.firm'
## and 'big.game.hunter' and 'big.tobacco.accus' and 'biggest.demonstr'
## and 'biggest.illeg' and 'biggest.leak' and 'bijoy' and 'bijoy.das'
## and 'bill.c51' and 'billion.annual' and 'billion.australian' and
## 'billion.climat.chang' and 'billion.european' and 'billion.fossil' and
## 'billion.fossil.fuel' and 'billion.light' and 'billion.light.year' and
## 'billion.peopl.face' and 'billion.renew.energi' and 'billion.year.ago'
## and 'billionair.oligarch' and 'binari' and 'binyamin.netanyahus'
## and 'biotechnolog' and 'birth.twin' and 'black.market.american'
## and 'black.site' and 'blacken' and 'blazer' and 'bleach.event' and
## 'bleach.event.underway' and 'blitz' and 'block.final' and 'blogger.amo'
## and 'blogger.amo.yee' and 'blogger.hack' and 'blogger.hack.death' and
## 'blogger.raif' and 'blogger.raif.badawi' and 'blow.ancient' and 'blower'
## and 'bnd' and 'boat.coast' and 'bodi.immun' and 'bodi.immun.system' and
## 'boko.haram.appear' and 'boko.haram.camp' and 'boko.haram.fighter' and
## 'boko.haram.forc' and 'boko.haram.past' and 'boko.haram.stronghold' and
## 'bolivar' and 'bomb.boko' and 'bomb.boko.haram' and 'bomb.doctor' and
## 'bomb.doctor.border' and 'bomb.fear' and 'bomb.russian' and 'bomb.sikh'
## and 'bomb.sikh.templ' and 'bomb.yemeni' and 'bomber.enter' and
## 'bomber.identifi' and 'booksel' and 'booksel.miss' and 'bookshelv' and
## 'booster' and 'border.call' and 'border.control.face' and 'border.hospit'
## and 'border.hospit.afghanistan' and 'border.refuge' and 'border.serbia' and
## 'border.staff' and 'bori.nemtsov' and 'bounti.head' and 'boy.
preds <- predict(svm_model, xtest, probability = T)

preds_y <- predict(svm_model, xtest, probability = F)

#Confusion Matrix
table(preds_y,ytest)
##        ytest
## preds_y   0   1
##       0  81  91
##       1 105 101
```

```
#Accuracy
mean(preds_y==ytest)
## [1] 0.4814815
#Misclassification rate
mean(preds_y!=ytest)
## [1] 0.5185185
# Put results into dataframe for plotting.
results <- data.frame(pred=attributes(preds)$probabilities[,2], actual=ytest)

# Plot dual densities
ggplot(results, aes(x=attributes(preds)$probabilities[,2], color=as.factor(actual))) +
geom_density()
```



```
prediction <- prediction(attributes(preds)$probabilities[,2], ytest)
perf <- performance(prediction, measure = "tpr", x.measure = "fpr")

auc <- performance(prediction, measure = "auc")
auc <- auc@y.values[[1]]
auc
## [1] 0.5229895
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
   geom_ribbon(alpha=0.2) +
   geom_line(aes(y=tpr)) +
   geom_abline(slope=1, intercept=0, linetype='dashed') +
   ggtitle("ROC Curve") +
   ylab('True Positive Rate') +
   xlab('False Positive Rate')
```


ROC Curve
True Positive Rate
False Positive Rate
0.00
0.25
0.50
0.75
1.00
0.00
0.25
0.50
0.75
1.00