How to Get a Kickstart: An Analysis of Kickstarter Success

Hoyt Gong
Younghu Park
Hiyori Yoshida

Contents

Summary

The dataset that we used comes from [scraping kickstarter projects across 2019] (https://webrobots.io/kickstarter-datasets/?fbclid=IwAR2_FwI9w0M7loFhTbALyzQ0knddvD8M6IPqYQKuJ-jAP2gAc9vr4CEz95o) and consists of various characteristics of these crowdsourced companies/projects. Our dataset captures all past and current kickstarter project postings as of the November 14, 2019 scrape date from the website.

As the burgeoning landscape of startups are shifting towards [alternative sources of funding outside traditional Venture Capital funding] (https://www.nytimes.com/2019/01/11/technology/start-ups-rejecting-venture-capital.html) , an increasing number of founders are turning towards crowdsourcing. [Kickstarter] (https://help.kickstarter.com/hc/en-us/articles/115005028514-What-are-the-basics-) is arguably the most well known crowdsourcing online platform where founders can post their projects for financial backing. Since Kickstarter's founding in 2009, over 450,000 projects have successfully raised more than $4.6 billion on the platform from 17.2 million friends, family and fans who want to help get those ventures off the ground. These supporters, officially termed backers by Kickstarter, are the cardinal drivers in this grassroots-style fundraising strategy of the entrepreneurship ecosystem. Importantly, while founders are turning to these crowdsourcing methods for purposes of de-risking and reducing financial leverage from external institutions, it currently has a [37.44% success rate] (https://www.statista.com/statistics/235405/kickstarter-project-funding-success-rate/) as defined by the multinational data portal, Statista. As such, we are interested in building a predictive classification model that can forecast the success of a Kickstarter project based on its features. For advisory purposes, we hope that this project's multifactorial analysis can provide insight into critical characteristics of a successful project on Kickstarter for recommendation to future companies hoping to raise capital via online crowdsourcing. Beyond a founder's perspective, our analysis is also insightful for cases of dual-financing where investors and financial analysts from the sell-side are able to adjust project valuation based on our model results.

Description of the Problem

First, some terms are defined to help explain how Kickstarter works.

A **creator** is the person or team behind a project idea.

A **project** is a finite work with a clear goal for which a creator is seeking funding.

The **funding goal** is the amount of money that a creator seeks to raise for their project.

Backers are people who **pledge** money to help creators make their projects a reality.

Creators who seek funding on Kickstarter want their projects to succeed so they can bring their ideas to life. Funding on Kickstarter is all-or-nothing. This means the creator will not receive the money people have pledged toward their project unless the project reaches its funding goal. Thus, "success" in this case is reaching the funding goal. As demonstrated in Kickstarter's Creator Handbook, creators put a lot of effort into designing their project, funding, promotion, and communication to increase their chances of reaching their funding goal.

The goal of this paper is to use attributes of existing Kickstarter projects in classifiers to predict whether a project will succeed or fail. Determining the atrributes of a project that are most valuable in predicting success can help creators design their projects to maximize their chances of reaching their funding goal.

Description of the Data

Web Robots has been scraping data from Kickstarter projects monthly starting March 2016. The most recent data from November 2019 is used in this project. The response variable that is predicted in this project is `successful`, a categorical variable that is either 1 for "successful" or 0 for "failed". The original data includes "canceled" as a state, which are converted to "failed". This is because it is assumed that canceled projects would likely have failed. The following independent variables are considered in the analysis:

Independent Variable	Description
`backers_count`	The number of backers that have pledged some amount of money to the project.
`blurb`	A short summary at the top of project pages that describe/promote the project.
`category`	The category of the project. This dataset includes 160 different categories, including "public Art," "software," "narrative film," etc.
`slug` -> `description`	Select keywords to describe the project, renamed to description
`country`	The country in which the project is based.
`location`	The city and state in wich the project is based.
`deadline`	The date and time of the deadline until which the project will raise funds. After the deadline, the project is determined to be "successful" or "failed" depending on whether it reached its funding goal.
`launched_at`	The date and time at which the project was made public and started raising funds.
`duration_days`	deadline - launched_at. The fundraising duration of the project in days.
`goal`	The funding goal of the project.
`name`	The title of the project.
`usd_pledged`	The amount pledged towards the project in USD.
`staff_pick`	True if the Kickstarter staff have chosen to feature the project in their "Projects We Love" program, false otherwise.

Data Cleaning and Preparation

```r
#to import dataset
kick_data <- read.csv("kick_data.csv", header=TRUE)
```

```r
dim(kick_data)
```

```
## [1] 93247    40
```

```r
colnames(kick_data)
```

```
##  [1] "X.1"                     "X"
##  [3] "backers_count"           "blurb"
##  [5] "category"                "converted_pledged_amount"
##  [7] "country"                 "country_displayable_name"
##  [9] "created_at"              "creator"
## [11] "currency"                "currency_symbol"
## [13] "currency_trailing_code"  "current_currency"
## [15] "deadline"                "disable_communication"
## [17] "friends"                 "fx_rate"
## [19] "goal"                        "id"
## [21] "is_backing"                  "is_starrable"
## [23] "is_starred"                  "launched_at"
## [25] "location"                    "name"
## [27] "permissions"                 "photo"
## [29] "pledged"                     "profile"
## [31] "slug"                        "source_url"
## [33] "spotlight"                   "staff_pick"
## [35] "state"                       "state_changed_at"
## [37] "static_usd_rate"             "urls"
## [39] "usd_pledged"                 "usd_type"
```

First, because our data includes multiple entries for each company whenever the state of their project changes (i.e. from live to failed), we needed to get the rows with the companies most recent states. We did this by finding the max value of `state_changed_at`. Additionally, some rows seemed to contain duplicates of companies, with only the `url` to be different, so we simply utilized the autogenerated index row `X` to drop the duplicates and renamed this as `sub_id` for identification purposes.

```r
#filtering out duplicate entries by keeping the most recent item and additionally filtering by the uniq
kick_data.f <- kick_data %>% group_by(id)%>% filter(state_changed_at == max(state_changed_at)) %>% filt
```

Next, we wanted to filter down our columns to the ones we detailed above. We also included the `id` and `sub_id` columns to uniquely identify each row.

```r
#selecting columns to use, temporarily selecting id columns for merging and identification
to_use<- c("id", "sub_id",  "state", "backers_count", "blurb", "slug", "category", "converted_pledged_an
kick_data.f <- kick_data.f %>% select(to_use)
```

Next, we converted the `launched_at` and `deadline` columns to date formats.

```r
#converting launched_at and deadline to datetime formats
dts <- kick_data.f$launched_at
mydates = dts
class(mydates) = c('POSIXt','POSIXct')
kick_data.f$launched_at <- as.Date(mydates)

dts <- kick_data.f$deadline
mydates = dts
class(mydates) = c('POSIXt','POSIXct')
kick_data.f$deadline <-  as.Date(mydates)
```

Then, we proceeded to create our `duration_days` column to indicate the timespan of the project.

```r
#calcuation duration of project in days
kick_data.f$duration_days <- as.numeric(difftime(kick_data.f$deadline ,kick_data.f$launched_at , units
```

Next, we needed to handle to json data within the `category` and `location` columns. For the `category` column, we were able to successfully transform the data into a json and extract the `category` and `subcategory` columns. However, the location column had some erroneous formatting, and we thus manually extracted the name with string manipulation searching for the `displayable_name` tag. In order to cut down the number of unique locations, we also only preserved the country of origin. For projects in the United States, we seperated them by US Region.

```
#parsing our json data to extract categories
ParseJSONColumn <- function(x)  {
     str_c("[ ", str_c(x, collapse = ",", sep=" "), " ]") %>%
      fromJSON(flatten = T) %>%
     as.tibble()
}

JSONcolumn_data <- kick_data.f  %>%
    select(id, category)  %>%
    map_dfc(.f = ParseJSONColumn)  %>% select(value, name, slug) %>% rename(id = value, category = name


kick_data.f <- kick_data.f %>% select(-category) %>% rename(description=slug)

kick_data.f <-  merge(kick_data.f, JSONcolumn_data ,by="id")

#extracting locations through string splits
parsed_location <- kick_data.f %>% separate(location, c("foo", "bar"), '\"displayable_name\":\"') %>% se

kick_data.f$location <- parsed_location$location

northeast =c('CT', 'ME', 'MA', 'NH', 'RI', 'VT', 'NJ', 'NY', 'PA')
midwest = c('IL', 'IN', 'MI', 'OH', 'WI', 'IA', 'KS', 'MN', 'MO', 'NE', 'ND', 'SD')
west =c('AZ', 'CO', 'ID', 'MT', 'NV', 'NM', 'UT', 'WY', 'AL', 'CA', 'HI', 'OR', 'WA')
south = c('MD', 'DE', 'VA', 'WV','KY','TN','NC', "SC", "FL", "GA", "AL", "MS", "LA", "AK", "TK", "OK")
kick_data.f$location <- ifelse(kick_data.f$location %in% northeast, "US-Northeast", ifelse(kick_data.f$l
```

Next, we converted various columns to our desired variable type. We also further cleaned the `description` column to get rid of the hyphens between each words.

```
#converting types of columns
kick_data.f$location <- as.factor(kick_data.f$location)
kick_data.f$blurb <- as.character(kick_data.f$blurb)
kick_data.f$description <- as.character(kick_data.f$description)
kick_data.f$name <- as.character(kick_data.f$name)
kick_data.f$category <- as.factor(kick_data.f$category)

#replacing dashes with spaces in descriptions column and removing _truncated_ tag in blurb
kick_data.f$description <- gsub("-", " ",kick_data.f$description)
```

Finally, we created our indicator column to predict on. First, we filtered out the `live` projects as those were still ongoing and thus were not properly concluded. For the remaining projects, we classified all projects with the `state`, `successful` as 1, while `failed` and `canceled` projects would be 0.

```
#filtering out the live projects, but storing them in a seperate place
kick_data.f.live <- kick_data.f %>% filter(state=="live")
kick_data.f <- kick_data.f %>% filter(state != "live")
```

```
#creating predictor column
kick_data.f$successful<- as.factor(as.numeric(ifelse(kick_data.f$state=="successful", 1, 0)))
```

As only last sanity check, we took only the columns we wanted to move forward with.

```
#double checking that we all desired columns and removing ids
to_use<- c("state", "successful", "backers_count", "blurb", "category", "description", "converted_pledge

kick_data.f <- kick_data.f %>% select(to_use)
```

```
#remove converted_pledged_amount because it is the same as usd_pledged, just converted differently
kick_data.f <- subset(kick_data.f, select = -converted_pledged_amount)
```

The only columns with NAs appear to be `location` and `blurb`. There is one NA in `blurb` and 117 NAs in `location`. Since there is no reasonable way to interpolate the names of cities in which projects are based or the blurb of the project, rows that have NA values in `location` or `blurb` are removed from the dataset.

The dataset includes 85405 observations and 15 variables. For computational reasons, a quarter of the dataset is randomly sampled and used for the models instead of the entire dataset.

```
set.seed(10)
#First randomly sample a quarter of the data for computational purposes
sample.size <- round(dim(kick_data.f) * 0.25)
index.sample <- sample(nrow(kick_data.f), sample.size)
data <- kick_data.f[index.sample, ] #dim(data) = 21351        15
```

The final dataset used for the models includes 21351 observations and 15 variables.

The final dataset is split into training and testing sets, 70% for training and 30% for testing.

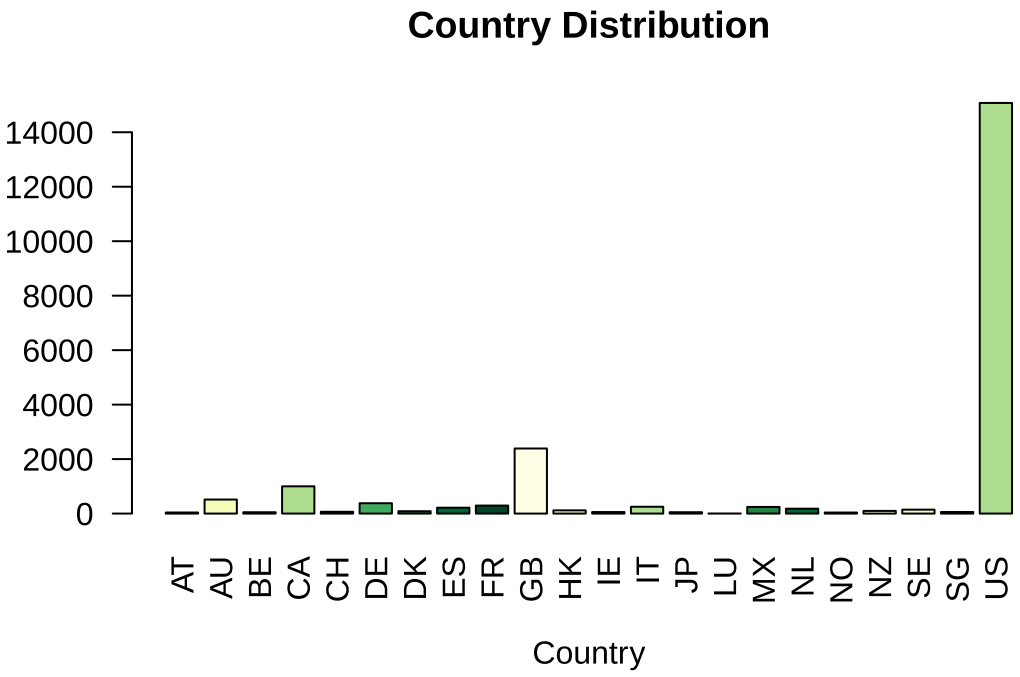
```
#split the data into training and testing sets
set.seed(10)

# use the 70/30 rule: set the training dataset size to be 70% of data size and testing to be 30%
train.size <- round(dim(data) * 0.7)
index.t <- sample(nrow(data), train.size)
train <- data[index.t, ] #dim(train) = 14946    15
test <- data[-index.t, ] #dim(test) = 6405    15
```

Exploratory Data Analysis

##Categorical Variables

As shown below, 22 countries are represented in the dataset. 71% of the projects are based in the US.

```
#Bar chart of Countries
counts <- table(data$country)
barplot(counts, main="Country Distribution",
   xlab="Country", las=2, col=brewer.pal(22, name="YlGn"))
```



Among the 160 categories included in the dataset, "Web" is the most popular, with 429 projects. The top 10 categories, grouped by `successful`, are shown below.

```
#category EDA
cat_counts <- table(data$category)
names(cat_counts)[which.max(cat_counts)]
```

```
## [1] "Web"
```

```
max(cat_counts)
```

```
## [1] 464
```

```
names(cat_counts)[which.min(cat_counts)]
```

```
## [1] "Taxidermy"
```

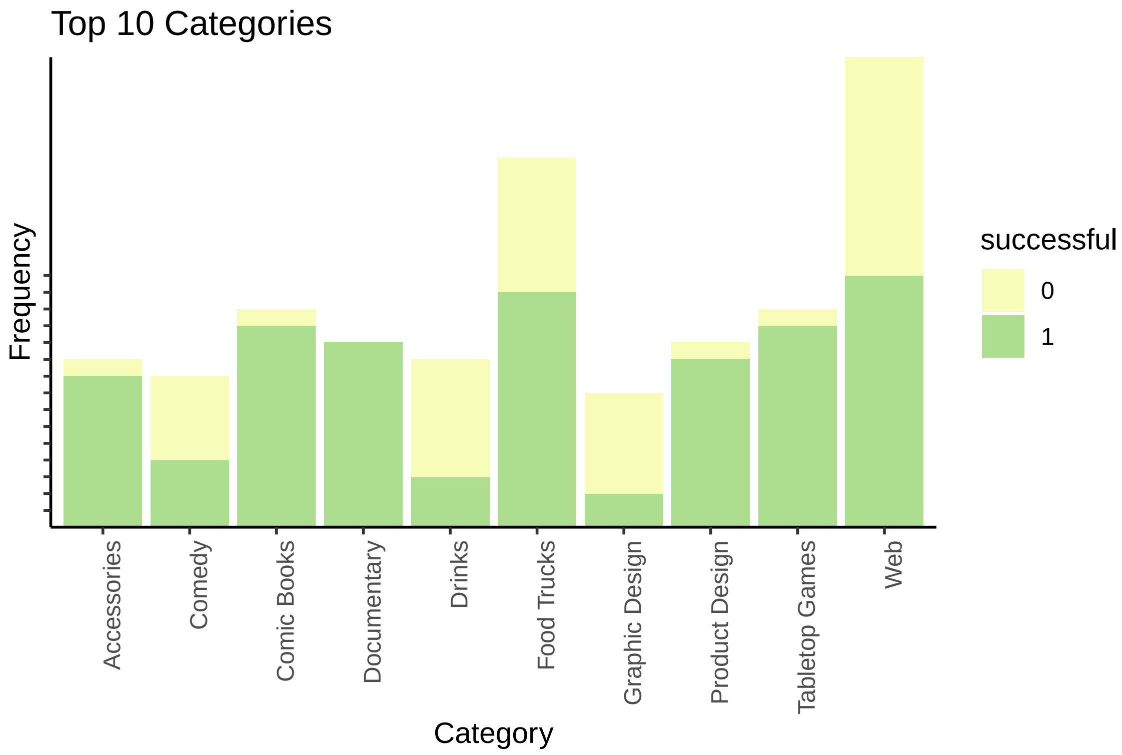
```
min(cat_counts)
```

```
## [1] 1
```

```
cat_counts.df <- data.frame(cat_counts)

cat.top.10 <- cat_counts.df %>% top_n(10)
```

```
## Selecting by Freq
```

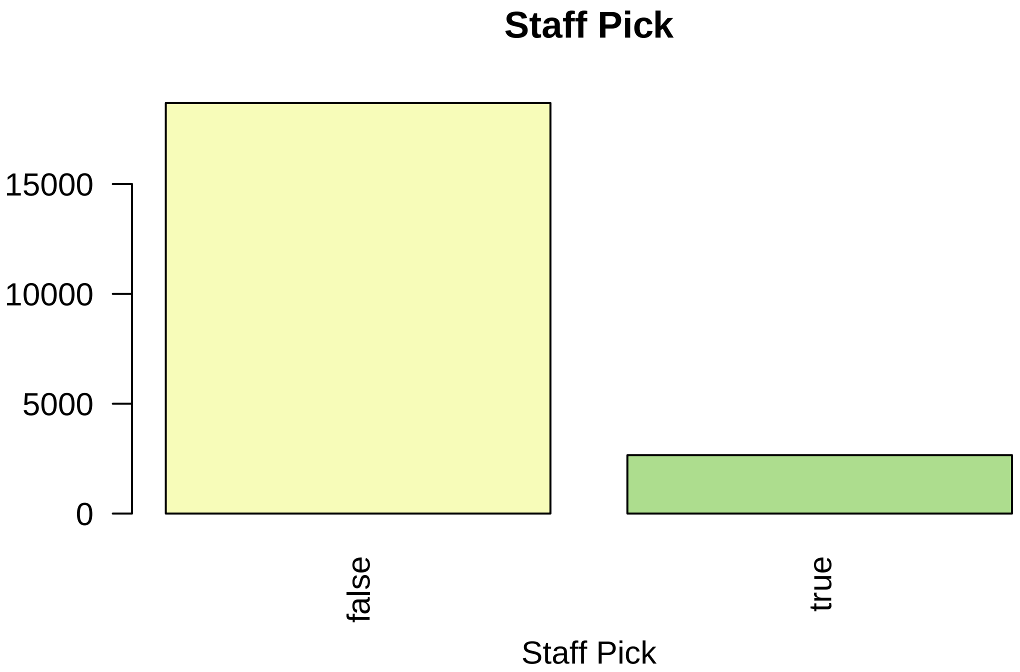
```
cat.top.10 <- data %>% select(successful, category) %>% group_by(category) %>% count() %>% arrange(desc

cat.top.10.grouped <- data %>% filter(category %in% cat.top.10$category) %>% group_by(category, successi

cat.add <- data.frame(c("Accessories", "Comic Books", "Product Design", "Tabletop Games"),
                      c("0", "0", "0", "0"),
                      c("0", "0", "0", "0"))
names(cat.add) <- c("category", "successful", "n")

cat.top.10.grouped <- rbind(cat.top.10.grouped, cat.add)

ggplot(cat.top.10.grouped, aes(x=category, y=n, group=successful, fill=successful)) + geom_bar(stat="id
```



The most popular successful category appears to be Hip-Hop, while the least

The bar chart of `staff_pick` shows that most projects are not chosen to be featured by the staff as part of the "Projects We Love" program. This suggests that the projects that are chosen are particularly featureable.

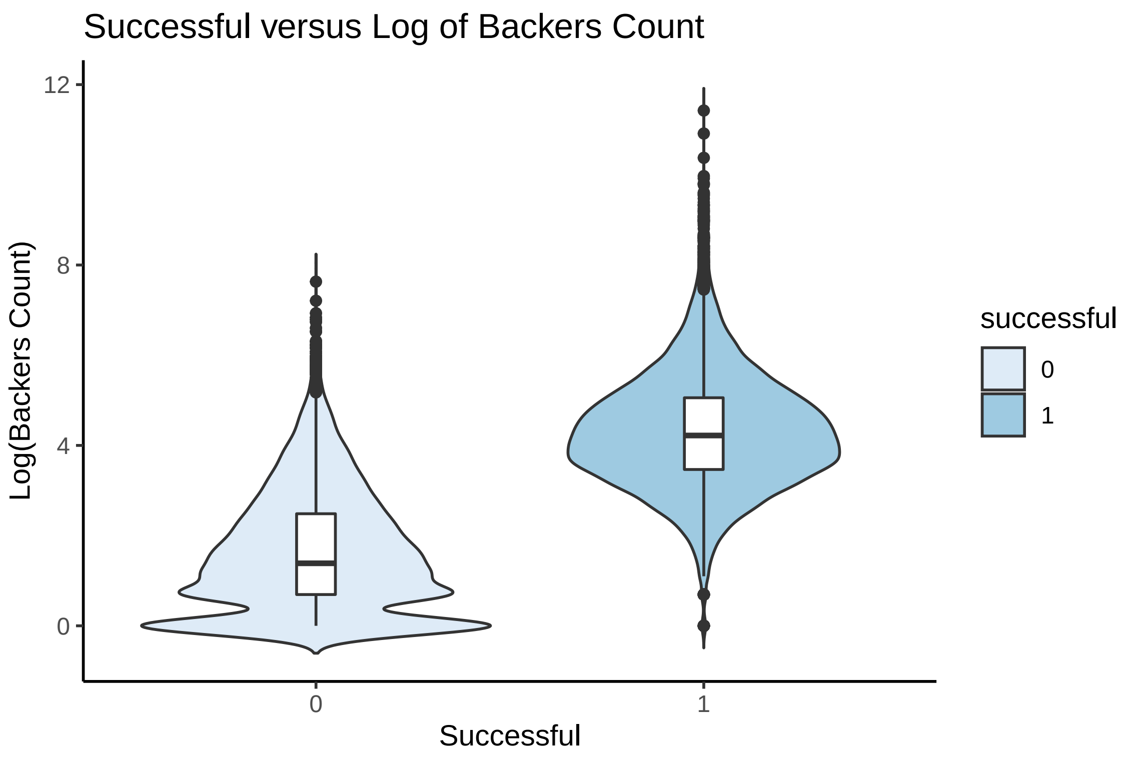
```
#Bar chart of staff_pick
counts <- table(data$staff_pick)
barplot(counts, main="Staff Pick",
   xlab="Staff Pick", las=2, col=brewer.pal(2, name="YlGn"))
```



##Continuous Variables Now, we want to analyse several continuous variables. First we look at the distribution of `backers_count`. The distribution of `backers_count` is highly skewed right. The mean is 139 backers, with a large standard deviation of 980 backers. The median is 26 backers, with a large interquartile range of 86-3=83 backers. Since the minimum number of backers is 0 while the maximum is 91585, `backers_count` has a wide range. Because of this high range, the violin plot below only shows `backers_count` under 500 people.

Below is a violin plot seperating out the data by their `successful` tag. For visualization purposes, we took the log of `backers_count`. Looking at the plots, we see that the median `backers_count` is significantly higher for successful projects than for unsuccessful ones.

```r
#violin plot of log backers count
ggplot(data, aes(x=successful, y=log(backers_count), group=successful, fill=successful)) + geom_violin(
```



```
summary(data$backers_count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     3.0    26.0   135.2    86.0 91585.0
```

```
sd(data$backers_count)
```

```
## [1] 957.3231
```

Next, we look at `goal`. Like `backers_count`, the distribution of `goal` is skewed right. The mean fundraising goal is about $42900 with a large standard deviation of $978141. The median is about $5000, with a large IQR of 15000-1500=$13500. The large range of $100000000 also shows that `goal` differs greatly among projects. Taking a look at the violin plot below, we see that the median goal for unsuccessful projects is higher than successful projects. This suggests that projects with more ambitious goals are less likely to succeed.

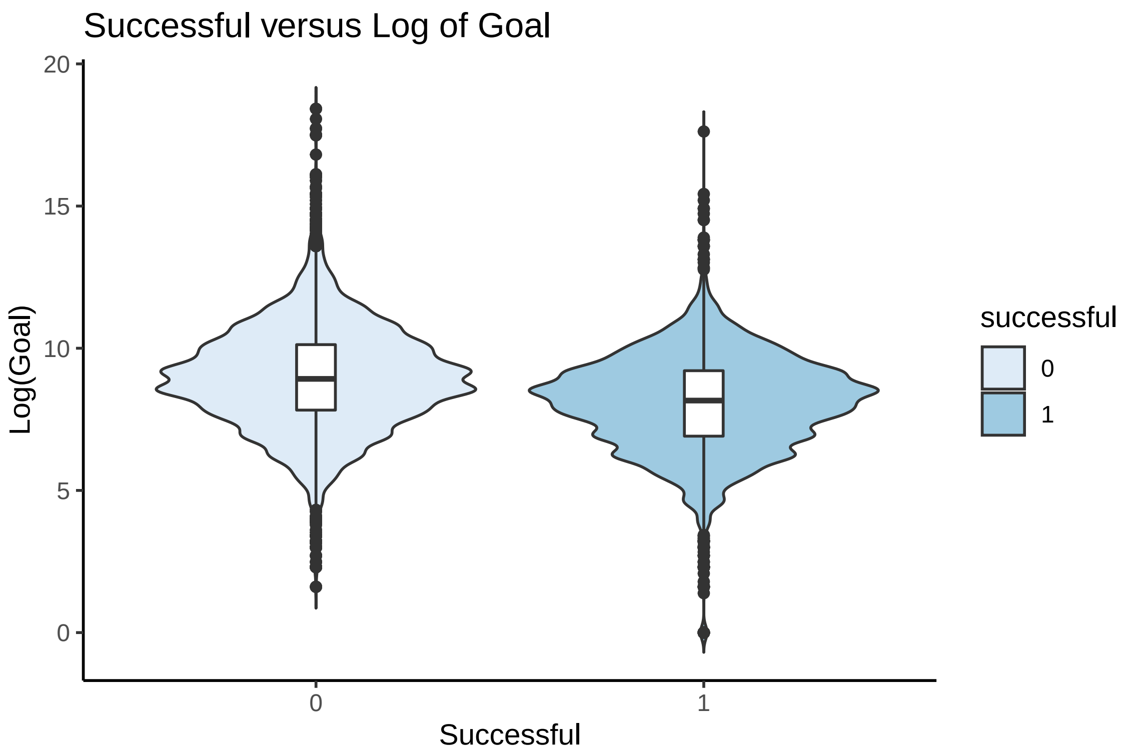
```
summary(data$goal)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
##         1      1500      5000     56082     15000 100000000
```

```
sd(data$goal)
```

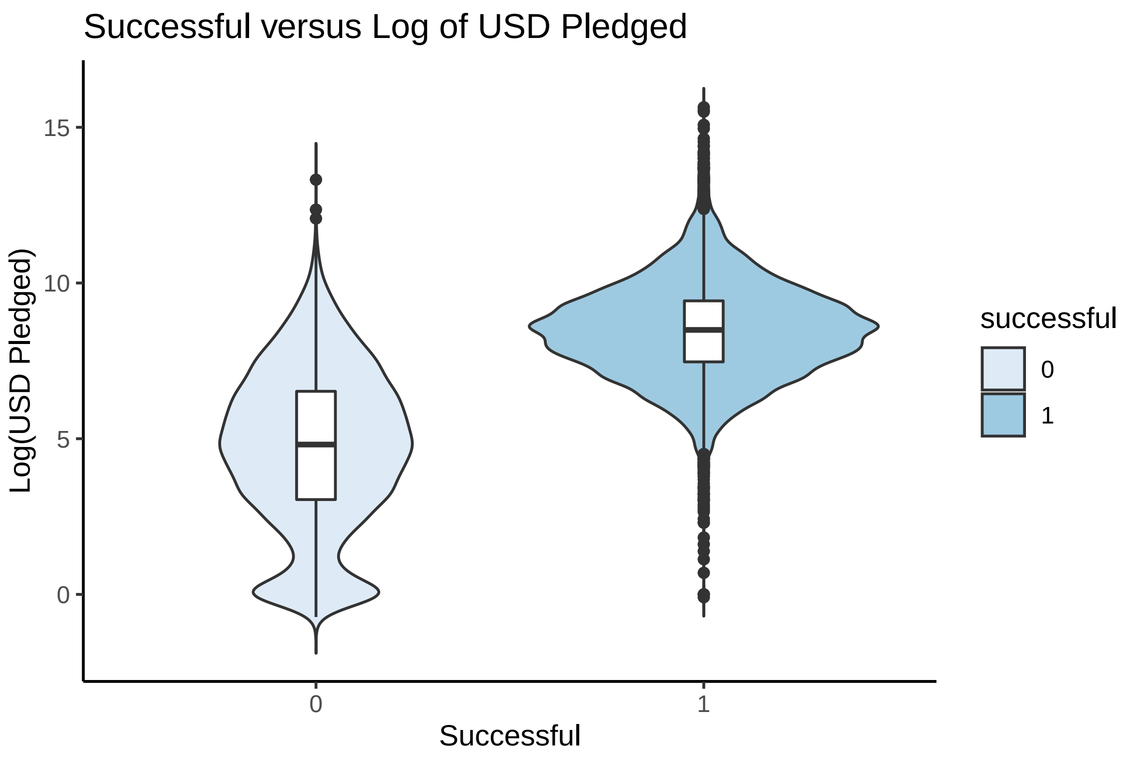
```
## [1] 1336167
```

```
#violin plot of goal
ggplot(data, aes(x=successful, y=log(goal), group=successful, fill=successful)) + geom_violin(trim=FALSE
```



For `usd_pledged`, the mean is $12975, with a large standard deviation of $96928. The median is $1500 with a large IQR of 6381-97=$6284. The range is also large, $6225355. Looking at the violin plots below, we see a much higher amount pledged for successful projects than for unsuccessful projects. The distribution for amount pledged for unsuccessful pledges is relatively wider than successful projects.

```
#histogram of usd_pledged
ggplot(data, aes(x=successful, y=log(usd_pledged), group=successful, fill=successful)) + geom_violin(tri
```



```
summary(data$usd_pledged)
```

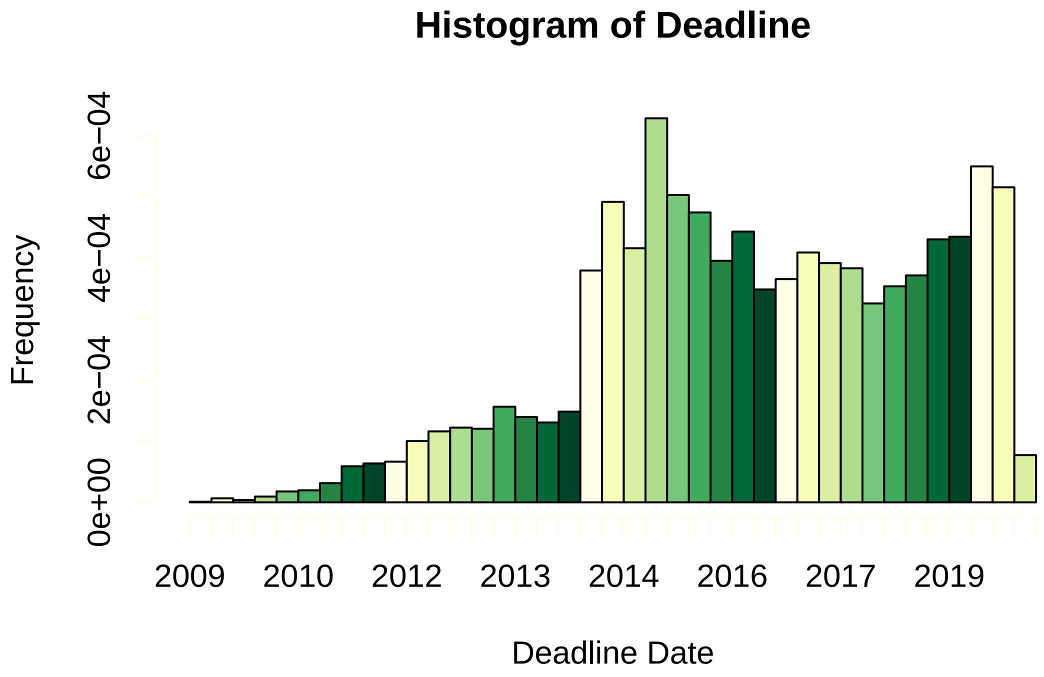
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     100    1470   12530    6339 6225355
```

```
sd(data$usd_pledged)
```

```
## [1] 95906.28
```

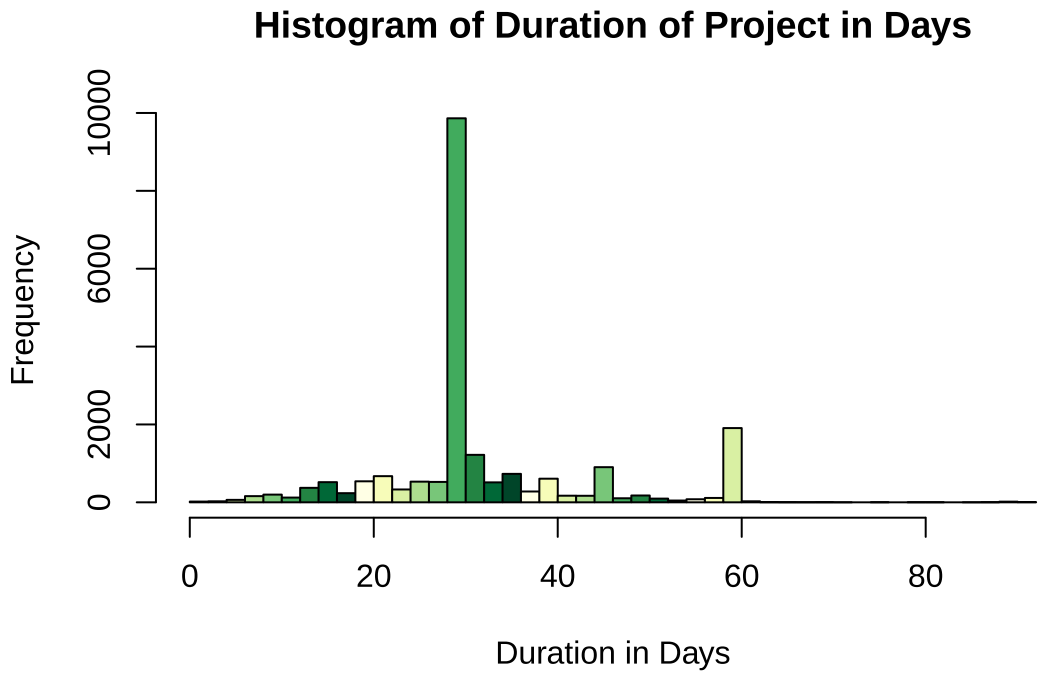
Now taking a look at `deadline`, as the histogram below shows, the distribution of deadlines for projects is skewed left, which is expected since the projects are those that exist in November 2019. The majority of projects have a deadline in the range 2014-2019. Late 2014 to early 2015 appears to have the most deadlines for projects.

```
#histogram of deadlines
histogram <- hist(data$deadline, breaks = 40, xlab = "Deadline Date", ylab = "Frequency",
               main = "Histogram of Deadline",col = brewer.pal(30, name="YlGn"))
```



The histogram of `duration_days` shows two modes, one at around 30 days and the other at around 60 days. This suggests that most projects are live for about a month, while others are live for about 2 months. The mean is about 32.7 days, with a standard deviation of 12 days.

```
#histogram of duration_days
histogram <- hist(data$duration_days, breaks = 40, xlab = "Duration in Days", ylab = "Frequency",
                  main = "Histogram of Duration of Project in Days",col = brewer.pal(30, name="YlGn"))
```



```r
summary(data$duration_days)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   30.00   30.00   32.81   35.00   91.00
```

```r
sd(data$duration_days)
```

```
## [1] 11.89923
```

Model Overview

For the first model, an Elastic Net is used to limit the size of the coefficient vector and impose sparcity among the coefficients. To locate a set of important features, a logistic regression using `glm()` is run with the variables obtained from Elastic Net on the training dataset.

For the final model, we looked to include text analysis on the `blurb` column of the data. We first created a document term-matrix and then took 165 mutually commom words as inputs. We then reran LASSO to limit the coefficients within our model, which we then inputted as variables into a logistic regression model using `glm()`.

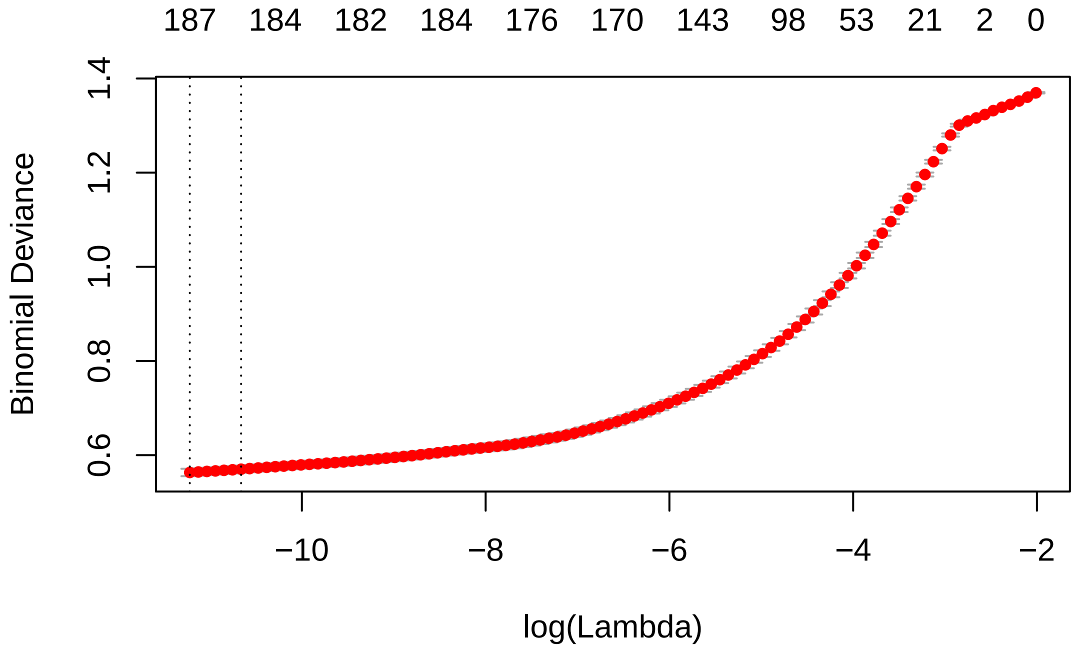
Additionally, we also apply a Random Forest model on the dataset. Using the Ranger R package for efficient implementation of Random Forests, we hypertune the parameter selection size (mtry) per resampled tree to arrive at a nonparametric tree-based prediction model.

In all of the following models, `location` is excluded because `country` is a more general variable that describes the location in which a project is based and including both would lead to high colinearity.

Models

Elastic Net

```
set.seed(10) # to have same sets of K folds
fit1.cv <- cv.glmnet(X, Y, alpha=.9, family="binomial", nfolds = 10, type.measure = "deviance")
plot(fit1.cv)
```



fit1.cv$lambda.1se is chosen because both fit1.cv$lambda.min and fit1.cv$lambda.1se return the same number of variables.

Replacing the variables for which factor levels (`categoryPunk`, `countryMX`) are outputted by the above Elastic Net with the corresponding categorical variable, the following eight variables are outputted by Elastic Net: `category`, `backers_count`, `country`, `deadline`, `duration_days`, `staff_pick`, `goal`, and `usd_pledged`.

Logistic Regression

```
Anova(fit.logit.1)
```

Since all variables are significant at the 0.01 level, the final logistic model without the text variables include the following eight variables: `category`, `backers_count`, `country`, `deadline`, `duration_days`, `staff_pick`, `goal`, and `usd_pledged`.

Evaluate the Logistic Regression Model

To evaluate the logistic regression model, `fit.logit.1`, the 1/2 thresholding rule is used for predicting `successful` on the testing data.

The testing missclassification error of the logistic regression model is about 0.087. Not bad!

Random Forest

```
RF.all <- ranger(successful~., data.for.train, mtry=4, # Although recommended mtry is p=n/3=15/3=5; we
                 num.trees = 250, splitrule = "gini",importance = "impurity")
RF.pred <-predict(RF.all, data.for.test, type = "response")
RF.test.error <- mean(data.for.test$successful!=RF.pred$predictions)

RF.test.error
```

```
## [1] 0.01795472
```

```
RF.all$prediction.error
```

```
## [1] 0.0177974
```

Text Mining

As an alternative model, we wanted to explore whether or not we could utilize text data contained within the `blurb` column to predict whether or not a company will be successful.

First, we isolated the text data:

```
data1.text <- data$blurb
print(data1.text[1:5]) # view a few documents
```

```
## [1] "I'm recording my ninth studio album with some outstanding musicians to release it on digital do
## [2] "Lost since 1946, this is the first reprint of the wild sci-fi adventures of Brok Windsor: Docto
## [3] "The one place someone of any age can go to learn, teach, or just DANCE. Education and fitness b
## [4] "A romantic thriller that depicts the turmoil of addiction and the power of music.  This is not 
## [5] "A coffee table book of photographs and stories of people who dreamt of an adventure and escaped
```

Overall, it appears as if the blurbs are roughly the same length. This is most likely due to the fact that there is a restriction on length for these blurbs on Kickstarter.

Corpus: a collection of text

Here, we move onto creating our corpus.

```
mycorpus1 <- VCorpus( VectorSource(data1.text))
mycorpus1
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 21351
```

Data cleaning using `tm_map()`

Before we transform our text into a word frequency matrix, we needed to standardized the text. Similar to our model in class, we used `removeNumbers()`, `removePunctuation()`, `removeWords()`, `stemDocument()`, and `stripWhitespace()` to go about cleaning our data.

```
#cleaning
mycorpus_clean <- tm_map(mycorpus1, content_transformer(tolower))
mycorpus_clean <- tm_map(mycorpus_clean, removeWords, stopwords("english"))
mycorpus_clean <- tm_map(mycorpus_clean, removePunctuation)
mycorpus_clean <- tm_map(mycorpus_clean, removeNumbers)
mycorpus_clean <- tm_map(mycorpus_clean, stemDocument, lazy = TRUE)
```

Word frequency matrix

Now we transform each review into a word frequency matrix using the function `DocumentTermMatrix()`.

```
dtm1 <- DocumentTermMatrix( mycorpus_clean )   ## library = collection of words for all documents
```

Intially, there are a total of 26541 words that could be utilized as predictors. This is clearly, too many, so we decided to cut down the bag to only include the words appearing at least 1% (or the frequency of your choice) of the time to reduce the dimension of the features extracted to be analyzed.

```
threshold <- .01*length(mycorpus_clean)   # 1% of the total documents
words.10 <- findFreqTerms(dtm1, lowfreq=threshold)  # words appearing at least among 1% of the document
```

```
dtm.10<- DocumentTermMatrix(mycorpus_clean, control = list(dictionary = words.10))
dim(as.matrix(dtm.10))
```

```
## [1] 21351   167
```

```
colnames(dtm.10)[1:25] #preview
```

```
##  [1] "adventur" "age"      "album"    "american" "anim"     "app"
##  [7] "around"   "art"      "artist"   "back"     "band"     "base"
## [13] "beauti"   "becom"    "best"     "book"     "brand"    "bring"
## [19] "build"    "busi"     "can"      "card"     "celebr"   "chang"
## [25] "children"
```

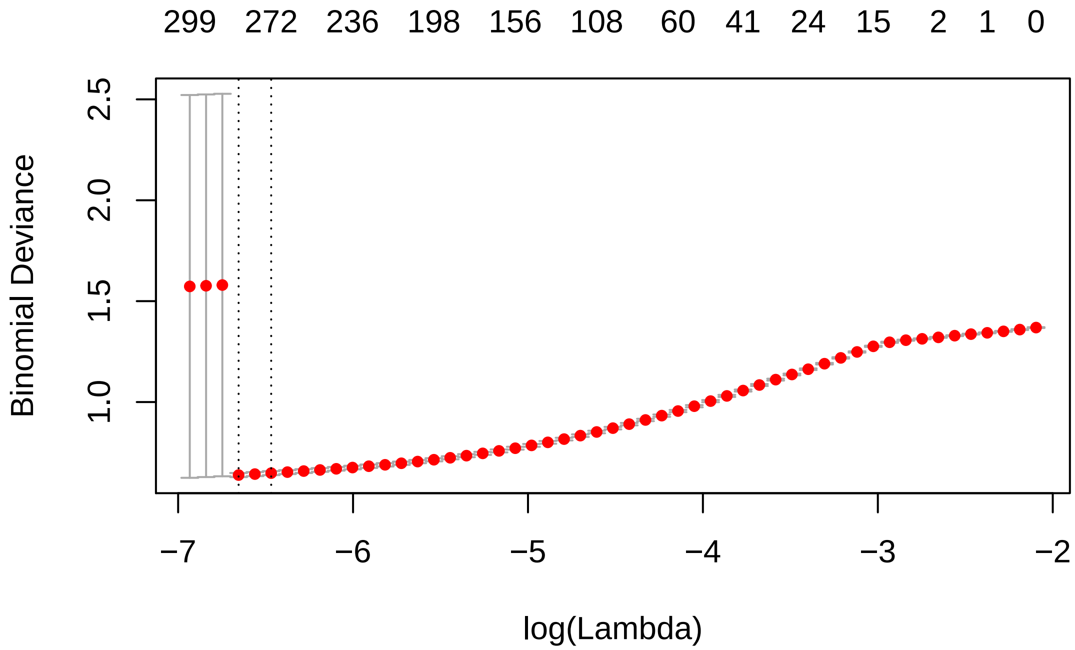
In the end, we ended up with 167 possible words to utilize as predictors.

Data preparation

Next, we used lasso to see what words are actually meaningful predictors for our data.

###LASSO

```
plot(result.lasso.text)
```



```
beta.lasso <- coef(result.lasso.text, s="lambda.1se")    # output lasso estimates
beta <- beta.lasso[which(beta.lasso !=0),] # non zero beta's
beta <- as.matrix(beta)
beta <- rownames(beta)
#non-word-frequency features
non_word_beta <- beta[str_detect(beta, "backers_count|category|country|deadline|launched_at|duration_day

#word-frequency features
word_beta <- beta[!str_detect(beta, "backers_count|category|country|deadline|launched_at|duration_days|
length(word_beta)
```

```
## [1] 115
```

In order to get the number of words it preserved, we first filtered out any features that were not related to word frequency. We then saw that lasso preserved 161 words out of 167, which is still quite a significant amount. This suggests that there may not be that much common difference in distribution in word frequencies between successful and failed projects. Nevertheless, we decided to run a logistic regression on the word frequencies.

```
# Input the words from LASSO into glm
glm.input <- as.formula(paste("successful~category+country+duration_days+goal+usd_pledged+staff_pick+",
result.glm <- glm(glm.input, family=binomial, data.text.train ) # instantly

# Testing errors

data.text.test <- data.text.test %>% filter(category != "Taxidermy")
predict.glm <- predict(result.glm, data.text.test, type = "response")
class.glm <- rep("0", nrow(data.text.test))
class.glm[predict.glm > .5] <- "1"
```

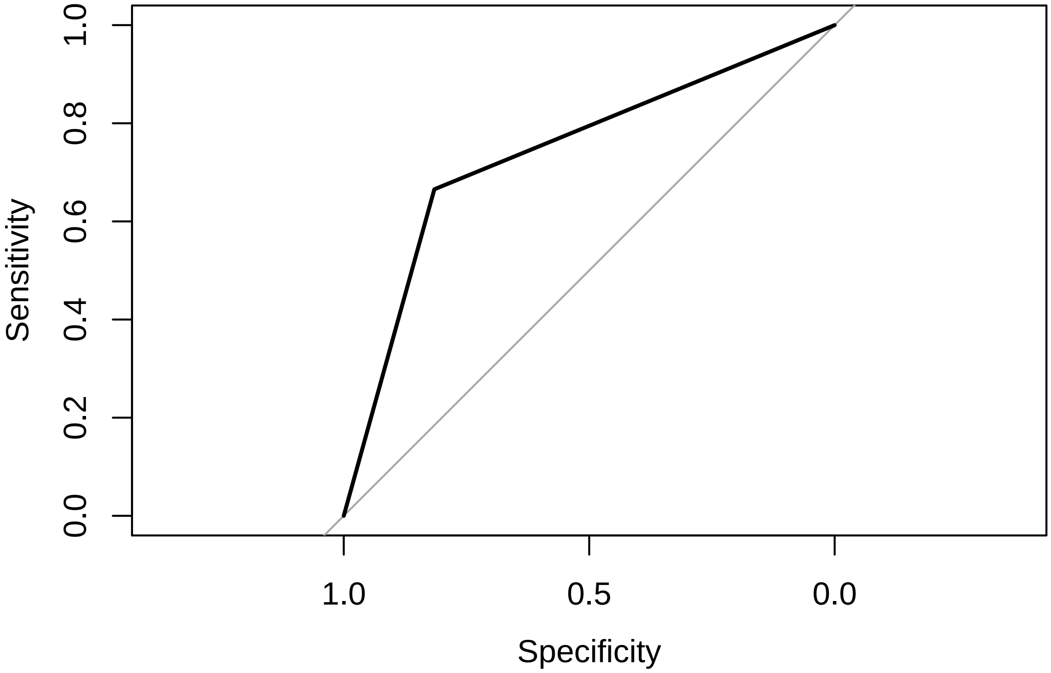
```
testerror.glm <- mean(data.text.test$successful != class.glm)
testerror.glm
```

```
## [1] 0.2686963
```

```
pROC::roc(data.text.test$successful, predict.glm, plot=T)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = data.text.test$successful, predictor = predict.glm,     plot = T)
##
## Data: predict.glm in 2812 controls (data.text.test$successful 0) < 3593 cases (data.text.test$succes
## Area under the curve: 0.7404
```

Adding in text data,we actually saw our testing error increase to 0.1188. However, taking a look at our ROC, we saw a ROC of 95.4%, meaning that the data did very well in predicting our test data!

Comparison of Methods

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
fit1.roc <-roc(test$successful, as.numeric(fit.logit.pred)) #Hiroyi
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

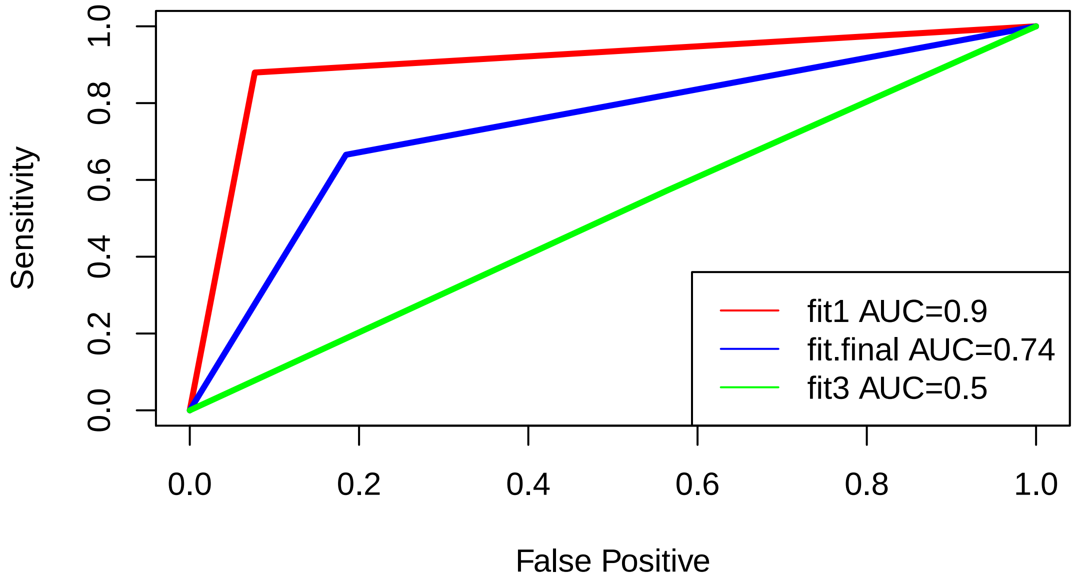
```r
fit2.roc <-roc(data.text.test$successful, predict.glm) #Younghu
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
fit3.roc <-roc(data.text.test$successful, as.numeric(RF.pred$predictions)) #Hoyt
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(1-fit1.roc$specificities,
     fit1.roc$sensitivities, col="red", lwd=3, type="l",
     xlab="False Positive",
     ylab="Sensitivity")
lines(1-fit2.roc$specificities, fit2.roc$sensitivities, col="blue", lwd=3)
lines(1-fit3.roc$specificities, fit3.roc$sensitivities, col="green", lwd=3)
legend("bottomright",
       c(paste0("fit1 AUC=",round(fit1.roc$auc,2)),
         paste0("fit.final AUC=",round(fit2.roc$auc, 2)),
         paste0("fit3 AUC=",round(fit3.roc$auc, 2))),
       col=c("red", "blue", "green"),
       lty=1)
```



The MCEs of the models are as follows: | Model | MCE | |—–|—–| |Logistic Regression without Text| 0.087 | | Random Forest | 0.080 | | Logistic Regression with Text | 0.118 |

According to MCE, the most accurate model is the Random Forest model because it has the lowest MCE. However, according to the AUC scores of the final ROC plots, we determine that the overall most sensitive and specific model is the Logistic Regression with Text. The ROC curve plots how a discrimination threshold for classification as successful or failure, varied across different values, performs as a predictive model with regards to false positive and true positive diagnostic ability. As this Logistic Regression with Text model holds the highest AUC score from the ROC, we decide to ultimately select this model as our final model.

Final Model

Overall, we determine the final model across our various trails is the feature-engineered logistic regression that holds the text. Via text mining on the company names and description, we were able to include the additional information from text data in our predictive model. These additional features we engineered allowed us to construct a more accurate logisitc model after accounting for regularization from the Elastic Net that we also applied. The process from text mining to applying LASSO/Elastic Net to finally assessing the accuracy of our model via the MCE and aucROC allow us to conclude the important nature of text data and the significant company statistics in predicting the success of a kickstarter project.

Some limitations of this model include the following. First, only data from November was used because including more data would have been too computationally overwhelming. Similarly, only a randomly sampled quarter of the November data was used for computational reasons. However, even after limiting the dataset, a significant amount of data (21351 observations) was used. Moreover, our model is ultimately predicting on the final state of the projects as opposed to predicting while the project is ongoing. A more sophisticated analysis would probably involve predicting the model from its beginning state and utilizing the rate of changes over time in its features like `backers_count` and do real-time predictions on whether or not the project is successful.

Future improvements could include using data across several years or training a Bootstrap or Neural Network model.

Conclusion

The final model suggests that future Kickstarter creators may increase their chances of success by focusing on the category and the country in which the project is based. Creators should try to limit the number of days to keep their project live since an increase in `duration_days` decreases the log odds of success. Having an earlier deadline, i.e. closer to the current date, increases the log odds of success, so creators should avoid setting deadlines too far into the future. Since a dollar increase in the funding goal decreases the log odds of success, creators should avoid setting goals that are too high compared to other projects on Kickstarter. Creators should aim to get their projects featured in the “Projects We Love” program, as `staff_pick` being true increases the log odds of success. Finally, as expected, creators should aim to obtain more backers and pledged money, as increases in both of these increases the log odds of success.

Further analysis on word frequency differences can be found within the Appendix.

Overall, this project shows that success of Kickstarter projects can be relatively accurately predicted using public features of the projects.

Appendix

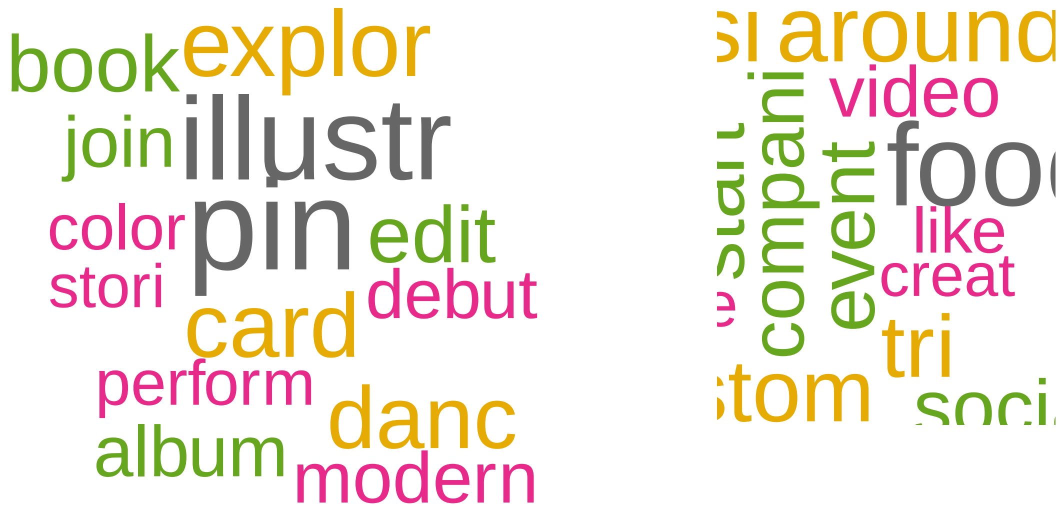
##Word Cloud Analysis

```r
#Using only the docmuent term matrix
data.cloud.temp <- data.frame(as.matrix(dtm.10))
data.cloud.temp$successful <- data$successful
data.cloud <- data.cloud.temp %>% select(successful, everything())
```

```r
# Input the words found by our dtm into glm
glm.input <- as.formula(paste("successful", "~", paste(names(data.cloud)[-1],collapse = "+")))
result.glm <- glm(glm.input, family=binomial, data.cloud)
```

```r
result.glm.coef <- coef(result.glm)
# pick up the positive coef's which are positively related to the prob of being a good review
success.glm <- result.glm.coef[which(result.glm.coef > 0)]
success.glm <-success.glm[-1]  # took intercept out
cor.special <- brewer.pal(8,"Dark2")  # set up a pretty color scheme
success.fre <- sort(success.glm, decreasing = TRUE) # sort the coef's
# hist(as.matrix(good.fre), breaks=30, col="red")
success.word <- names(success.fre)  # good words with a decreasing order in the coeff's
fail.glm <- result.glm.coef[which(result.glm.coef < 0)]
# names(bad.glm)[1:50]
cor.special <- brewer.pal(6,"Dark2")
fail.fre <- sort(-fail.glm, decreasing = TRUE)
# hist(as.matrix(bad.fre), breaks=30, col="green")
fail.word <- names(fail.fre)
par(mfrow=c(1,2))
cor.special <- brewer.pal(8,"Dark2")
wordcloud(success.word[1:20], fail.fre[1:20],
          colors=cor.special, ordered.colors=F)
```

```r
wordcloud(fail.word[1:20], fail.fre[1:20],
          color=cor.special, ordered.colors=F)
```



```r
par(mfrow=c(1,1))
```

Here, we see that “pin”, which indicates whether or not it was on the front page, was the highest scoring word. This word was most likely autogenerated by Kickstarter and placed into the blurb for the projects that were doing well. Further analysis would look to probably remove this word in order to not give the model an advantage. Interestingly, we see words related to art/culture such as “comic” “modern” enamel" and “illust” to be dominant in successful businesses, whereas words related to business such as “app”, “busi”, “even”, “product” were more common to the projects that failed. This seems to suggest that companies with more creative/art focused projects do better on kickstarter as opposed to “strictly business” companies.

Works Cited

“Creator Handbook.” Kickstarter, https://www.kickstarter.com/help/handbook?ref=whatarebasics.

“Kickstarter Datasets.” Web Scraping Service, 25 Nov. 2019, https://webrobots.io/kickstarter-datasets/.

“What Are the Basics?” Kickstarter Support, https://help.kickstarter.com/hc/en-us/articles/115005028514-What-are-the-ba