PH125.9x Data Science Capstone - Kaggle Kickstarter Dataset Analysis and Predictions

Marina Ganopolsky

12/3/2020

Contents

Overview

This is Marina Ganopolsky's implementation of the **Choose Your Own Project** of the **Harvard: PH125.9x Data Science, Summer 2020: Capstone** course.

The objectives of this project are:

1. To find a dataset on which to conduct Exploratory Data Analysis (EDA), Feature Engineering, and on which to perform **machine learning tasks**.
2. To use as least 2 methods beyond linear regression during the project.
3. To achieve an accuracy of **> 50%**, and validate this with the **test data set**, as described below.
4. To summarize the findings, draw conclusions, and provide further recommendations, if needed.

The dataset I have chosen is the Kickstarter Dataset from Kaggle. This dataset represents Kickstarter projects between the years of **2009 and 2018**, as well as their respective success and failure information. Information about the country of origin of the project, the category, the goal amount, currency, etc is also present. A quick summary of the data wrangling modifications will be provided, as well as some visual representations of patterns present in the data, in the **Method and Analysis** section. This will inform the reader about the various trends present in the dataset.

An exploratory analysis will be performed to generate the predicted succeses levels of the Kickstarter Project, with various models, to be specified; a best-performing algorithm will be chosen, and the results of the calculations will be analyzed and explained; finally a conclusion will be provided.

Several models will be developed, ranging from the most naive to the most complex. These will be tested and compared using the resulting RMSE in order to assess their quality. In contrast to the MovieLens project there is no ideal evaluation criteria for this project. However, since we have several models available, and we will pick the model with the highest accuracy as the chosen model to produce the accuracy of the model.

The data is broken into a **training dataset** (80%), and a **testing dataset**(20%). After evaluating every available algorithm I have come up with, the best-resulting model will be used on the **test dataset** to evaluate the quality of the predictions of the movie ratings.

This project can be found on **GitHub** here .

Dataset

The **Kickstarter** dataset is automatically downloaded with the code provided; The data sets can be additionally downloaded here:

- https://raw.githubusercontent.com/mganopolsky/kickstarter/master/data/ks-projects-201801.csv

As specified by the project description (and the provided code) the data is split into a **test set (20%)** and the **training set(80%)**, after some feature engineering is applied to it. The calculations and algorithm verification are done on the training set, and each algorithm is tested on the test set.

As a first step, in order to get some basic information about the data we're working with, we examine the **`kickstarter`** dataset. The original dataset contains 15 variables :

```
glimpse(data)
```

```
Rows: 378,661
Columns: 15
$ ID               <dbl> 1000002330, 1000003930, 1000004038, 1000007540, 10...
$ name             <chr> "The Songs of Adelaide & Abullah", "Greeting From ...
$ category         <chr> "Poetry", "Narrative Film", "Narrative Film", "Mus...
$ main_category    <chr> "Publishing", "Film & Video", "Film & Video", "Mus...
$ currency         <chr> "GBP", "USD", "USD", "USD", "USD", "USD", "USD", "...
$ deadline         <date> 2015-10-09, 2017-11-01, 2013-02-26, 2012-04-16, 2...
$ goal             <dbl> 1000, 30000, 45000, 5000, 19500, 50000, 1000, 2500...
$ launched         <dttm> 2015-08-11 12:12:28, 2017-09-02 04:43:57, 2013-01...
$ pledged          <dbl> 0.00, 2421.00, 220.00, 1.00, 1283.00, 52375.00, 12...
$ state            <chr> "failed", "failed", "failed", "failed", "canceled"...
$ backers          <dbl> 0, 15, 3, 1, 14, 224, 16, 40, 58, 43, 0, 100, 0, 0...
$ country          <chr> "GB", "US", "US", "US", "US", "US", "US", "US", "U...
$ `usd pledged`    <dbl> 0.00, 100.00, 220.00, 1.00, 1283.00, 52375.00, 120...
$ usd_pledged_real <dbl> 0.00, 2421.00, 220.00, 1.00, 1283.00, 52375.00, 12...
$ usd_goal_real    <dbl> 1533.95, 30000.00, 45000.00, 5000.00, 19500.00, 50...
```

```
#There aree 378,661 rows


#according to the documentatino, `usd pledged` isn't needed since there is the usd_ple
```

A fair amount of data manipulation has been done to the datasets created with the provided code. The changes include:

- **New Field** : Adding a time interval in days, as the difference between the launch date and the deadline date : **time_int**
- **New Field** : Extracting the month of the year from the launch date, as saving it as a factor in **launched_month**
- **New Field** : Extracting the year from the launch date, as saving it as a factor in **launched_year**

- **New Field** : Extracting the day of the week from the launch date, as saving it as a factor in **launched_day_of_week**
- **New Field** : Added a field as a rounded numeric representing the pledge:goal ratio in USD **pledged_ratio**
- **New Field** : Added a field as a rounded numeric representing the average pledge per backer in USD **avg_backer_pldg**
- Changed the **launched** field into a date (as opposed to a date/time field)
- **Removed Field** : Removed the **`usd pledged`** field, as there was the usd_pledged_real represented what was actually needed.
- After the data wrangling and data visualization exercises (see below) are complete, the dataset is split up into :
  - **A training dataset** - the data the algorithms will be trained on, **80%** of the remaining data.
  - **A testing dataset** - the data the algorithms will be tested on, **20%** of the remaining data.

Per row, this is a representation of a single Kickstarter campaign.
As a first glance, here's a snapshot of dataset:

```
       ID                 name              category         main_category
 Min.   :5.971e+03   Length:378661      Length:378661      Length:378661
 1st Qu.:5.383e+08   Class :character   Class :character   Class :character
 Median :1.075e+09   Mode  :character   Mode  :character   Mode  :character
 Mean   :1.075e+09
 3rd Qu.:1.610e+09
 Max.   :2.147e+09

    currency          deadline                goal              launched
 USD    :295365   Min.   :2009-05-03   Min.   :        0   Min.   :1970-01-01
 GBP    : 34132   1st Qu.:2013-06-08   1st Qu.:     2000   1st Qu.:2013-05-07
 EUR    : 17405   Median :2015-01-14   Median :     5200   Median :2014-12-10
 CAD    : 14962   Mean   :2014-11-01   Mean   :    49081   Mean   :2014-09-28
 AUD    :  7950   3rd Qu.:2016-04-28   3rd Qu.:    16000   3rd Qu.:2016-03-24
 SEK    :  1788   Max.   :2018-03-03   Max.   :100000000   Max.   :2018-01-02
 (Other):  7059
    pledged             state              backers           country
 Min.   :       0   Length:378661      Min.   :     0.0   Length:378661
 1st Qu.:      30   Class :character   1st Qu.:     2.0   Class :character
 Median :     620   Mode  :character   Median :    12.0   Mode  :character
 Mean   :    9683                      Mean   :   105.6
```

```
 3rd Qu.:    4076                      3rd Qu.:    56.0
 Max.   :20338986                      Max.   :219382.0

 usd_pledged_real   usd_goal_real          time_int        pledged_ratio
 Min.   :       0   Min.   :        0   Min.   :    1.00   Min.   :     0.00
 1st Qu.:      31   1st Qu.:     2000   1st Qu.:   30.00   1st Qu.:     0.00
 Median :     624   Median :     5500   Median :   30.00   Median :     0.13
 Mean   :    9059   Mean   :    45454   Mean   :   34.48   Mean   :     3.24
 3rd Qu.:    4050   3rd Qu.:    15500   3rd Qu.:   37.00   3rd Qu.:     1.07
 Max.   :20338986   Max.   :166361391   Max.   :16739.00   Max.   :104277.89

 avg_backer_pldg    launched_month   launched_day_of_week launched_year
 Min.   :    0.00   07      : 36367   1:61228              2015   :77300
 1st Qu.:   13.00   03      : 33946   2:77307              2014   :67745
 Median :   40.00   10      : 33490   3:67438              2016   :57184
 Mean   :   64.58   11      : 32890   4:60596              2017   :52200
 3rd Qu.:   75.00   05      : 32888   5:58327              2013   :44851
 Max.   :10000.00   06      : 32623   6:32717              2012   :41165
                    (Other):176457   7:21048              (Other):38216
```

The first few lines of the dataset look like this:

```
          ID                                                      name
1 1000002330                                 The Songs of Adelaide & Abullah
2 1000003930                   Greeting From Earth: ZGAC Arts Capsule For ET
3 1000004038                                                Where is Hank?
4 1000007540                ToshiCapital Rekordz Needs Help to Complete Album
5 1000011046 Community Film Project: The Art of Neighborhood Filmmaking
6 1000014025                                           Monarch Espresso Bar
        category main_category currency   deadline  goal   launched pledged
1         Poetry    Publishing      GBP 2015-10-09  1000 2015-08-11       0
2 Narrative Film  Film & Video      USD 2017-11-01 30000 2017-09-02    2421
3 Narrative Film  Film & Video      USD 2013-02-26 45000 2013-01-12     220
4          Music         Music      USD 2012-04-16  5000 2012-03-17       1
5   Film & Video  Film & Video      USD 2015-08-29 19500 2015-07-04    1283
6    Restaurants          Food      USD 2016-04-01 50000 2016-02-26   52375
       state backers country usd_pledged_real usd_goal_real time_int
1     failed       0      GB               0       1533.95       59
2     failed      15      US            2421      30000.00       60
3     failed       3      US             220      45000.00       45
4     failed       1      US               1       5000.00       30
5   canceled      14      US            1283      19500.00       56
6 successful     224      US           52375      50000.00       35
  pledged_ratio avg_backer_pldg launched_month launched_day_of_week
```

```
1          0.00                0          08                   2
2          0.08              161          09                   6
3          0.00               73          01                   6
4          0.00                1          03                   6
5          0.07               92          07                   6
6          1.05              234          02                   5
  launched_year
1          2015
2          2017
3          2013
4          2012
5          2015
6          2016
```

Methods and Analysis

Data Insights, Cleaning & Visualizations

Initial Insights

Glancing over the information most things seem fine at a first review; one thing that jumps out is the minimum value of the **launched** field, which is 1/1/1970. This is probably an error in the data - let's examine it:

```
ds %>% distinct(launched) %>% arrange(launched) %>% head()
```

```
# A tibble: 6 x 1
  launched
  <date>
1 1970-01-01
2 2009-04-21
3 2009-04-23
4 2009-04-24
5 2009-04-25
6 2009-04-27
```

Viewing a list of the furthest ordered launch dates, we see that 1/1/1970 is a strange outlier. We dig further by analyzing the entries with this information:

```
ds %>% filter(launched=='1970-01-01') %>% dplyr::select(launched, name, category, deadli
```

```
# A tibble: 7 x 6
  launched   name                                category  deadline   backers  goal
  <date>     <chr>                               <chr>     <date>       <dbl> <dbl>
1 1970-01-01 "Salt of the Earth: A Dead Sea ~ Film & V~ 2010-09-15        0  5000
2 1970-01-01 "1st Super-Size Painting - Soci~ Art       2010-08-14        0 15000
3 1970-01-01 "\"ICHOR\"  (Canceled)"          Film & V~ 2010-05-21        0   700
4 1970-01-01 "Support Solo Theater! Help \"U~ Theater   2010-06-01        0  4000
5 1970-01-01 "Help RIZ Make A Charity Album:~ Music     2010-05-04        0 10000
6 1970-01-01 "Identity Communications Infogr~ Design    2010-04-10        0   500
7 1970-01-01 "Student Auditions Music 2015"   Publishi~ 2015-10-31        0  1900
```

```
#let's remove the dates on 1/1/1970
```

Based on the names, most of these campaigns have been cancelled. It's worth noting that in the predictive modeling section, our models will exclude projects that do not fall into the "successful" or "failed". Therefor, we will remove these entires from the data.

Pledge Information

What kind of information can be gleaned from failed projects? We examine projects where the pledge ratio is less then 1 (meaning that they did not meet their fundraise goal.)

```
failed <- ds %>% filter(pledged_ratio < 1)

failed %>% distinct(state)
```

```
# A tibble: 6 x 1
  state
  <chr>
1 failed
2 canceled
3 live
4 undefined
5 suspended
6 successful
```

```
#it appeaers that some are succeessful despite having not met their goal!
```

It appears that some projects are succesful despite not having met their funding goal!

```
failed %>% filter(state=="successful")
```

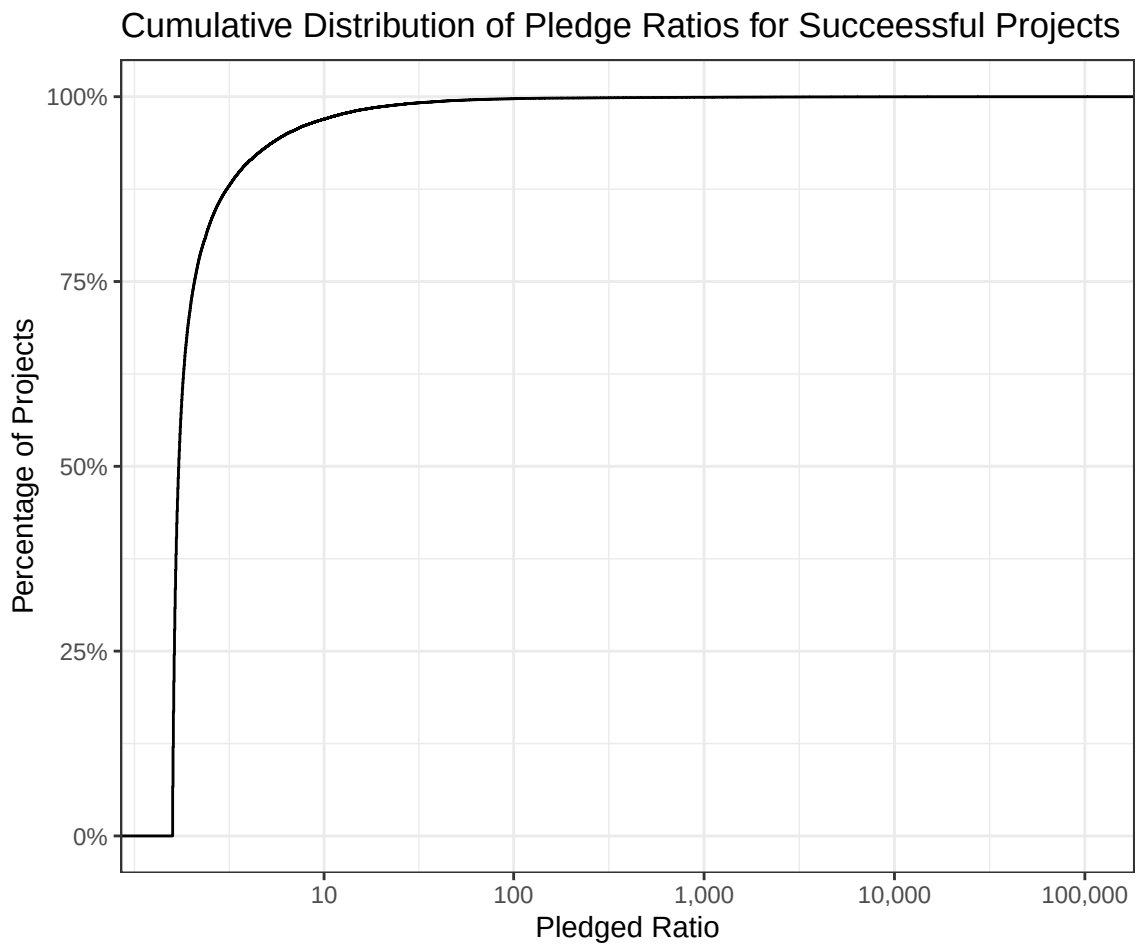
```
# A tibble: 1 x 20
      ID name  category main_category currency deadline    goal launched
   <dbl> <chr> <chr>    <chr>         <fct>    <date>     <dbl> <date>
1 1.77e9 Bord~ Shorts   Film & Video  USD      2015-12-27 36000 2015-11-12
# ... with 12 more variables: pledged <dbl>, state <chr>, backers <dbl>,
#   country <chr>, usd_pledged_real <dbl>, usd_goal_real <dbl>, time_int <dbl>,
#   pledged_ratio <dbl>, avg_backer_pldg <dbl>, launched_month <fct>,
#   launched_day_of_week <fct>, launched_year <fct>
```

```
#interesting , but both proejcets raised a significant maount of money and continued o

#what's the density distribution of the pledge ratio that most successful projectsc ha
```

The project raised a significant sum though - 85% of what it neeeded; and it went on to be successful. However, it's the only one of its kind. This of course, begs the question of what happens

Viewing the pledge ratios' cumulative distribution, **it is obvious that most of the successful projects are funded around 100% or slightly higher - with the ratios hovering around 1. However, there are actually a significant amount of projects that are funded at thousands of times their goal value.**:



Data Manipulation

Since we will be using models that require fields set as factors, the predictive fields of the dataset will be manipulated to be changed into factors.

The levels of the factors of these fields can be viewed here:

```
levels(ds$category)
```

```
  [1] "3D Printing"        "Academic"           "Accessories"
  [4] "Action"             "Animals"            "Animation"
  [7] "Anthologies"        "Apparel"            "Apps"
 [10] "Architecture"       "Art"                "Art Books"
 [13] "Audio"              "Bacon"              "Blues"
 [16] "Calendars"          "Camera Equipment"   "Candles"
 [19] "Ceramics"           "Children's Books"   "Childrenswear"
 [22] "Chiptune"           "Civic Design"       "Classical Music"
 [25] "Comedy"             "Comic Books"        "Comics"
```

```
 [28] "Community Gardens"  "Conceptual Art"     "Cookbooks"
 [31] "Country & Folk"     "Couture"            "Crafts"
 [34] "Crochet"            "Dance"              "Design"
 [37] "Digital Art"        "DIY"                "DIY Electronics"
 [40] "Documentary"        "Drama"              "Drinks"
 [43] "Electronic Music"   "Embroidery"         "Events"
 [46] "Experimental"       "Fabrication Tools"  "Faith"
 [49] "Family"             "Fantasy"            "Farmer's Markets"
 [52] "Farms"              "Fashion"            "Festivals"
 [55] "Fiction"            "Film & Video"       "Fine Art"
 [58] "Flight"             "Food"               "Food Trucks"
 [61] "Footwear"           "Gadgets"            "Games"
 [64] "Gaming Hardware"    "Glass"              "Graphic Design"
 [67] "Graphic Novels"     "Hardware"           "Hip-Hop"
 [70] "Horror"             "Illustration"       "Immersive"
 [73] "Indie Rock"         "Installations"      "Interactive Design"
 [76] "Jazz"               "Jewelry"            "Journalism"
 [79] "Kids"               "Knitting"           "Latin"
 [82] "Letterpress"        "Literary Journals"  "Literary Spaces"
 [85] "Live Games"         "Makerspaces"        "Metal"
 [88] "Mixed Media"        "Mobile Games"       "Movie Theaters"
 [91] "Music"              "Music Videos"       "Musical"
 [94] "Narrative Film"     "Nature"             "Nonfiction"
 [97] "Painting"           "People"             "Performance Art"
[100] "Performances"       "Periodicals"        "Pet Fashion"
[103] "Photo"              "Photobooks"         "Photography"
[106] "Places"             "Playing Cards"      "Plays"
[109] "Poetry"             "Pop"                "Pottery"
[112] "Print"              "Printing"           "Product Design"
[115] "Public Art"         "Publishing"         "Punk"
[118] "Puzzles"            "Quilts"             "R&B"
[121] "Radio & Podcasts"   "Ready-to-wear"      "Residencies"
[124] "Restaurants"        "Robots"             "Rock"
[127] "Romance"            "Science Fiction"    "Sculpture"
[130] "Shorts"             "Small Batch"        "Software"
[133] "Sound"              "Space Exploration"  "Spaces"
[136] "Stationery"         "Tabletop Games"     "Taxidermy"
[139] "Technology"         "Television"         "Textiles"
[142] "Theater"            "Thrillers"          "Translations"
[145] "Typography"         "Vegan"              "Video"
[148] "Video Art"          "Video Games"        "Wearables"
[151] "Weaving"            "Web"                "Webcomics"
[154] "Webseries"          "Woodworking"        "Workshops"
[157] "World Music"        "Young Adult"        "Zines"
```

```r
levels(ds$main_category)
```

```
 [1] "Art"          "Comics"       "Crafts"       "Dance"        "Design"
 [6] "Fashion"      "Film & Video" "Food"         "Games"        "Journalism"
[11] "Music"        "Photography"  "Publishing"   "Technology"   "Theater"
```

```r
levels(ds$currency)
```

```
 [1] "AUD" "CAD" "CHF" "DKK" "EUR" "GBP" "HKD" "JPY" "MXN" "NOK" "NZD" "SEK"
[13] "SGD" "USD"
```

```r
levels(ds$state)
```

```
[1] "canceled"   "failed"     "live"       "successful" "suspended"
[6] "undefined"
```

```r
levels(ds$launched_month)
```

```
 [1] "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11" "12"
```

```r
levels(ds$launched_day_of_week)
```

```
[1] "1" "2" "3" "4" "5" "6" "7"
```

```r
levels(ds$launched_year)
```

```
 [1] "1970" "2009" "2010" "2011" "2012" "2013" "2014" "2015" "2016" "2017"
[11] "2018"
```

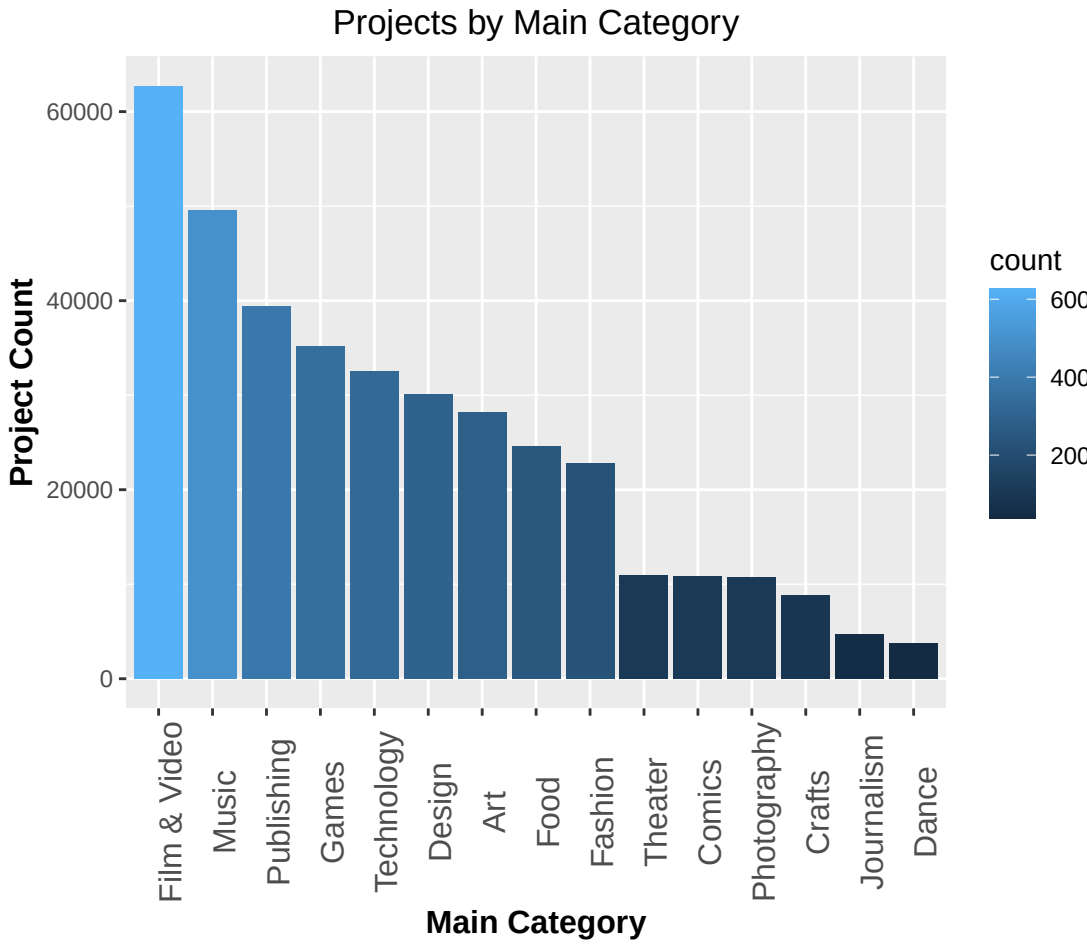
```r
#Country "N,0\"" seems strange - we'll keep an eye out for that later.
#Let's remove those
```

It now seems that the country field has some strange values (“N,0””) in it; we will remove these records.

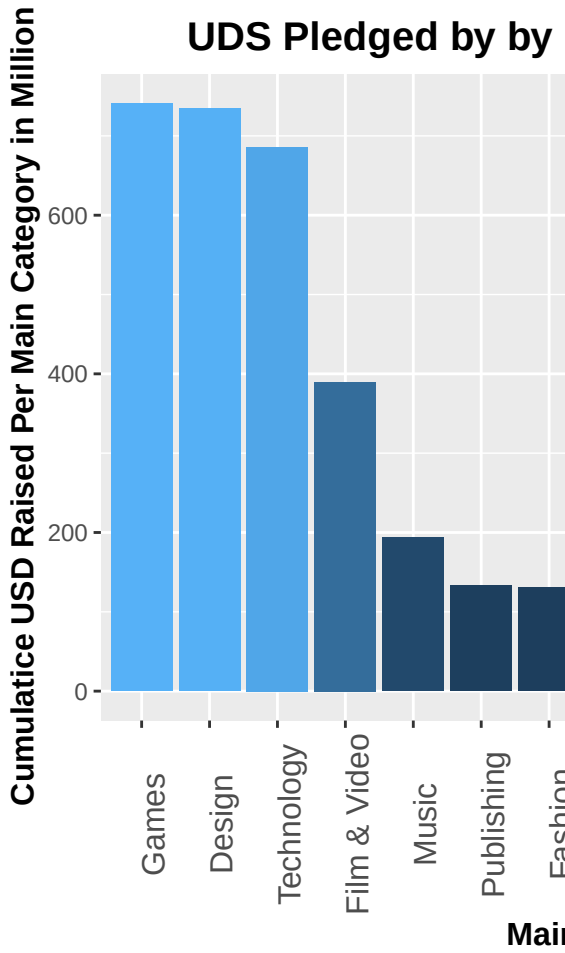
```
Rows: 374,857
Columns: 20
$ ID                  <dbl> 1000002330, 1000003930, 1000004038, 1000007540...
$ name                <chr> "The Songs of Adelaide & Abullah", "Greeting F...
$ category            <fct> Poetry, Narrative Film, Narrative Film, Music,...
$ main_category       <fct> Publishing, Film & Video, Film & Video, Music,...
$ currency            <fct> GBP, USD, USD, USD, USD, USD, USD, USD, USD, U...
$ deadline            <date> 2015-10-09, 2017-11-01, 2013-02-26, 2012-04-1...
$ goal                <dbl> 1000, 30000, 45000, 5000, 19500, 50000, 1000, ...
$ launched            <date> 2015-08-11, 2017-09-02, 2013-01-12, 2012-03-1...
$ pledged             <dbl> 0.00, 2421.00, 220.00, 1.00, 1283.00, 52375.00...
$ state               <fct> failed, failed, failed, failed, canceled, succ...
$ backers             <dbl> 0, 15, 3, 1, 14, 224, 16, 40, 58, 43, 0, 100, ...
$ country             <fct> GB, US, US, US, US, US, US, US, US, US, CA, US...
$ usd_pledged_real    <dbl> 0.00, 2421.00, 220.00, 1.00, 1283.00, 52375.00...
$ usd_goal_real       <dbl> 1533.95, 30000.00, 45000.00, 5000.00, 19500.00...
$ time_int            <dbl> 59, 60, 45, 30, 56, 35, 20, 45, 35, 30, 30, 30...
$ pledged_ratio       <dbl> 0.00, 0.08, 0.00, 0.00, 0.07, 1.05, 1.21, 0.02...
$ avg_backer_pldg     <dbl> 0, 161, 73, 1, 92, 234, 75, 11, 142, 145, 0, 1...
$ launched_month      <fct> 08, 09, 01, 03, 07, 02, 12, 02, 04, 07, 09, 03...
$ launched_day_of_week <fct> 2, 6, 6, 6, 6, 5, 1, 1, 4, 5, 1, 6, 2, 2, 3, 4...
$ launched_year       <fct> 2015, 2017, 2013, 2012, 2015, 2016, 2014, 2016...
```

Categories & Funds Pledged

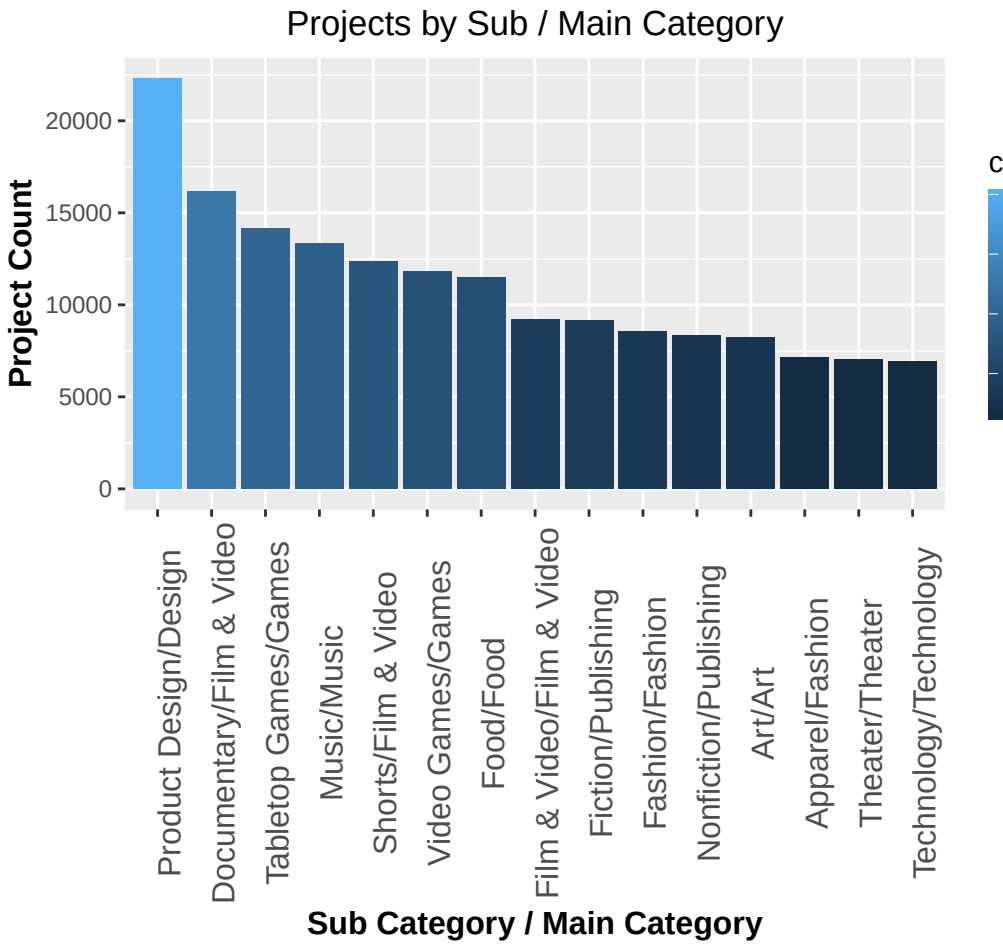
Each projecet has a **main_category** and a **category** section. Which **main_category** do users attempt to fund most often?



In terms of the main categoriex, **Film & video** projects seem to be the **most** wide-spreead, while

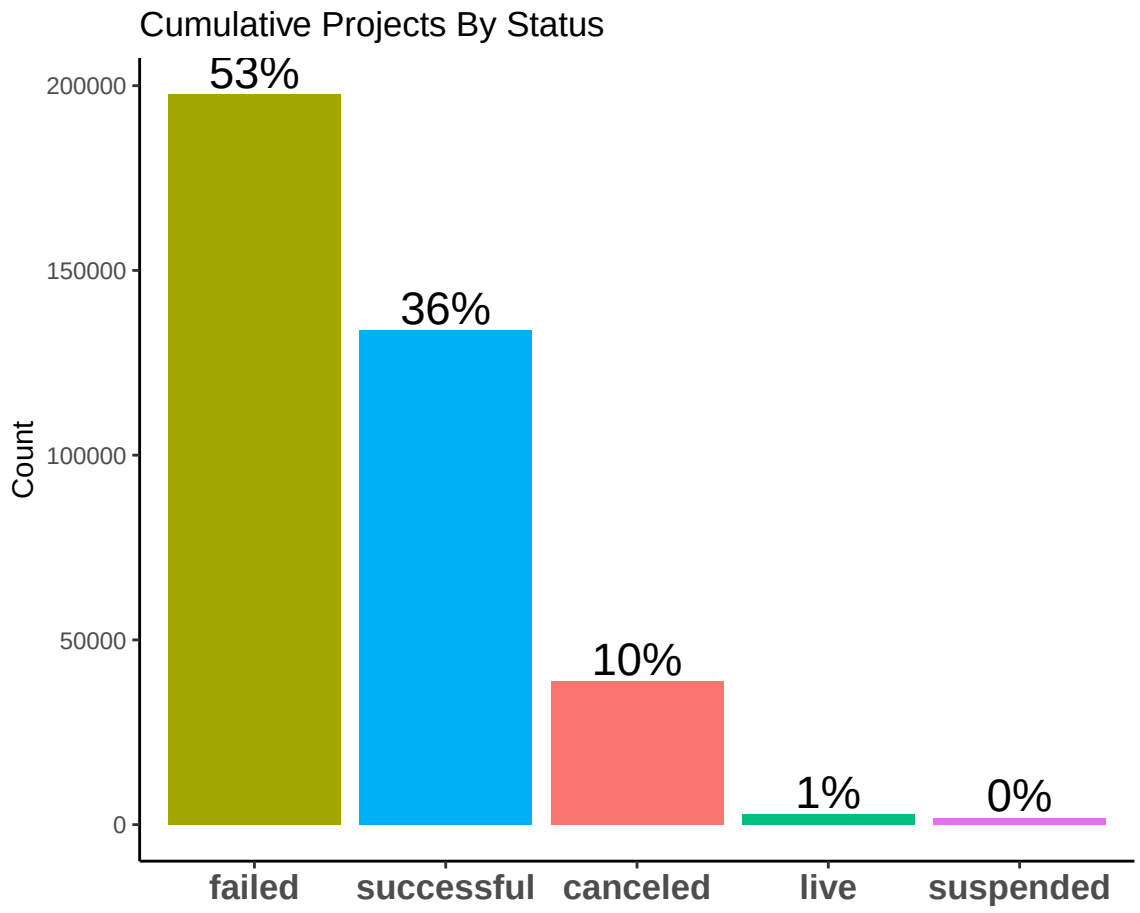


Dance is the **least**. What does this look like in terms of funds raised?



And what about sub-categories?

Project States

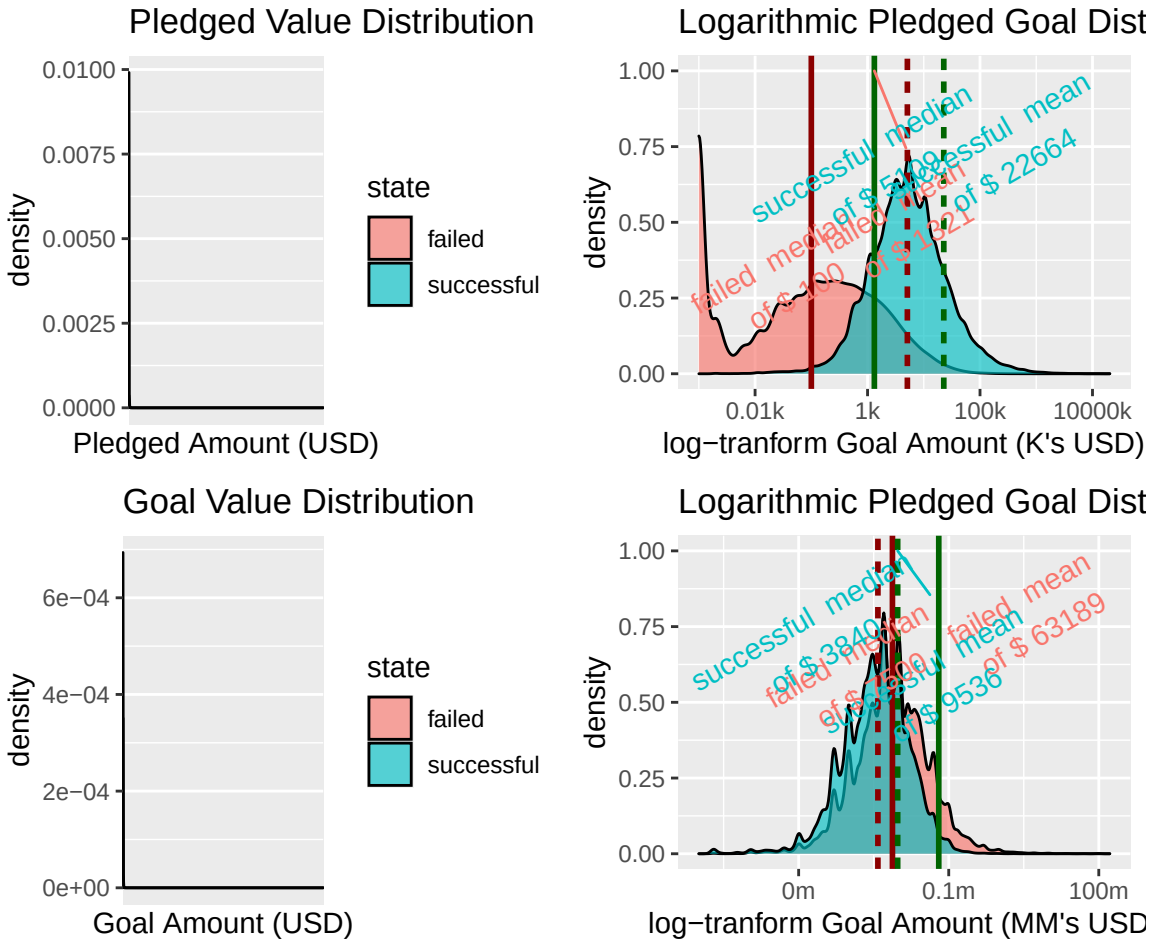


From the data and the bar graph of the project states, we can make these conclusions: " 1. The vast majority of the projects fall into the **"failed"** or **"successful"** category. 2. There are 6 discrete optionss. 3. Most projects **FAIL**.

The information in the dataset includes some facts, but is hardly a full picture of what happens in a project. We don't know about the marketing efforts, time/funds spent on those, or anything else. Therefor, it's hardly a detailed model we're building, but it's a start. As mentioned earlier, we will be focusing exclusively on the projecets that fall into the **"failed"** or **"successful"** categories.

Pledged vs Goal

Let's examine the distribution of pledged value and goal value in successful vs failed projects; As is evidenced by the densty plots on the left side, little info can be gleaned here. However, once the visualization is **log-transformed**, the distributions of the successful projects seem to be nearing normal. The pledged values are normally distributed for succesful projects, but not so much for the failed ones; this makes intuituve sense since many failed projects had raised no money



at all.

Digging into this further, a box plot of pledge ratios paints a very clear picture of failed vs. successful projects - and this makes sense intuitively, as well. We've already seen that nost successful projets are funded at over 100$ of goal.

```
round(quantile(ds$pledged_ratio[ds$state == 'failed'] , probs = seq(0.1 , 0.9 , 0.1), na
```

```
 10%  20%  30%  40%  50%  60%  70%  80%  90%
0.00 0.00 0.00 0.01 0.02 0.04 0.08 0.16 0.30
```

```
#Kickstarter has a requirement of 50% funding for a project. Here, we can see that the

# Quantiles of succesesful projects pledged_ratio - all over 100%
```

And if we examine the pledge ratios quantiles, this is also obvious. Faled:

```
round(quantile(ds$pledged_ratio[ds$state == 'successful'] , probs = seq(0.1 , 0.9 , 0.1)
```

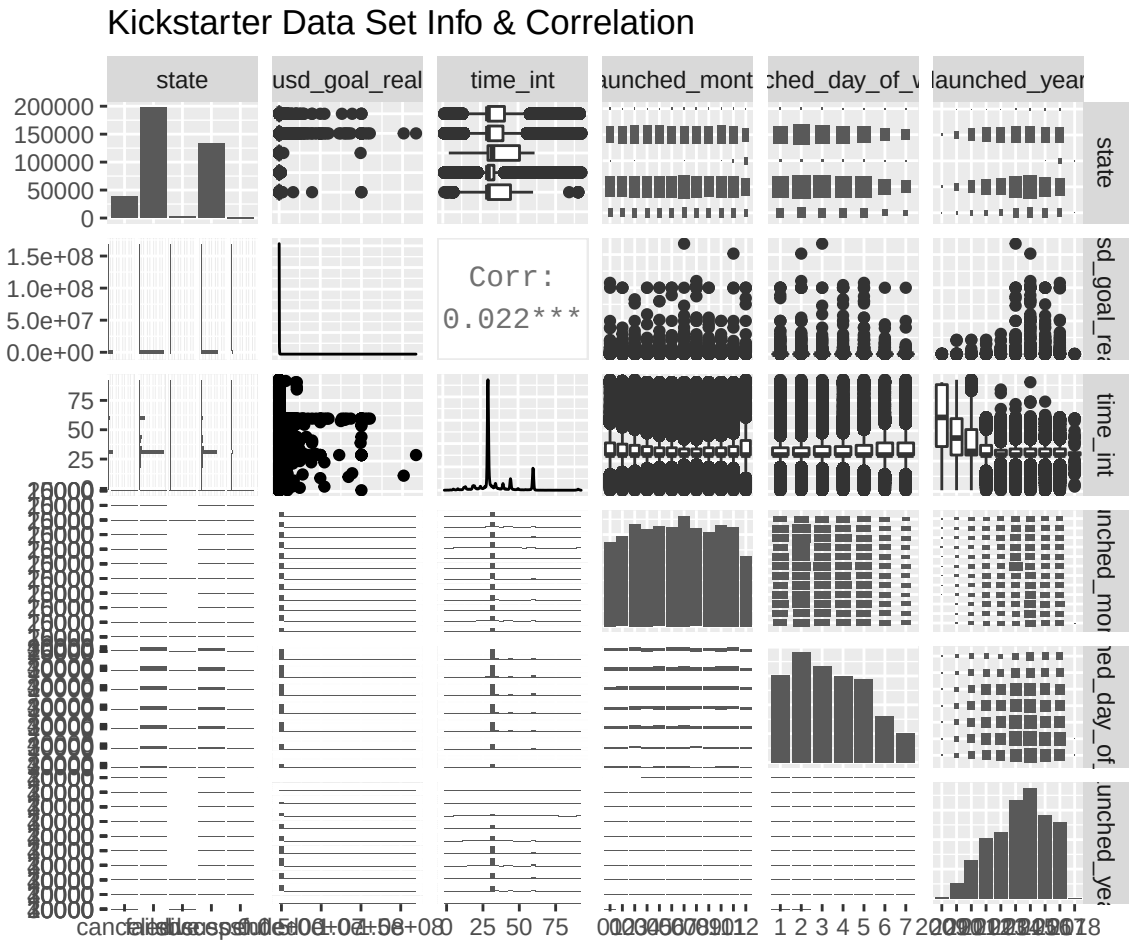
```
 10%  20%  30%  40%  50%  60%  70%  80%  90%
1.01 1.03 1.06 1.10 1.17 1.27 1.46 1.91 3.39
```

```
#Kickstarter has a requirement of 50% funding for a project. Here, we can see that the


#correlation of some of the features and overview of their relationship is shown here.
```

Feature Correlation & Pairs

Correlation of some of the features and overview of their relationship is shown here. The only items correlated very slightly are the time interval and the goal amount in USD. This makes sense intuitively as the planners expect to raise some amount of money in a specific time.



It looks like the only real correlation - .0022 is shown between pledged amounts and the amount of backers. This makes sense of course, since the amount of backers will be a small indicator of the amount of funds raised.

Functions of Note

This project used several different functions to help with the creation of dynamic model formulas, as well as some wrappers to quickly and neatly get accuracy and confusion matrix data. I'm including these here.

```
results_GLM <- function(train_data, predicted_field_name, predictors_list, model_family=
  fm_string <- paste(predicted_field_name, "~")
  for (i in 1:length(predictors_list)) {
    fm_string <- paste(fm_string, predictors_list[i])
    if (i < length(predictors_list))
      fm_string <- paste(fm_string, "+")
  }
  print(fm_string)
  fm <- as.formula(fm_string)

  glm_fit <- glm(fm , train_data, family = model_family)
  return (glm_fit)
}


get_confusion_matrix <- function(model_fit, test_data)
{
  aug_model <- augment(model_fit, newdata = test_data, type.predict = 'response')

  result <- aug_model %>% mutate(Prediction = factor(round(.fitted)), Reference = state)
    dplyr::select(Reference , Prediction ) %>% table()

  return (caret::confusionMatrix(result))
}

get_accuracy <- function(cf_matrix)
{
  cf_accuracy <- sum(diag(cf_matrix)) / sum(cf_matrix)
  return(cf_accuracy)
}

#since the outcome is a factor and not just numeric, we can't use simple linear regres
#predictions based solely on category
```

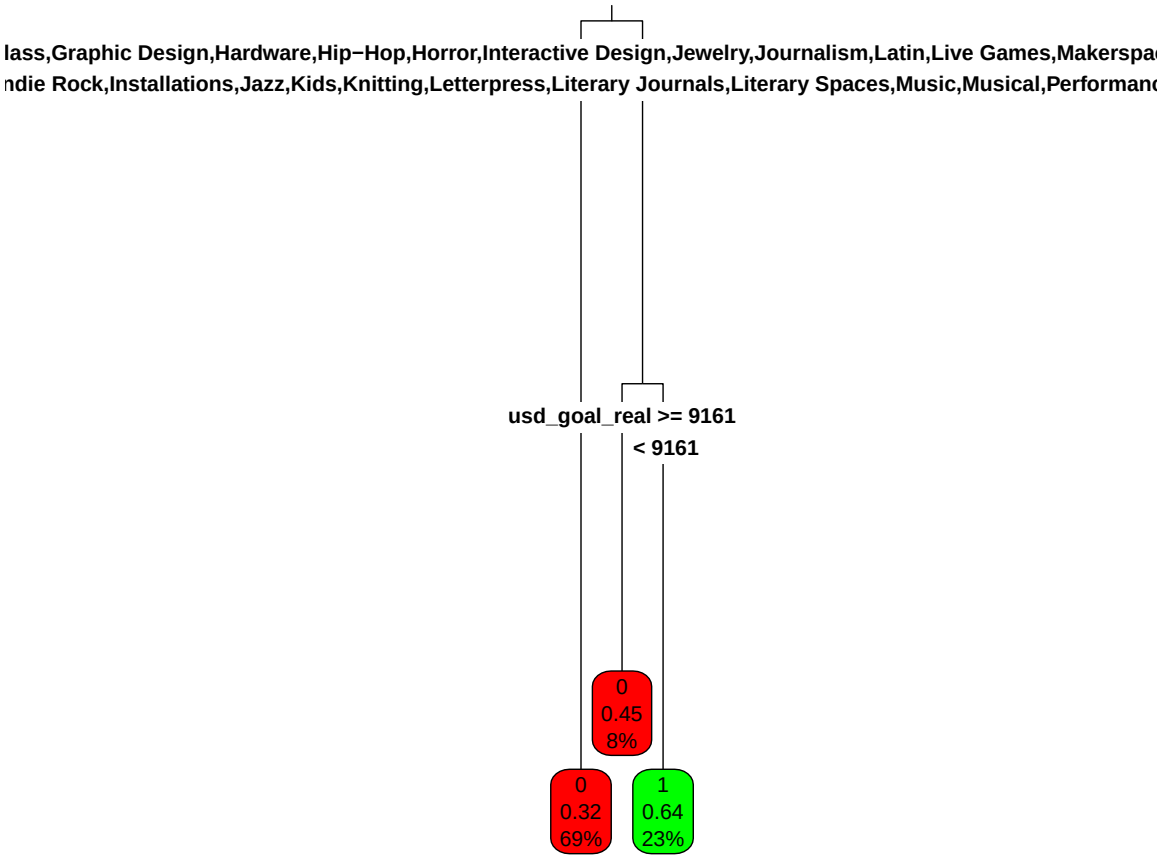
Rating Models

Since we are trying to predict the variable **state** and it is **categorical**, as well as are many of the predictors, **linear regression was not a good option for for this project**. Therefor, we attempt the

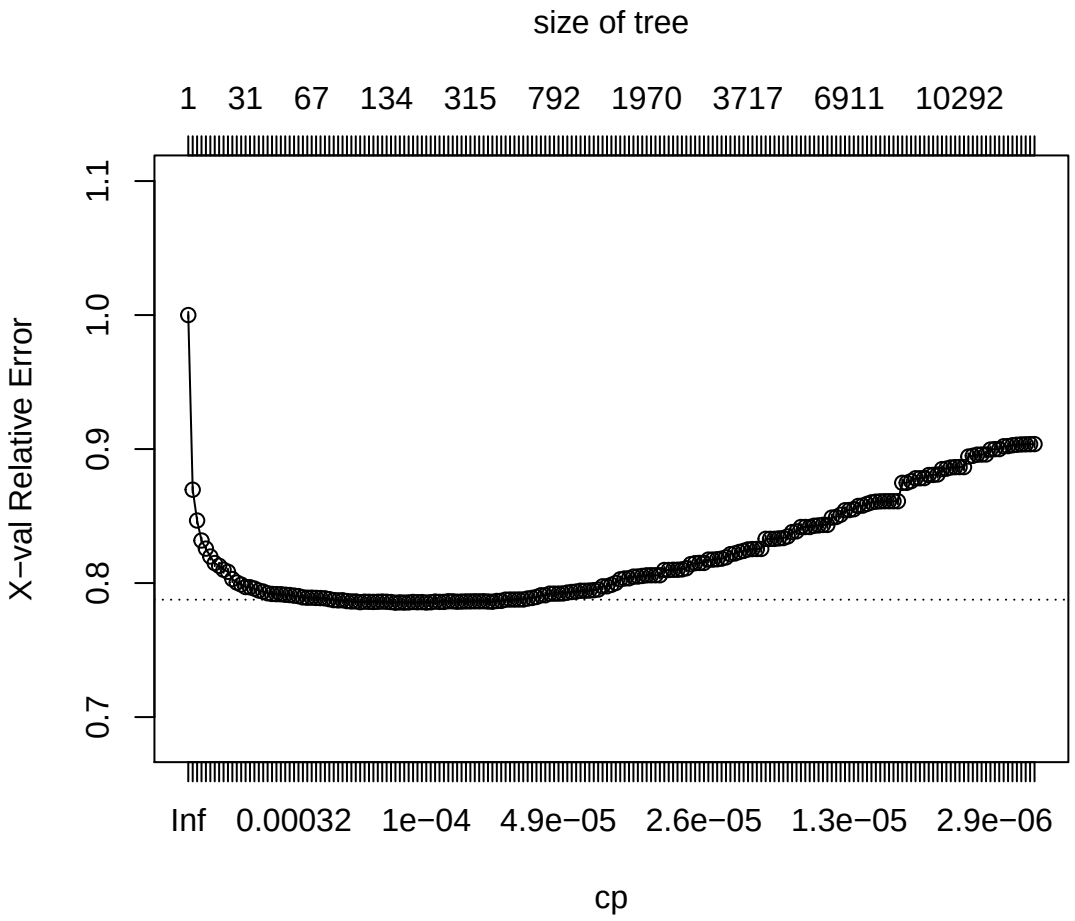
models below.

1. Classification Trees

The first model I will attempt will be **classification trees**. The initial attempt used the following parameters: 1. maxdepth = 5 (the maximum depth of any node of the final tree) 2. minsplit = 200 (the minimum number of observations that must be observed before splitting the tree)



The next attempt once again runs the model, but this time with all default parameters, save **cp**, the complexity parameter, which we set to 0. The logic behind this is that we make a detailed tree and prune it AFTER the model has run its course. **The plot of the cp values** is shown below, showing the optimal (lowest) value for the relative error.



The code then selects this minimal value, 2.2257502×10^{-4}, and prunes the tree with that complexity paramter. The accuracy of the resulting model is **0.6844813**. As I continue with the project, I will display results of the accuracy evaluations of every model, and print them out as shown below - so that the data can be reviewed as the project progresses.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813

Next, still attempting to optimize classification trees, I will attempt to run the model with ALL default parameters set, and then again post-prune the resulting tree by finding the minimum **cp** parameter.

This time, the code selects a minimal cp value of **0.01** to prune the tree. The accuracy of the resulting model is **0.6589737**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737

2. Random Forest

When first attempting to run the **random forest** algorithm on the data, it became obvious that the general implementation wouldn't be plausable here due to some of the factors (the categories in particular) having more then 32 levels. (This is a limitation of the random forest algorithm implementation in R.) Therefor, the data had to be converted into a matrix with dummy variables. A sample of 70,000 was taken to evaluate the training set; the processing time for this was very long - and in the future implementation of the model I will attempt to run random forest on a larger subset of data. The final random forest model accuracy with **200 trees and a 70,000 samples** was **0.683546**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460

3. Naive Bayes

The next attemped model is the Naive Bayes Model. In the initial attempt, I used the same training/testing dataframes as in the classification trees attempts described above. This produced an accuracy of **0.4431321**, which is so far the lowest we've seen (and will prove to be the worst).

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321

However, the algorithm also takes matrices as input, and at the next iteration the model will attempt to evaluate the accuracy on the matrix data (which is simply the dataset with dummay

variables.) Here, the accuracy was much higher, at **0.5416931**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931

4. Generalized Linear Models (GLM)

The next, and most extensive set of models uses **Generalized Linear Regression** - a variation of linear regression that allows for predictions of categorical variables. Since here we are trying to predict the variable **state**, which is categorical, GLM are a good option to experiment with. I attempt to run the models on several of the build-in and engineered predictors, as shown above. The best-performing model in the GLM category is the one that uses the most factors.

4.1 1 Predictor - “category”

First, we use the sole variable **“category”** as a predictor.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829

4.2 2 Predictors - “category” and Time In Days

Next, we use the variables **“category”** and **time_int** (the time in days from launch to goal date) as a predictor. This gives us slightly better results, and is marked as **“GLM 2”**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737

model	accuracy
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954

4.3 3 Predictors - "category" and Time In Dayls, and Country

Next, we use the variables **"category"** and **time_int**, and **country**. This gives us slightly better results, and is marked as **"GLM 3"**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172

4.4 4 Predictors - "category" and Time In Dayls, Country and the Goal Amount in $USD

Next, we use the variables **"category"** and **time_int**, **country**, **usd_goal_real**. This gives us slightly better results, and is marked as **"GLM 4"**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172
GLM 4 predictors	0.6678432

4.5 5 Predictors - “category” and Time In Dayls, Country, the Goal Amount in $USD, and Launched Month

Next, we use the variables **“category”** and **time_int**, **country**, **usd_goal_real**, and **“launched_month”**. This gives us slightly better results, and is marked as **“GLM 5”**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172
GLM 4 predictors	0.6678432
GLM 5 predictors	0.6680695

4.6 6 Predictors - “category” and Time In Dayls, Country, the Goal Amount in $USD, Launched Month, and Launched Day of the Week

Next, we use the variables **“category”** and **time_int**, **country**, **usd_goal_real**, and **“launched_month”**. This gives us slightly better results, and is marked as **“GLM 6”**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172
GLM 4 predictors	0.6678432
GLM 5 predictors	0.6680695
GLM 6 predictors	0.6691556

4.7 7 Predictors - “category” and Time In Dayls, Country, the Goal Amount in $USD, Launched Month, Launched Day of the Week, and Currency

Next, we use the variables **“category”** and **time_int**, **country**, **usd_goal_real**, **“launched_month”**, and **“currency”**. This gives us slightly better results, and is marked as **“GLM 7”**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172
GLM 4 predictors	0.6678432
GLM 5 predictors	0.6680695
GLM 6 predictors	0.6691556
GLM 7 predictors	0.6691556

4.8 8 Predictors - “category”, Time In Dayls, Country, the Goal Amount in $USD, Launched Month, Launched Day of the Week, Currency, and Launched Year

Next, we use the variables **“category”** and **time_int**, **country**, **usd_goal_real**, **“launched_month”**, **“currency”**, and **“launched_year”**. This gives us slightly better results, and is marked as **“GLM 8”**.

model	accuracy
Classification Trees mex depth=5,minsplit=200	0.6589737
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
Random Forest, ntree=200	0.6835460
Naive Bayes	0.4431321
Naive Bayes w/matrix dummy variables	0.5416931
GLM 1 predictor	0.6507829
GLM 2 predictors	0.6574954
GLM 3 predictors	0.6587172
GLM 4 predictors	0.6678432
GLM 5 predictors	0.6680695
GLM 6 predictors	0.6691556
GLM 7 predictors	0.6691556
GLM 8 predictors	0.6728211

Results

After creating a number of predictive algorithms, the best model as per the project requirements is the one with the highest accuracy so far - this happens to be the **cp=0 Classification Trees post-pruned with min cp=0.000222575021790562** model, with the accuracy of **0.6844813**.

model	accuracy
cp=0 Classification Trees post-pruned with min cp=0.000222575021790562	0.6844813
Random Forest, ntree=200	0.6835460
GLM 8 predictors	0.6728211
GLM 6 predictors	0.6691556
GLM 7 predictors	0.6691556
GLM 5 predictors	0.6680695
GLM 4 predictors	0.6678432
Classification Trees mex depth=5,minsplit=200	0.6589737
Classification Trees, default params, un-pruned	0.6589737
Classification Trees, default params, post-pruned cp=0.01	0.6589737
GLM 3 predictors	0.6587172
GLM 2 predictors	0.6574954
GLM 1 predictor	0.6507829
Naive Bayes w/matrix dummy variables	0.5416931
Naive Bayes	0.4431321

Conclusion

Report Summary

Other Options

Using Various Regression Functions.

Possible Improvements

I think improvements on the RMSE could be achieved by evaluating the project with other means. Different machine learning models could also improve the results further, but hardware limitations (memory and processing power) may be a constraint.

The possible evaluation models we can also attempt to get even better results would include:

- Matrix factorization / Single Value Decomposition (SVD) / Principal Component Analysis (PCA)
- Gradient Descent
- SGD Code Walk

Appendix - Environment

```
print("Operating System:")
```

```
[1] "Operating System:"
```

```
version
```

```
               _
platform       x86_64-apple-darwin17.0
arch           x86_64
os             darwin17.0
system         x86_64, darwin17.0
status
major          4
minor          0.2
year           2020
month          06
day            22
svn rev        78730
language       R
version.string R version 4.0.2 (2020-06-22)
nickname       Taking Off Again
```