KickStarter Project

(Eliza) Zijin Huang, Tian Xia

4/17/2019

Part I. Report

Summary and Problem Statement

A description of the problem that is driving the project (comprehensible by an educated lay person).

In this project, we are predicting the success rate of Kickstarter campaigns. Kickstarter is the world's largest funding platform for creative projects, with projects ranging from technology to arts. Kickstarter's rule states that if a project falls short of meeting its minimum funding goal by its deadline, the project will not receive any fund. Thus very important to set a realistic goal. However, there is currently no practical suggestion or guidance, and the analytics dashboard is hard to set up. Therefore, we are interested in predicting the projects' success rate given a project's goal and other independent variables. Predicting the success rate increases the number of funding goals reached. Ultimately, it supports the entrepreneurs, and in the process support art, advocate for social causes, and bring innovation ideas and products to the world.

Data

A rationale for selecting the data collected used to investigate the problem.

The dataset we used is pre-scraped data stored in multiple csv files. It includes 7,574 projects from North America, Europe, Pacific countries, and Japan. Each project in our dataset has either failed, successful, live, suspended or canceled status. Since we are only interested in projects with status of either successful or failed, we filtered out projects whose status is not successful or failed. In addition we also dropped projects that have missing values.

Our selected dataset is complete and good for us to analyze the data and construct data predictive models. We payed attention to some specific parameters that are important for our analysis such as projects' goal, amount pledged, category, duration, region, etc. All those features are included, even though the data might be untidy or not in the best format for analysis. However, some unrelated parameters such as projects' description, urls, photo have nothing to do with success rate. Thus, we drop those columns and leave parameters that are closely related with project's success.

Cleaning, Visualization, and Prediction

Justification for the processes implemented, and the technical choices you made for your project.

For data cleaning, we chose to use tidyverse, lubridate packages with regular expression and some baseR functions. In order to extract useful information and get rid of irrelevant data, we used regular expression to extract category, location and tidy other parameters into standard format. Also we used lubridate package to create duration columns based on given information about launched date and end date.

In the process of data exploration and visualization analysis, we chose to use ggthemes and rworldmap packages for visualizing the data and dplyr packages to select relevant data. Ggthemes can help us make data more visualized in terms of colors of bars, lines, and points. For example, the higher proportion the category takes or the higher the success rate is, the darker the color of that category is. We also used rworldmap package because we want to present and acknowledge where our data is coming from. It is relatively wide-ranged, even though it does not cover Africa and South America.

For prediction, we used the following models: k-nearest neighbour, logistic regression, support vector machine, and naive bayes. K-nn model is commonly used in classification and regression predictive problems, but has a long calculation time. Simple logistic regression model is used because there are multiple independent variables. SVM is used for classification because if can find an optimal boundary between the classes. NB model takes less time to train, and requires less data. Thus, we choose to implement the above four models for our prediction/classification task.

Issues and Solution

A description of any issues you ran into and how you resolved them.

When we clean the dataset, one challenge we had is that some of columns' data are stored in JSON format and they actually compress a lot of relevant data. The approach we took to solve this problem is by using string extract and replace with regular expression to extract useful information and split them into several columns. In addition, the original data does not have the duration in format of number of days, instead we had launched date and end date, both in Unix time. We used r packages to convert them into readable year/month/day format, and computed two duration variables with the three given timestamp.

In the prediction process, we encounter the following problems:

First, the data frame contains a mix of independent variables and dependent variables. To solve this problem, we implemented a correlation plot of all the variables, and dropped those that are in fact dependent variables. For instance, we dropped "spotlight" (meaning if a Kickstarter project is featured on the homepage of the website), because it is highly correlated with the final state (successful or unsuccessful), and is in fact a dependent variable.

Second, some models (SVM) takes has a long calculation time in R. As a result, we choose to implement the models in python using jupyter notebook for shorter time and more suitable visualization tools (such as Matlibplot).

Please refer to Part II and Part III of this report for input code and session outputs.

Insigths and Potential Future Work

Insights on what you learned and potential future work.

Throughout this project, We have learned some things that we do not learn in lectures or homeworks. First, it is our first time to play around with such huge, untidy, and real data. We had some challenges for cleaning data but solved them eventually (refer to the issue section for more details). We realized that a fairly confident prediction can be generated based on our chosen variables. Thus, it is possible to predict the success, and potentially make modification to the project setup, in order to improve the success rate of Kickstarter crowdfunding projects.

In future, we are planning to do a multi-class/multinomial classification or prediction, instead of the binary classification process we are using. This would give more precise feedbacks to the project initiators.

We are also considering building interactive visualization tools (possibly chrome extension) to analyze a specific project's parameters, and give real-time feedback to the project initiators. The feedback could focus on variables such as project duration, pledged amount, areas of improvement on project description etc. Some examples of the suggestions are: "Use a longer time period of 100 days." "Reduce the pledged amount to 1000 dollars." "Replace the word 'must' with 'could'." "Set more pledging options with smaller amounts, such as $5."

Part II. Data Cleaning and Visualization

#1. **Data Acquisition and Integration

```
data_source_1 <- read.csv("./data/Kickstarter001.csv", header = TRUE, sep = ",")
data_source_2 <- read.csv("./data/Kickstarter002.csv", header = TRUE, sep = ",")

raw_data <- rbind(data_source_1, data_source_2)
```

#2. **Data Cleaning There are 3784 data totally and there are 3680 projects are completed.

```
live_data <- raw_data %>% filter(raw_data$state == "live")
```

To clean the "catagory" column

```
raw_data$category <- raw_data$category %>%
  str_extract("slug\":\".+\",\"") %>%
  str_replace_all("\",\"", "") %>%
  str_replace_all("slug\":\"", "") %>%
  str_replace("/.+", "")
```

To clean the "location" column

```
raw_data$location <- raw_data$location %>%
  str_extract("name\":\".+\",\"") %>%
  str_replace("\",\".+", "") %>%
  str_replace_all("name\":\"", "")
```

To get rid of creator,photo, slug, urls column

```
raw_data$creator <- NULL
raw_data$photo <- NULL
raw_data$slug <- NULL
raw_data$urls <- NULL
```

To add a preparation_duration column

```
raw_data$preparation_duration <- raw_data$launched_at - raw_data$created_at
raw_data$preparation_duration_r <- seconds_to_period(raw_data$preparation_duration)
```

To add a launch_duration column

```
raw_data$launch_duration <- raw_data$deadline - raw_data$launched_at
raw_data$launch_duration_r <- seconds_to_period(raw_data$launch_duration)
raw_data$launch_duration_r <- day(raw_data$launch_duration_r)
```

To convert epoch seconds to readable time

```
raw_data$created_at_readable <- anytime(raw_data$created_at)
raw_data$deadline_readable <- anytime(raw_data$deadline)
raw_data$launched_at_readable <- anytime(raw_data$launched_at)
raw_data$preparation_duration_r <- NULL
```

Transfer raw data into a new variable

```
clean_data <- raw_data
write.csv(clean_data, "./data/data.csv")
head(clean_data)
```

```
##   backers_count
```

```
## 1           30
## 2            0
## 3          102
## 4           22
## 5            2
## 6            7
##
## 1 Experience tea and coffee as it should be in our handmade, fine bone china mugs. Made exclusively
## 2    Playing Roles Outside of Basic Education (P.R.O.B.E)\nThe magazine that highlights extracurricul
## 3                                                                                       A pilot for
## 4 A film about suicide. The struggles of our modern world taking people to their limit and how commo
## 5              Fusing the technical qualities and accuracy of photography with a digital process to
## 6                                                 A digital, interactive magazine and online community fo
##       category converted_pledged_amount country created_at currency
## 1       crafts                     1547      GB 1515610761      GBP
## 2   publishing                        0      US 1426362805      USD
## 3 film & video                     8101      US 1525106061      USD
## 4 film & video                     1566      GB 1519854040      GBP
## 5          art                       11      US 1407346285      USD
## 6   publishing                      826      US 1411150798      USD
##   currency_symbol currency_trailing_code current_currency   deadline
## 1               £                  false              USD 1521190409
## 2               $                   true              USD 1429112946
## 3               $                   true              USD 1531713540
## 4               £                  false              USD 1522443600
## 5               $                   true              USD 1410484909
## 6               $                   true              USD 1414008752
##   disable_communication friends  fx_rate  goal         id is_backing
## 1                 false         1.308394  1000 1361161119
## 2                 false         1.000000  5000  746509287
## 3                 false         1.000000  6000 1402909261
## 4                 false         1.308394   400  311541751
## 5                 false         1.000000 11000  466957735
## 6                 false         1.000000  2000 1471254290
##   is_starrable is_starred launched_at       location
## 1        false             1517306009         London
## 2        false             1426520946       Columbus
## 3        false             1529070876 St. Petersburg
## 4        false             1519937886         Dorset
## 5        false             1407892909      Ypsilanti
## 6        false             1411416752  San Francisco
##                                         name permissions pledged
## 1 Fine Bone China Ceramic Mugs, Made in England            1111.0
## 2                          P.R.O.B.E. Magazine               0.0
## 3                      'Merican Wasteland Pilot            8101.0
## 4                          Cliff - Feature Film            1116.5
## 5                              Photo to Artwork              11.0
## 6   Lilah Magazine 1st issue launching Dec 2014             826.0
##
## 1
## 2
## 3
## 4 {"id":3322408,"project_id":3322408,"state":"active","state_changed_at":1523464237,"name":"CLIFF -
## 5
```

```
## 6
##                                                                  source_url
## 1                   https://www.kickstarter.com/discover/categories/crafts
## 2 https://www.kickstarter.com/discover/categories/publishing/periodicals
## 3         https://www.kickstarter.com/discover/categories/film%20&%20video
## 4         https://www.kickstarter.com/discover/categories/film%20&%20video
## 5        https://www.kickstarter.com/discover/categories/art/digital%20art
## 6 https://www.kickstarter.com/discover/categories/publishing/periodicals
##   spotlight staff_pick      state state_changed_at static_usd_rate
## 1      true      false successful       1521190409        1.413819
## 2     false      false     failed       1429112947        1.000000
## 3      true      false successful       1531713540        1.000000
## 4      true      false successful       1522443600        1.390235
## 5     false      false     failed       1410484909        1.000000
## 6     false      false     failed       1414008752        1.000000
##   usd_pledged      usd_type preparation_duration launch_duration
## 1    1570.753 international              1695248         3884400
## 2       0.000 international               158141         2592000
## 3    8101.000 international              3964815         2642664
## 4    1552.197 international                83846         2505714
## 5      11.000      domestic               546624         2592000
## 6     826.000 international               265954         2592000
##   launch_duration_r created_at_readable    deadline_readable
## 1                44 2018-01-10 13:59:21 2018-03-16 04:53:29
## 2                30 2015-03-14 15:53:25 2015-04-15 11:49:06
## 3                30 2018-04-30 12:34:21 2018-07-15 23:59:00
## 4                29 2018-02-28 16:40:40 2018-03-30 17:00:00
## 5                30 2014-08-06 13:31:25 2014-09-11 21:21:49
## 6                30 2014-09-19 14:19:58 2014-10-22 16:12:32
##   launched_at_readable
## 1  2018-01-30 04:53:29
## 2  2015-03-16 11:49:06
## 3  2018-06-15 09:54:36
## 4  2018-03-01 15:58:06
## 5  2014-08-12 21:21:49
## 6  2014-09-22 16:12:32
```

#3. **Data exploration and Visualization

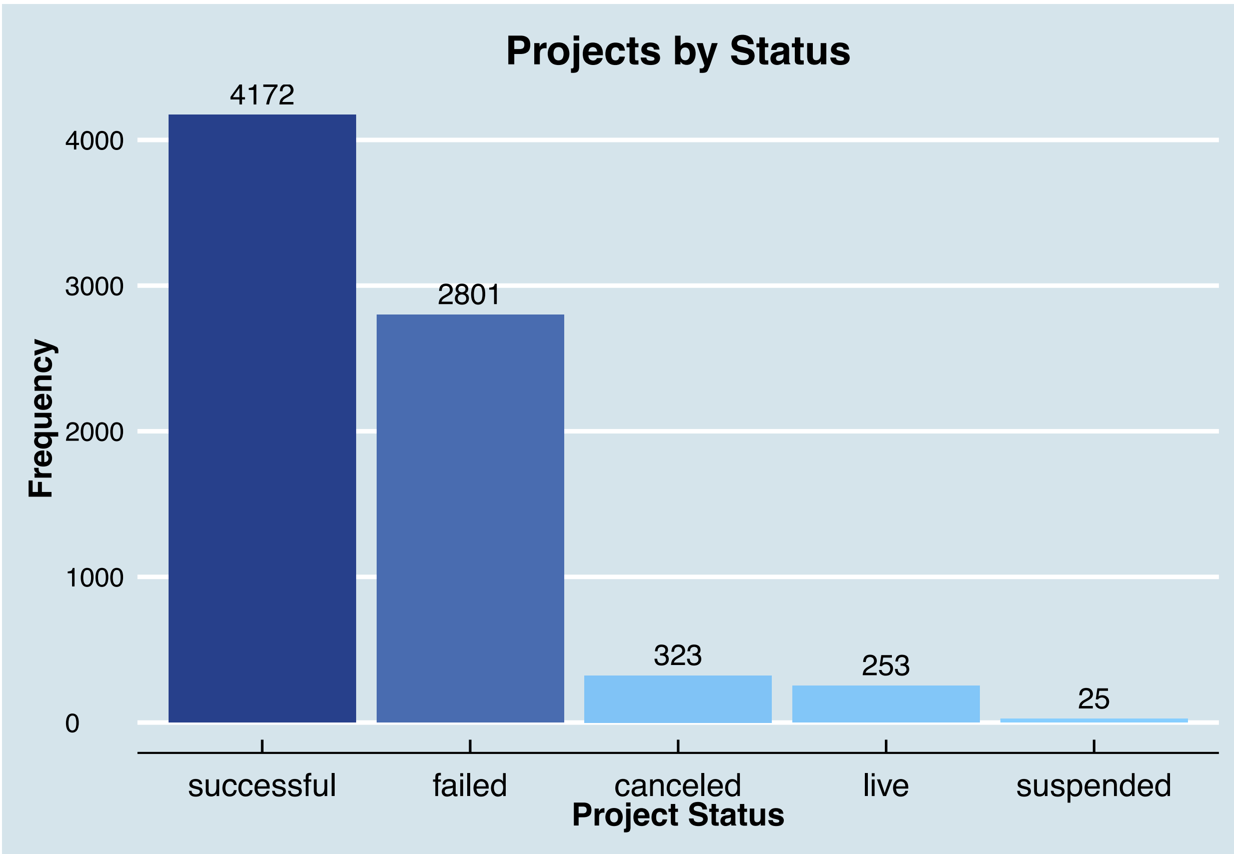
3.1 Summarise the number of projects for each status

```r
status_prjects <- clean_data %>%
  group_by(clean_data$state) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
head(status_prjects)
```

```
## # A tibble: 5 x 2
##   `clean_data$state` count
##   <fct>              <int>
## 1 successful          4172
## 2 failed              2801
## 3 canceled             323
## 4 live                 253
## 5 suspended             25
```

Plot the number of projects for each status

```
ggplot(status_prjects, aes(reorder(status_prjects$`clean_data$state`, -count), count, fill=count)) + ge
  ggtitle("Projects by Status") + xlab("Project Status") + ylab("Frequency") +
  geom_text(aes(label=count), vjust=-0.5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```



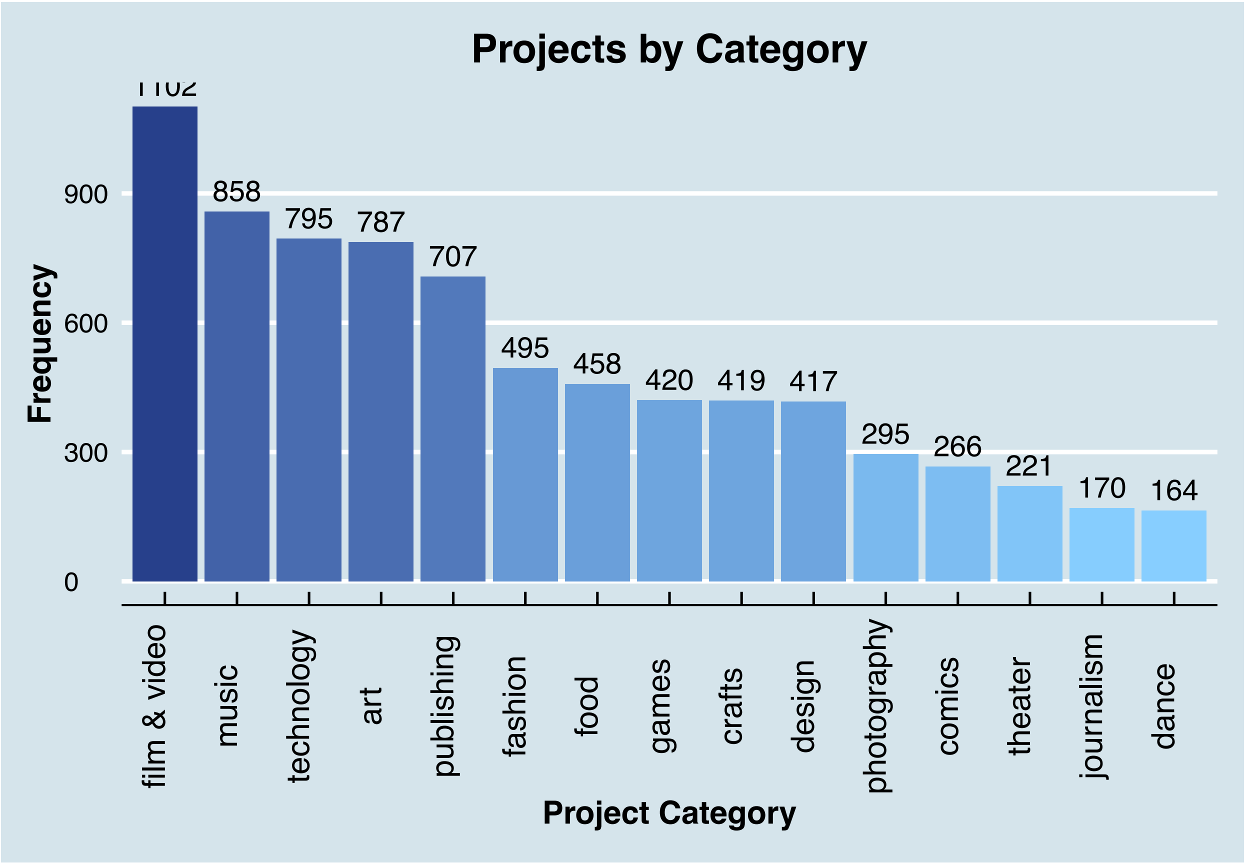
3.2 Summarise the number of projects for each catagory

```
catagory_projects <- clean_data %>%
  group_by(clean_data$category) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
head(catagory_projects)
```

```
## # A tibble: 6 x 2
##   `clean_data$category` count
##   <chr>                 <int>
## 1 film & video           1102
## 2 music                   858
## 3 technology              795
## 4 art                     787
## 5 publishing              707
## 6 fashion                 495
```

Plot the popularity of each category, which is dertermined by the number of projects

```
ggplot(catagory_projects, aes(reorder(catagory_projects$`clean_data$category`, -count), count, fill=cou
  ggtitle("Projects by Category") + xlab("Project Category") + ylab("Frequency") +
  geom_text(aes(label=count), vjust=-0.5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12, angle=90), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```



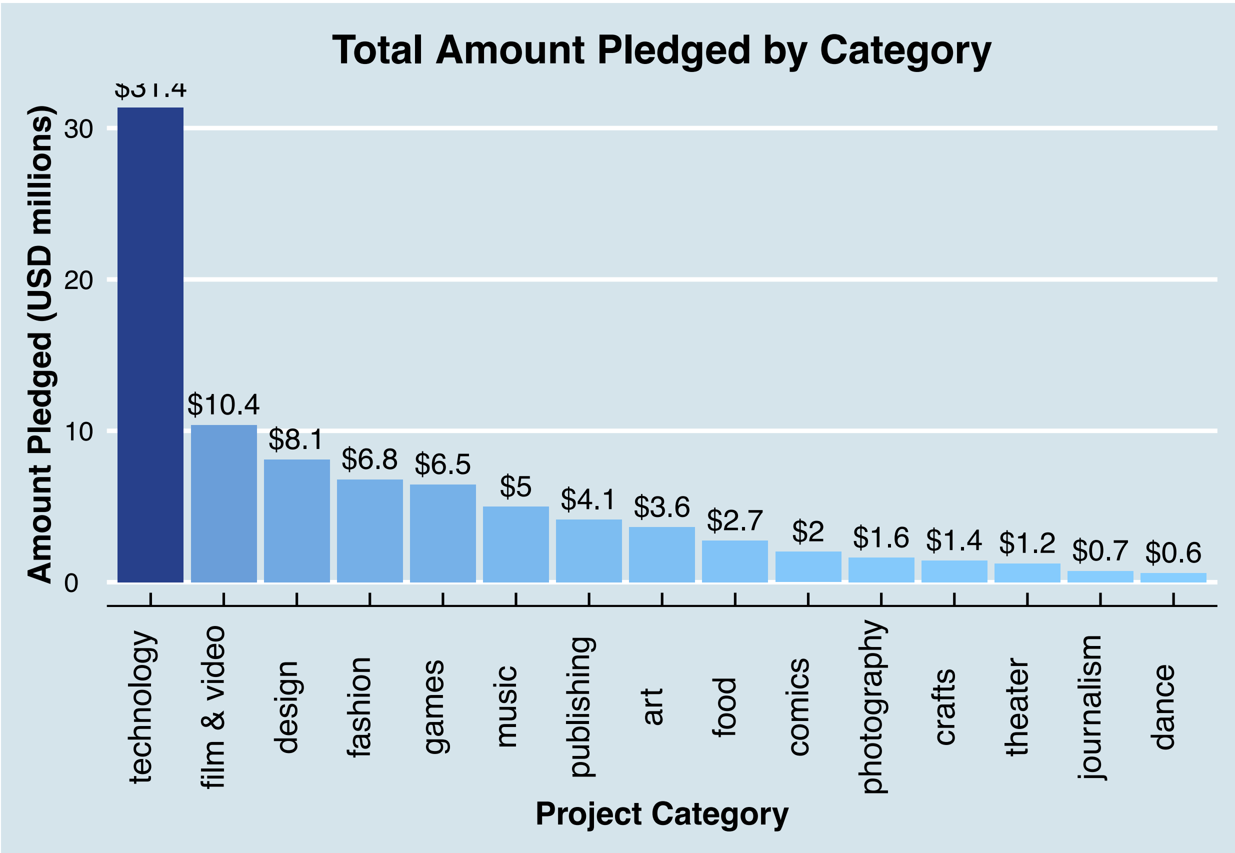
3.3 What types of projects are being funded?

```
pledged_category <- clean_data %>%
  group_by(clean_data$category) %>%
  summarise(total = sum(usd_pledged)) %>%
  arrange(desc(total))
head(pledged_category)
```

```
## # A tibble: 6 x 2
##   `clean_data$category`     total
##   <chr>                     <dbl>
## 1 technology            31373736.
## 2 film & video          10398557.
## 3 design                 8108525.
## 4 fashion                6797765.
## 5 games                  6465490.
## 6 music                  4988363.
```

Plot the amount pledged by each category

```r
ggplot(pledged_category, aes(reorder(pledged_category$`clean_data$category`, -total), total/1000000, fi
  ggtitle("Total Amount Pledged by Category") + xlab("Project Category") +
  ylab("Amount Pledged (USD millions)") +
  geom_text(aes(label=paste0("$", round(total/1000000,1))), vjust=-0.5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12, angle=90), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```



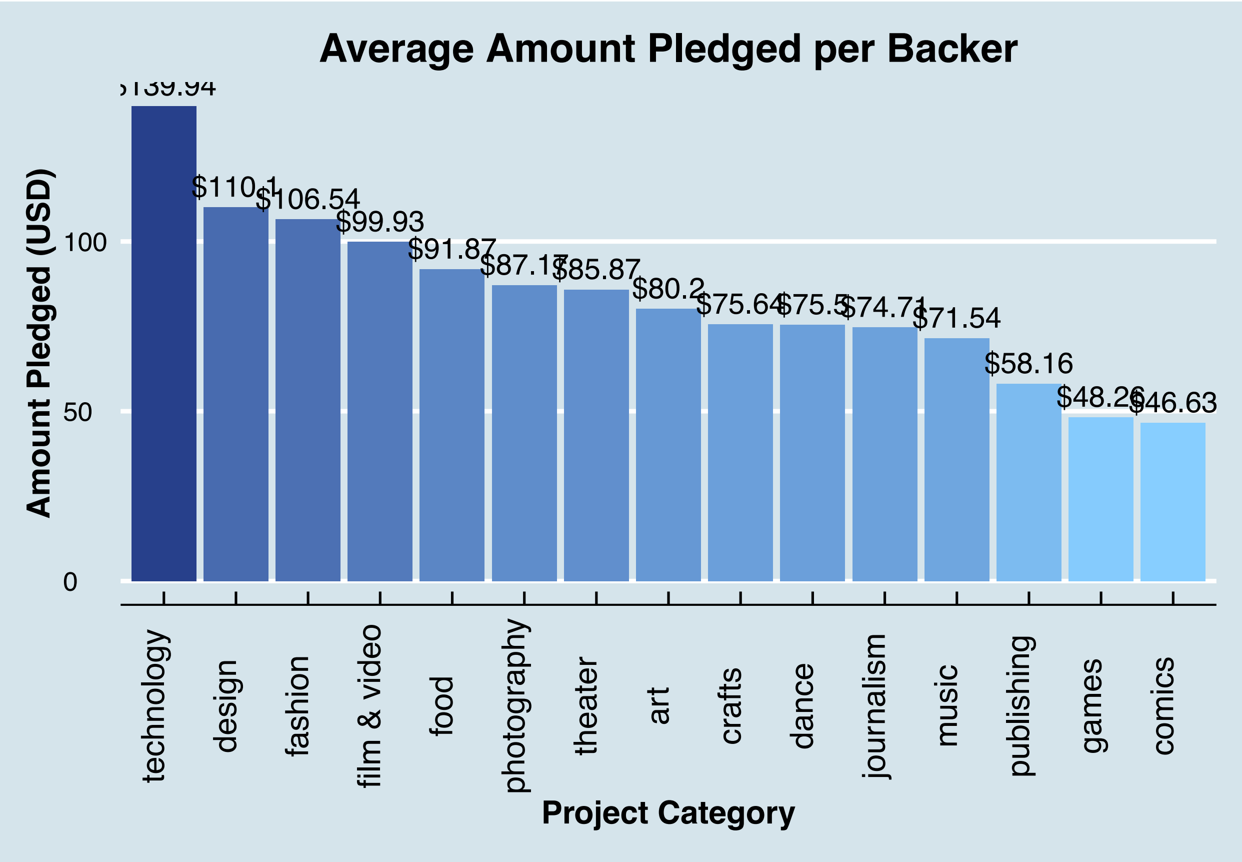
3.4 How much is pledged per backer for each category?

```r
pledged_avg_category <- clean_data %>%
  group_by(clean_data$category) %>%
  summarise(pledged = sum(usd_pledged), backers=sum(backers_count)) %>%
  mutate(avg = pledged/backers) %>%
  arrange(desc(avg))
head(pledged_avg_category)
```

```
## # A tibble: 6 x 4
##   `clean_data$category`   pledged backers   avg
##   <chr>                     <dbl>   <int> <dbl>
## 1 technology            31373736.  224188 140.
## 2 design                 8108525.   73647 110.
## 3 fashion                6797765.   63802 107.
## 4 film & video          10398557.  104058  99.9
## 5 food                   2745202.   29880  91.9
## 6 photography            1620027.   18585  87.2
```

Plot the amount pledged per backer for each category

```r
ggplot(pledged_avg_category, aes(reorder(pledged_avg_category$`clean_data$category`, -avg), avg, fill=av
  ggtitle("Average Amount Pledged per Backer") + xlab("Project Category") +
  ylab("Amount Pledged (USD)") +
  geom_text(aes(label=paste0("$", round(avg,2))), vjust=-0.5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12, angle=90), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```



3.5 Get the 10 highest goal successful projects

```r
top_ten_success <- clean_data[clean_data$state == "successful",] %>%
  select("category", "goal", "state") %>%
  arrange(desc(goal))
head(top_ten_success)
```

```
##      category    goal      state
## 1  technology 1500000 successful
## 2  technology  800000 successful
## 3  technology  800000 successful
## 4 photography  700000 successful
## 5  technology  500000 successful
## 6      design  500000 successful
```

3.6 Get the average project goal

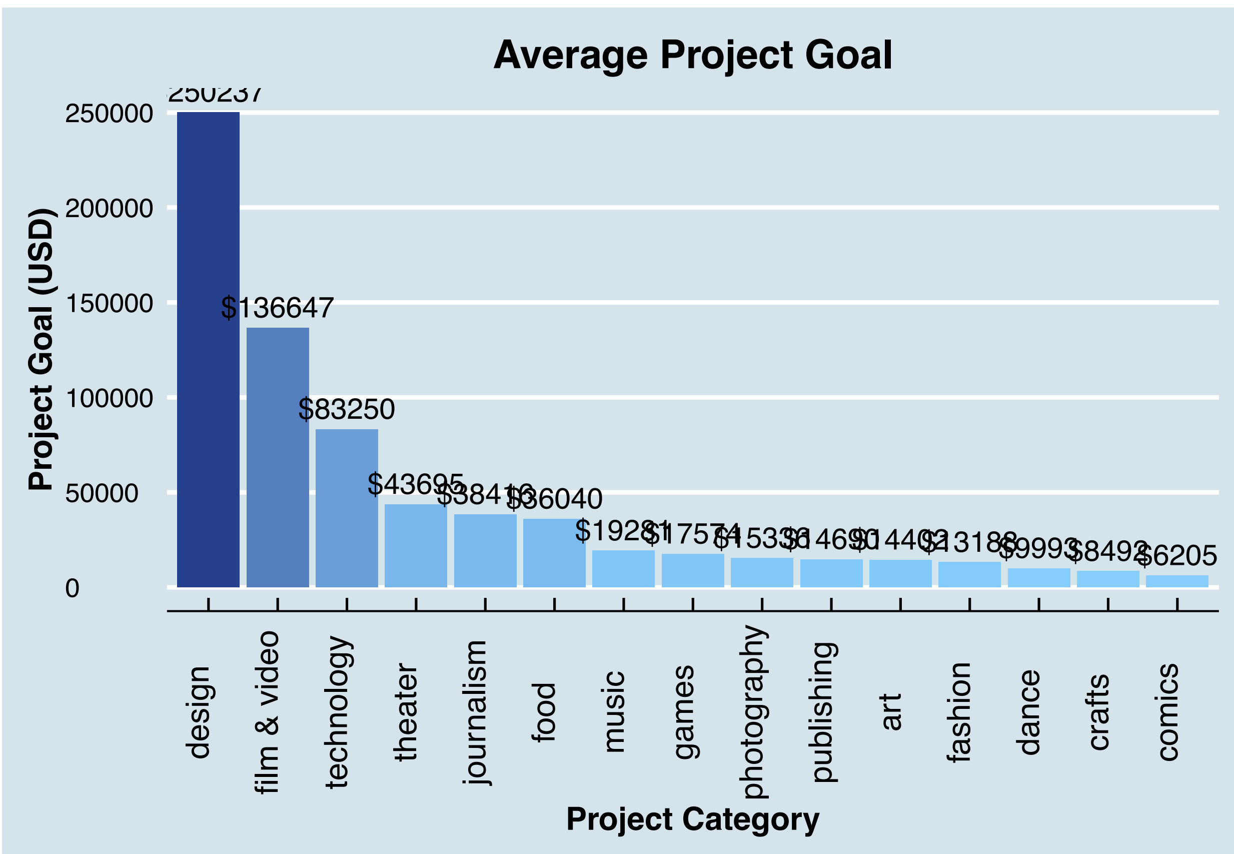
```r
goal_avg <- clean_data %>%
  group_by(category) %>%
  summarise(goals = sum(goal), projects = n()) %>%
```

```r
  mutate(avg = goals/projects) %>%
  arrange(desc(avg))
head(goal_avg)
```

```
## # A tibble: 6 x 4
##   category         goals projects     avg
##   <chr>            <dbl>    <int>   <dbl>
## 1 design       104348985      417 250237.
## 2 film & video 150585489.    1102 136647.
## 3 technology    66183820      795  83250.
## 4 theater        9656489      221  43695.
## 5 journalism     6530680      170  38416.
## 6 food          16506111      458  36040.
```

Plot the average project goal.

```r
ggplot(goal_avg, aes(reorder(goal_avg$category, -avg), avg, fill=avg)) + geom_bar(stat="identity") +
  ggtitle("Average Project Goal") + xlab("Project Category") + ylab("Project Goal (USD)") +
  geom_text(aes(label=paste0("$", round(avg,0))), vjust=-0.5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12, angle=90), legend.position="null") +
  scale_fill_gradient(low="skyblue1", high="royalblue4")
```



3.7 percentage for projects in each category

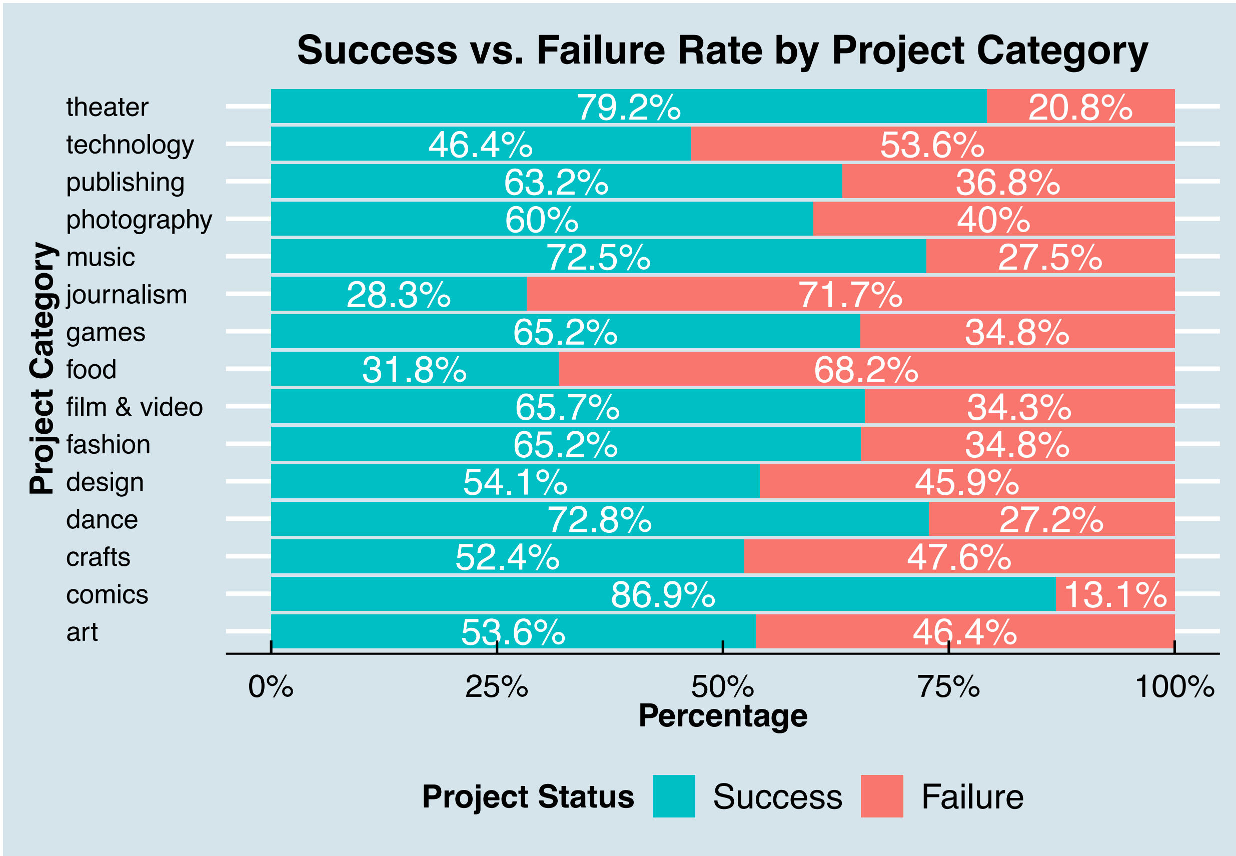
```r
perc_projects <- clean_data %>%
  filter(state %in% c("successful", "failed")) %>%
  group_by(category, state) %>%
```

```
  summarize(count=n()) %>%
  mutate(pct=count/sum(count)) %>%
  arrange(desc(state), pct)
head(perc_projects)
```

```
## # A tibble: 6 x 4
## # Groups:   category [6]
##   category   state      count   pct
##   <chr>      <fct>      <int> <dbl>
## 1 journalism successful    43 0.283
## 2 food       successful   132 0.318
## 3 technology successful   344 0.464
## 4 crafts     successful   199 0.524
## 5 art        successful   390 0.536
## 6 design     successful   198 0.541
```

Plot the percentage for each category

```
ggplot(perc_projects, aes(perc_projects$category, pct, fill=state)) + geom_bar(stat="identity") +
  ggtitle("Success vs. Failure Rate by Project Category") +
  xlab("Project Category") + ylab("Percentage") + scale_y_continuous(labels=scales::percent) +
  scale_fill_discrete(name="Project Status", breaks=c("successful", "failed"),
                      labels=c("Success", "Failure")) +
  geom_text(aes(label=paste0(round(pct*100,1),"%")), position=position_stack(vjust=0.5),
            colour="white", size=5) + theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"),
        axis.text.x=element_text(size=12), legend.position="bottom",
        legend.title=element_text(size=12, face="bold")) + coord_flip()
```

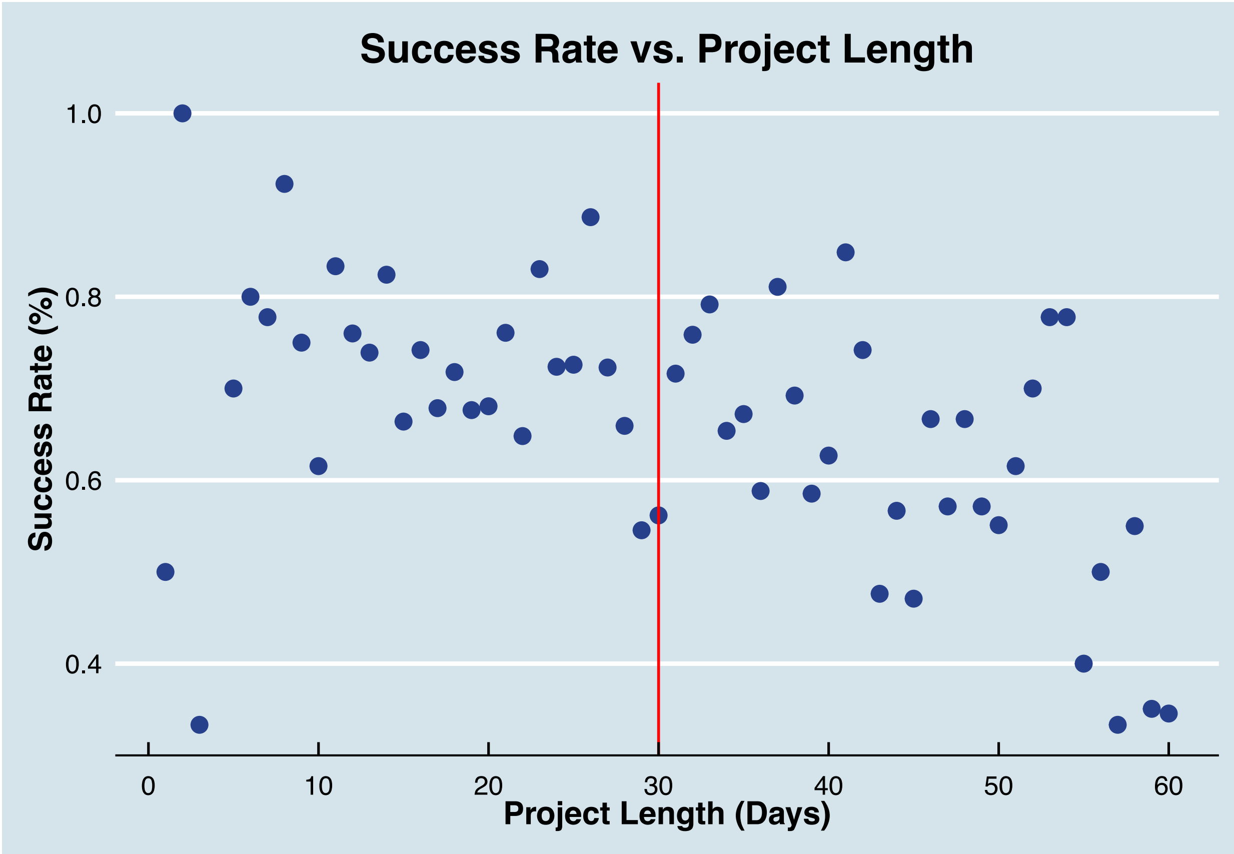


3.8 Does project length affect success rate?

```
perc_length <- clean_data %>%
  filter(state %in% c("successful", "failed"), launch_duration_r < 61) %>%
  group_by(launch_duration_r, state) %>%
  summarize(count=n()) %>%
  mutate(pct=count/sum(count))
head(perc_length)
```

```
## # A tibble: 6 x 4
## # Groups:   launch_duration_r [4]
##   launch_duration_r state      count   pct
##               <dbl> <fct>      <int> <dbl>
## 1                 1 failed         1 0.5
## 2                 1 successful     1 0.5
## 3                 2 successful     1 1
## 4                 3 failed         6 0.667
## 5                 3 successful     3 0.333
## 6                 4 failed         2 1
```

```
ggplot(perc_length[perc_length$state=="successful",], aes(launch_duration_r, pct)) +
  geom_point(colour="royalblue4", size=2.5) + ggtitle("Success Rate vs. Project Length") +
  xlab("Project Length (Days)") + ylab("Success Rate (%)") +
  scale_x_continuous(breaks=c(0,10,20,30,40,50,60)) + geom_vline(xintercept=30, colour="red") +
  theme_economist() +
  theme(plot.title=element_text(hjust=0.5), axis.title=element_text(size=12, face="bold"))
```



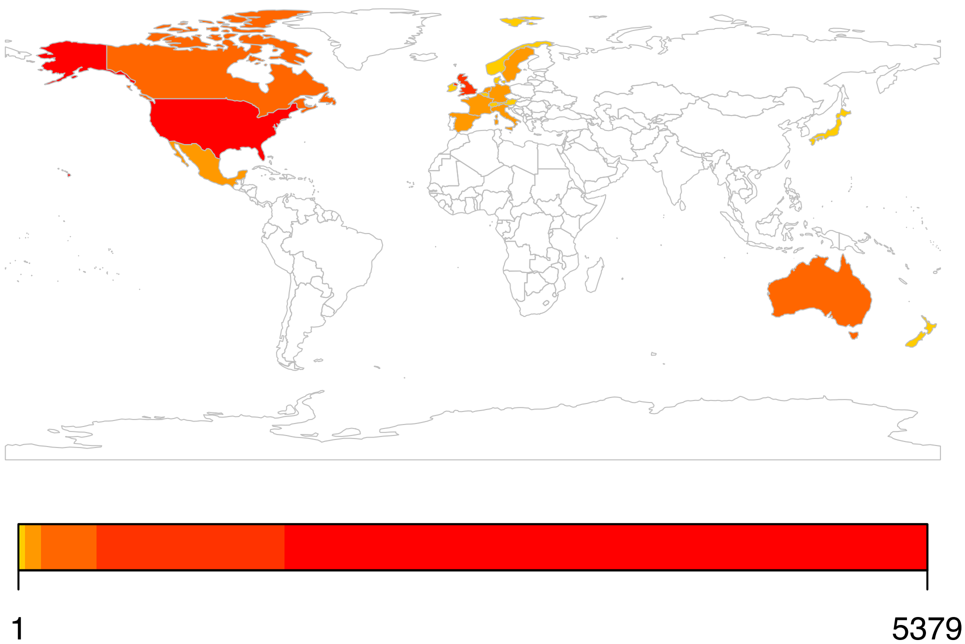
3.9 Where it is coming form?

```
countries_freq <- clean_data %>%
  group_by(country) %>%
  summarize(count=n())

countries.match <- joinCountryData2Map(countries_freq, joinCode="ISO2", nameJoinColumn="country")
```

```
## 22 codes from your data successfully matched countries in the map
## 0 codes from your data failed to match with a country code in the map
## 220 codes from the map weren't represented in your data
```

```
mapCountryData(countries.match, nameColumnToPlot="count",
               mapTitle="Number of Projects by Country", catMethod="logFixedWidth",
               colourPalette="heat")
```

Number of Projects by Country



Part III. Data Model Construction and Prediction

April 17, 2019

1 Predicting Crowdfunding Success

Using kNN, logistic regression model, support vector machine, naive bayes to predict success rate of crowdfunding.

```
In [1]: import re
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import datetime
        import matplotlib.pyplot as plt
        from sklearn.pipeline import Pipeline, FeatureUnion
        from sklearn.pipeline import make_pipeline
        from sklearn.model_selection import train_test_split

        from sklearn.metrics import make_scorer, accuracy_score
        from sklearn.model_selection import cross_val_score
        from sklearn.model_selection import GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsClassifier

        from sklearn.svm import SVC
        from sklearn.metrics import classification_report, confusion_matrix

        from sklearn.naive_bayes import GaussianNB

        %matplotlib inline
```

2 Data Cleaning & Processing

```
In [2]: df = pd.read_csv('./data/data.csv')
        df.head(1)

Out[2]:    Unnamed: 0  backers_count  \
        0           1             21
```

```
                                             blurb category  \
0  2006 was almost 7 years ago... Can you believ...     Rock

   converted_pledged_amount country  created_at currency currency_symbol  \
0                       802      US  1387659690      USD               $

   currency_trailing_code         ...          static_usd_rate  usd_pledged  \
0                    True         ...                      1.0        802.0

        usd_type preparation_duration  preparation_duration_r  \
0  international               351356          4d 1H 35M 56S

   launch_duration  launch_duration_r  created_at_readable  \
0          3888000       45d 0H 0M 0S  2013-12-21 16:01:30

     deadline_readable launched_at_readable
0  2014-02-08 17:37:26  2013-12-25 17:37:26

[1 rows x 41 columns]
```

In [3]:
```
df.shape
```

Out[3]:
```
(3779, 41)
```

In [4]:
```
df.state.value_counts()
```

Out[4]:
```
successful    2224
failed        1276
canceled       149
live           120
suspended       10
Name: state, dtype: int64
```

In [5]:
```
# drop status rows labeled as live, canceled, suspended.
df = df[~df['state'].isin(['live', 'canceled', 'suspended'])]
df.shape
```

Out[5]:
```
(3500, 41)
```

In [6]:
```
# drop irrelevant or independent variables
df.drop(['Unnamed: 0', 'blurb', 'created_at', 'currency_symbol', 'currency_trailing_co
         'deadline', 'disable_communication', 'friends', 'id',
         'is_backing', 'is_starred', 'launched_at', 'state_changed_at',
         'name', 'permissions', 'profile', 'source_url', 'staff_pick',
         'preparation_duration_r', 'launch_duration_r',
         'created_at_readable', 'deadline_readable', 'launched_at_readable',
         'location', 'usd_type'], axis = 1, inplace = True)
df.head()
```

Out[6]:

```
   backers_count     category  converted_pledged_amount country currency  \
0             21         Rock                       802      US      USD
1             97  Mixed Media                      2259      US      USD
2             88   Photobooks                     29638      US      USD
3            193     Footwear                     49158      IT      EUR
4             20     Software                       549      US      USD

    fx_rate     goal  is_starrable  pledged  spotlight       state  \
0  1.000000    200.0         False    802.0       True  successful
1  1.000000    400.0         False   2259.0       True  successful
2  1.000000  27224.0         False  29638.0       True  successful
3  1.128433  40000.0         False  43180.0       True  successful
4  1.000000   1000.0         False    549.0      False      failed

   static_usd_rate   usd_pledged  preparation_duration  launch_duration
0         1.000000    802.000000                351356          3888000
1         1.000000   2259.000000                413843          1728000
2         1.000000  29638.000000                769946          2595600
3         1.136525  49075.152523                314662          3625358
4         1.000000    549.000000                212500          2592000
```

In [7]:

```python
df['state'] = df.state.str.contains('successful').astype(int)
```

In [8]:

```python
# add column representing continent
def classifier(row):
    if row.country in ['US', 'CA', 'GT', 'MX', 'PR', 'NI', 'SV', 'PA', 'BO', 'GU']:
        return 'America'
    elif row.country in ['NG', 'GH', 'ZA', 'KE', 'ET', 'CD', 'MA', 'TZ', 'ZM', 'LR', 'I
        return 'Africa'
    elif row.country in ['GB', 'NO', 'DE', 'SE', 'BA', 'IS', 'HU', 'IT', 'NL','FR', 'UI
       'TR','FI', 'CZ','AM', 'PT','DK','CH', 'SJ', 'RU', 'UA', 'BG','ES','PL', 'GE','II
        return 'Europe'
    elif row.country in ['JM', 'HT','BS','DO','LC', 'DO', 'TT']:
        return 'Carribean'
    elif row.country in ['CN', 'TW', 'HK', 'NP', 'ID', 'SG', 'IN', 'JP', 'LB', 'KZ', 'I
        return 'Asia'
    elif row.country in ['IL','QA', 'AF','KZ','AE','PS','SY','SA', 'IQ','IR','TJ',]:
        return 'Arab'
    else:
        return "Oceania"
df["continent"] = df.apply(classifier, axis=1)
```

In [9]:

```python
df.head()
```

Out[9]:

```
   backers_count     category  converted_pledged_amount country currency  \
0             21         Rock                       802      US      USD
1             97  Mixed Media                      2259      US      USD
2             88   Photobooks                     29638      US      USD
3            193     Footwear                     49158      IT      EUR
```

```
4            20     Software                            549      US      USD

    fx_rate     goal  is_starrable  pledged  spotlight  state  \
0  1.000000    200.0         False    802.0       True      1
1  1.000000    400.0         False   2259.0       True      1
2  1.000000  27224.0         False  29638.0       True      1
3  1.128433  40000.0         False  43180.0       True      1
4  1.000000   1000.0         False    549.0      False      0

   static_usd_rate   usd_pledged  preparation_duration  launch_duration  \
0         1.000000    802.000000                351356          3888000
1         1.000000   2259.000000                413843          1728000
2         1.000000  29638.000000                769946          2595600
3         1.136525  49075.152523                314662          3625358
4         1.000000    549.000000                212500          2592000

  continent
0   America
1   America
2   America
3    Europe
4   America
```

In [10]:
```python
from sklearn import preprocessing
def encode_features(df):
    features = ['category', 'country', 'currency', 'is_starrable', 'continent', 'spotl
    df_combined = pd.concat([df])

    for feature in features:
        le = preprocessing.LabelEncoder()
        le = le.fit(df_combined[feature])
        df[feature] = le.transform(df[feature])
    return df

data = encode_features(df)
data.head()
```

Out[10]:
```
   backers_count  category  converted_pledged_amount  country  currency  \
0             21       120                       802       20        13
1             97        83                      2259       20        13
2             88        99                     29638       20        13
3            193        59                     49158       12         4
4             20       126                       549       20        13

    fx_rate     goal  is_starrable  pledged  spotlight  state  \
0  1.000000    200.0             0    802.0          1      1
1  1.000000    400.0             0   2259.0          1      1
2  1.000000  27224.0             0  29638.0          1      1
```

```
3  1.128433  40000.0             0  43180.0          1      1
4  1.000000   1000.0             0    549.0          0      0

   static_usd_rate   usd_pledged  preparation_duration  launch_duration  \
0         1.000000    802.000000                351356          3888000
1         1.000000   2259.000000                413843          1728000
2         1.000000  29638.000000                769946          2595600
3         1.136525  49075.152523                314662          3625358
4         1.000000    549.000000                212500          2592000

   continent
0          0
1          0
2          0
3          2
4          0
```

In [11]:
```
df.continent.value_counts()
```

Out[11]:
```
0    2677
2     689
3      98
1      36
Name: continent, dtype: int64
```

In [12]:
```
X = df.drop(['preparation_duration', 'launch_duration', 'state', 'backers_count', 'spe
y = df['state']
```

In [13]:
```
from sklearn.preprocessing import Imputer
X = Imputer().fit_transform(X)
```

3 kNN Model

In [14]:
```
k_range = range(1,200)
k_scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=10, scoring = 'accuracy')
    k_scores.append(scores.mean())
print('Computed k_scores for k value in range 1 to 200.')
```

```
Computed k_scores for k value in range 1 to 200.
```

In [15]:
```
scores.mean()
```

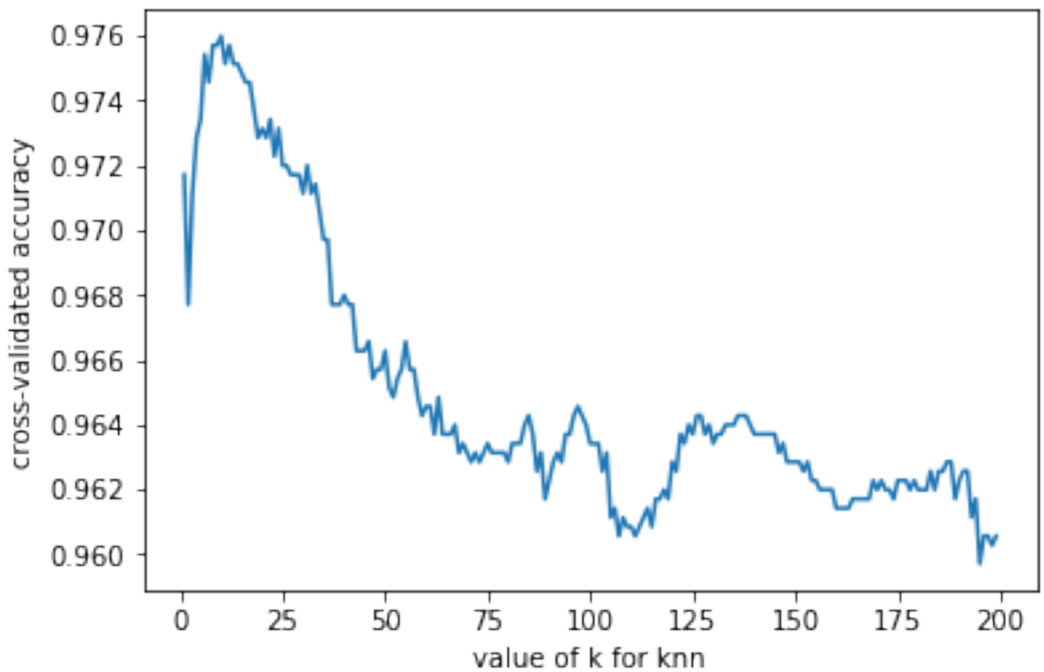
Out[15]:
```
0.9605670786816919
```

In [16]:
```
scores.max()
```

Out[16]: 0.9885386819484241

```
In [16]: plt.plot(k_range, k_scores)
         plt.xlabel('value of k for knn')
         plt.ylabel('cross-validated accuracy')
```

Out[16]: Text(0,0.5,'cross-validated accuracy')



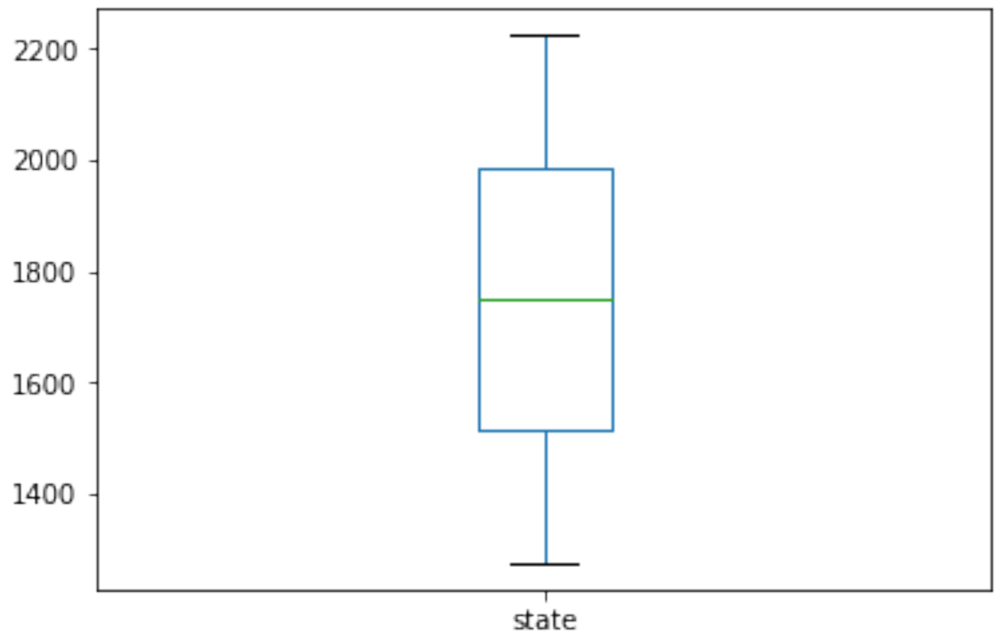

```
In [17]: MSE = [1 - x for x in k_scores]
         optimal_k = k_range[MSE.index(min(MSE))]
         print("The optimal number of neighbors is %d" % optimal_k)
```

The optimal number of neighbors is 10

4 Logistic Regression Model

```
In [18]: df.state.value_counts().plot(kind = 'box')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1a187ffd30>




```
In [19]: ss = StandardScaler()
         lr = LogisticRegression()
         lr_pipe = Pipeline([('sscale', ss), ('logreg', lr)])

In [20]: lr_pipe.fit(X, y)

Out[20]: Pipeline(memory=None,
              steps=[('sscale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('lo
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False))])

In [21]: lr_pipe.score(X,y)

Out[21]: 0.838

In [22]: # divide the dataset into
         #  - 70% training data
         #  - 30% test data
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)

In [23]: lr_pipe.fit(X_train, y_train)

Out[23]: Pipeline(memory=None,
              steps=[('sscale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('lo
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False))])
```

In [24]:
```
lr_pipe.score(X_test, y_test)
```

Out[24]:
```
0.8133333333333334
```

In [25]:
```
y_pred = lr_pipe.predict(X_test)
```

In [26]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, c
```

In [27]:
```
print(f1_score(y_test, y_pred, average="macro"))
print(precision_score(y_test, y_pred, average="macro"))
print(recall_score(y_test, y_pred, average="macro"))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.7979183032207384
0.802507012622721
0.794344333478072
[[282 110]
 [ 86 572]]
             precision    recall  f1-score   support

          0       0.77      0.72      0.74       392
          1       0.84      0.87      0.85       658

avg / total       0.81      0.81      0.81      1050
```

5 Support Vector Machine

In [ ]:
```
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

In [31]:
```
y_pred = svclassifier.predict(X_test)
```

In [32]:
```
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[386   0]
 [  0 664]]
             precision    recall  f1-score   support

          0       1.00      1.00      1.00       386
          1       1.00      1.00      1.00       664

avg / total       1.00      1.00      1.00      1050
```

6 Naive Bayes

```
In [28]: gnb = GaussianNB()
         y_pred = gnb.fit(X_train, y_train).predict(X_test)
```

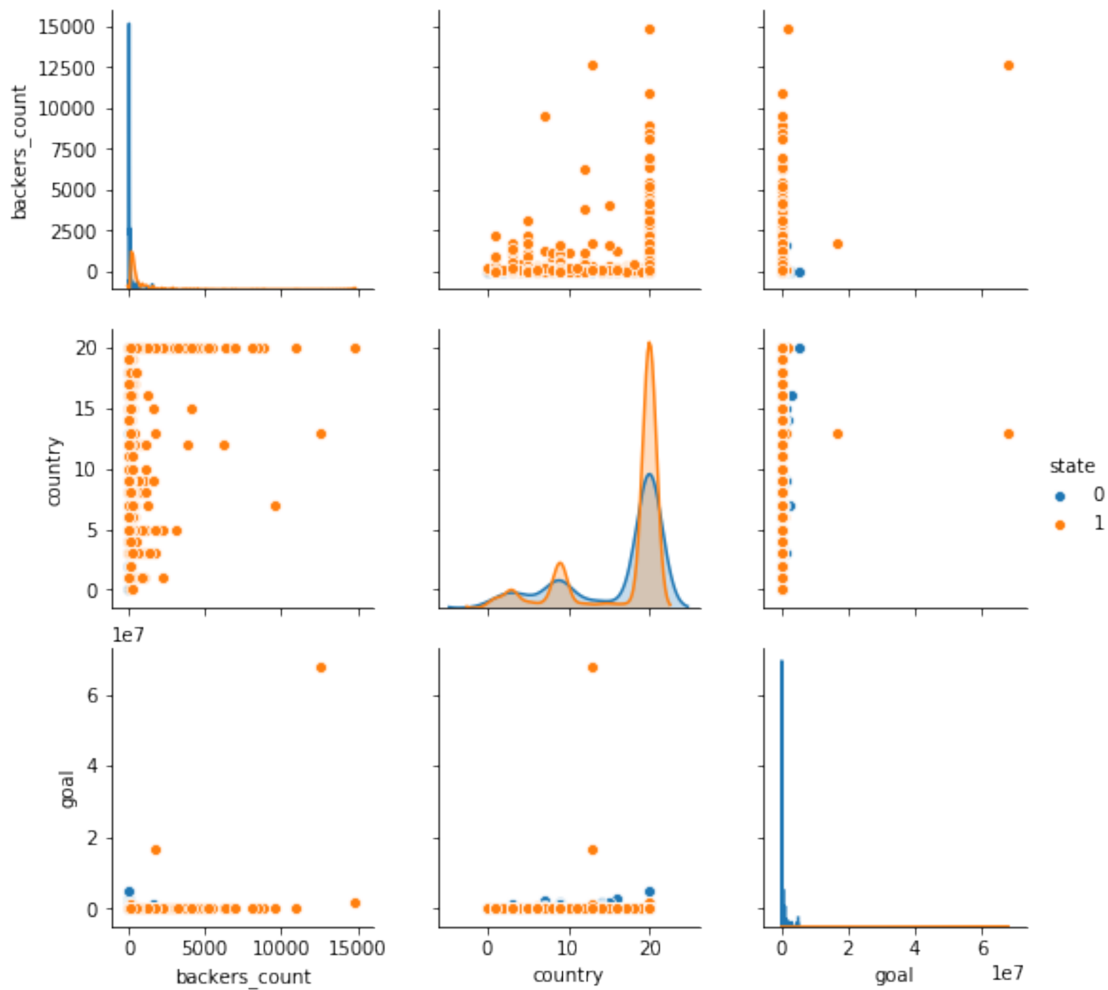
```
In [29]: print(confusion_matrix(y_test,y_pred))
         print(classification_report(y_test,y_pred))
```

```
[[383   9]
 [479 179]]
             precision    recall  f1-score   support

          0       0.44      0.98      0.61       392
          1       0.95      0.27      0.42       658

avg / total       0.76      0.54      0.49      1050
```

7 Data Visualization

```
In [33]: plots = sns.pairplot(df, vars = ['backers_count', 'country', 'goal'],
                              hue="state")
```

```
/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [34]: df = df.drop(['is_starrable'], axis=1)
         corr = df.corr()
         corr
```

Out[34]:

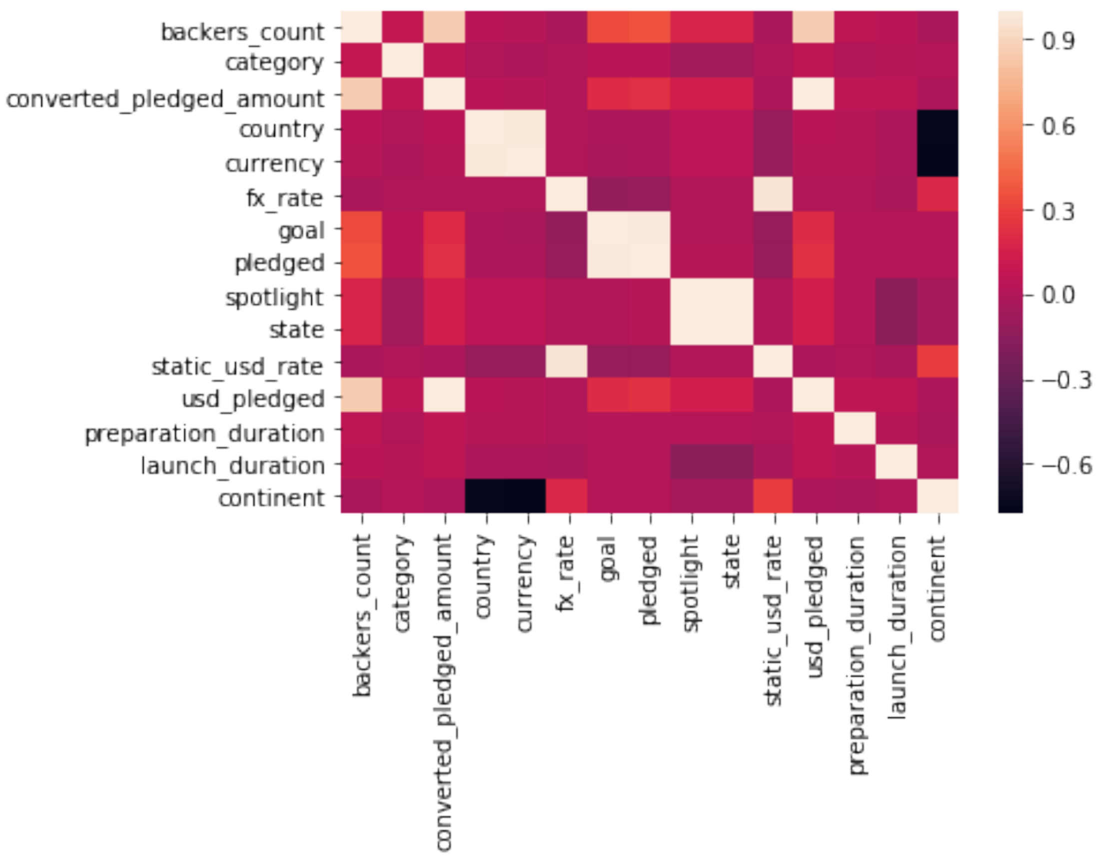
	backers_count	category	converted_pledged_amount \
backers_count	1.000000	0.074279	0.855609
category	0.074279	1.000000	0.056564
converted_pledged_amount	0.855609	0.056564	1.000000
country	0.031723	-0.007631	0.031882
currency	0.027896	-0.011621	0.027261
fx_rate	-0.023226	-0.007235	-0.008231
goal	0.330618	0.039287	0.198685
pledged	0.357206	0.034792	0.229725
spotlight	0.174372	-0.062161	0.140590
state	0.174372	-0.062161	0.140590
static_usd_rate	-0.031860	-0.006960	-0.015069

```
usd_pledged                   0.855154  0.056879                  0.999866
preparation_duration          0.051476 -0.007718                  0.061744
launch_duration               0.043758  0.028363                  0.051582
continent                    -0.023403  0.023315                 -0.015755

                          country  currency   fx_rate      goal   pledged  \
backers_count            0.031723  0.027896 -0.023226  0.330618  0.357206
category                -0.007631 -0.011621 -0.007235  0.039287  0.034792
converted_pledged_amount  0.031882  0.027261 -0.008231  0.198685  0.229725
country                  1.000000  0.984185  0.005072 -0.015072 -0.011316
currency                 0.984185  1.000000  0.001718 -0.020479 -0.016289
fx_rate                  0.005072  0.001718  1.000000 -0.111385 -0.107551
goal                    -0.015072 -0.020479 -0.111385  1.000000  0.991735
pledged                 -0.011316 -0.016289 -0.107551  0.991735  1.000000
spotlight                0.047634  0.048726  0.002059 -0.000156  0.023665
state                    0.047634  0.048726  0.002059 -0.000156  0.023665
static_usd_rate         -0.102557 -0.106477  0.963558 -0.102855 -0.099714
usd_pledged              0.031217  0.026642 -0.008874  0.204915  0.235897
preparation_duration     0.026465  0.028097 -0.002326  0.009646  0.010948
launch_duration         -0.016174 -0.018033 -0.025344  0.020797  0.011791
continent               -0.746953 -0.773095  0.195684  0.015929  0.012159

                          spotlight     state  static_usd_rate  usd_pledged  \
backers_count              0.174372  0.174372        -0.031860     0.855154
category                  -0.062161 -0.062161        -0.006960     0.056879
converted_pledged_amount   0.140590  0.140590        -0.015069     0.999866
country                    0.047634  0.047634        -0.102557     0.031217
currency                   0.048726  0.048726        -0.106477     0.026642
fx_rate                    0.002059  0.002059         0.963558    -0.008874
goal                      -0.000156 -0.000156        -0.102855     0.204915
pledged                    0.023665  0.023665        -0.099714     0.235897
spotlight                  1.000000  1.000000        -0.004343     0.140219
state                      1.000000  1.000000        -0.004343     0.140219
static_usd_rate           -0.004343 -0.004343         1.000000    -0.015418
usd_pledged                0.140219  0.140219        -0.015418     1.000000
preparation_duration       0.012206  0.012206        -0.008198     0.061616
launch_duration           -0.144859 -0.144859        -0.025125     0.051350
continent                 -0.043037 -0.043037         0.285074    -0.015234

                          preparation_duration  launch_duration  continent
backers_count                         0.051476         0.043758  -0.023403
category                             -0.007718         0.028363   0.023315
converted_pledged_amount              0.061744         0.051582  -0.015755
country                               0.026465        -0.016174  -0.746953
currency                              0.028097        -0.018033  -0.773095
fx_rate                              -0.002326        -0.025344   0.195684
goal                                  0.009646         0.020797   0.015929
pledged                               0.010948         0.011791   0.012159
```

spotlight	0.012206	-0.144859	-0.043037
state	0.012206	-0.144859	-0.043037
static_usd_rate	-0.008198	-0.025125	0.285074
usd_pledged	0.061616	0.051350	-0.015234
preparation_duration	1.000000	0.029690	-0.026049
launch_duration	0.029690	1.000000	-0.006020
continent	-0.026049	-0.006020	1.000000

In [35]:
```
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1b3e6358>
```



8 Citation

- Lamidi, Adebola, and Adebola Lamidi. "Predicting the Success of Kickstarter Campaigns." Towards Data Science, Towards Data Science, 20 Sept. 2017, towardsdatascience.com/predicting-the-success-of-kickstarter-campaigns-3f4a976419b9.
- "Kickstarter: Exploratory Data Analysis with R." Kaggle, kaggle.com/andrewjmah/kickstarter-exploratory-data-analysis-with-r.

- “Kickstarter Datasets.” Web Scraping Service, webrobots.io/kickstarter-datasets/.
- Patel, Savan, and Savan Patel. “Chapter 2 : SVM (Support Vector Machine) - Theory.” Medium, Machine Learning 101, 3 May 2017, medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.
- “1.9. Naive Bayesű.” Scikit, scikit-learn.org/stable/modules/naive_bayes.html.
- “Documentation.” Matplotlib, matplotlib.org/.