

ITCS 5102 - SPL FINAL PROJECT

VISUALIZING AND PREDICTING KICKSTARTER PROJECTS DATA USING LIBRARIES & MACHINE LEARNING MODELS

Team Members:

1. Sai Krishna Mannava (801136361)
2. Kumar Mani Chandra Yelisetty (801168244)
3. Smirthi Meenakshisundaram (801129947)

1. Project Description

The main objective of this project is to get an insight about the various project's categories, the amount of investment that each category is getting. What it takes to make a project successful. The backers that a project gets and the influence that it has on the success of a project. We see the factors that influence the success of a project. The most awesome tracking categories and which are the categories most explored. Through this analysis investors and backers also get an idea of which project to back up or invest on. We use Machine learning models to predict the success of a project as well.

The programming language we chose is R language the specialty of which is aiding in statistical computing and graphics. It is also the perfect way of learning machine learning and deep learning concepts. In this project we use R to explore the different concepts that are used for data analysis and derive conclusions from them. We use R libraries to visualize the data and also derive the relationship between various attributes. We use different machine learning algorithms to predict one or more variables depending on the other in the dataset.

The link to the dataset is <https://www.kaggle.com/kemical/kickstarter-projects>. This is a dataset of the Kickstarter projects. Each row in the dataset is about a particular project and the columns give details about the project. Each column is discussed in detail below.

1.1. Dataset Description:

- ❖ **ID:** It is a unique 10-digit number assigned to each Kickstarter project
- ❖ **Name:** The name given to each Kickstarter project
- ❖ **Category:** The subcategory to the main category that each project belongs to
- ❖ **Main category:** All the projects are grouped in various categories like (food, art etc...).
- ❖ **Currency:** As the name says it is the currency type used for investment like (USD, GBP, AUD etc...)
- ❖ **Deadline:** The last day for the project to be submitted, evaluated for a status check, it is of type date.
- ❖ **Goal:** It is the remaining investment required after the pledge amount to successfully complete the project.
- ❖ **Launched:** The date and timestamp in which the project was launched or started.
- ❖ **Pledged:** The amount initially gathered by the project starter to showcase the project idea and to start the project which can be showcased to get the goal amount through crowdfunding to successfully run the project.
- ❖ **Backers:** The number of people who have supported or invested on that particular Kickstarter project.
- ❖ **Country:** The country to which the project belongs or where the project was initiated.
- ❖ **Usd pledged:** Conversion in US dollars of the pledged column (conversion done by Kickstarter).
- ❖ **Usd_pledged_real :** Conversion in US dollars of the pledged column (conversion from [Fixer.io API](#)).
- ❖ **Usd_goal_real:** Conversion in US dollars of the goal column (conversion from [Fixer.io API](#)).

2. Why did we choose R programming Language?

R is one of the most popular languages for statistical modeling and analysis. R is an open-source programming language. This means that anyone can work with R without any need for a license or a fee. R has a vast array of packages. R facilitates quality plotting and graphing. The popular libraries like *ggplot2* and *plotly* advocate for aesthetic and visually appealing graphs that set R apart from other programming languages. R is highly compatible and can be paired with many other programming languages like C, C++, Java, and Python. It can also be integrated with technologies like Hadoop and various other database management systems as well. R is a platform-independent language. It is a cross-platform programming language, meaning that it can be run quite easily on Windows, Linux, and Mac. R provides various facilities for carrying out machine learning operations like classification, regression and also provides features for developing artificial neural networks. R is a constantly evolving programming language. It is a state-of-the-art technology that provides updates whenever any new feature is added.

We considered other languages like python and we realized that R aids in data visualization effectively with its libraries like *ggplot2* and data manipulation with libraries like *dplyr*. Linear regression, Random forest, Decision Tree the algorithms which we planned on implementing could be easily and effectively done using R. So we chose this language.

3. Features of R Programming Language

1. For Data visualization the ggplot2 and dplyr are very useful.

Easy two-color gradients to distinguish positive and negative values in GAM surfaces. Easy access to ribbons with transparency, for confidence intervals. Combine multiple data sets into a single graph with a snap-together, building-block approach. Handsome default settings.

Approach your graph from a visual perspective rather than a programming perspective. Store any ggplot2 object for modification or future recall.

2. Turn a Cartesian graph into a polar graph with a single statement using ggplot2.
3. Wide selection of packages, CRAN or Comprehensive R Archive Network houses more than 10,000 different packages and extensions that help solve all sorts of problems in data science. High-quality interactive graphics, web application development, quantitative analysis or machine learning procedures, there is a package for every scenario available.

R contains a sea of packages for all the forms of disciplines like astronomy, biology, etc. While R was originally used for academic purposes, it is now being used in industries as well.

4. R has a very comprehensive development environment meaning it helps in statistical computing as well as software development. R is an object-oriented programming language.
5. R can be used to perform simple and complex mathematical and statistical calculations on data objects of a wide variety. It can also perform such operations on large data sets.
6. R is machine-independent. It supports the cross-platform operation. Therefore, it can be used on many different operating systems.

4. How we used the Features in our Project

1. We used the ggplot2 and dplyr library to visualize and manipulate the data respectively.

```
# ggplot used in barchart
ggplot(data = project) + geom_bar(mapping = aes(x =
main_category,fill=main_category))

#ggplot for boxplot
plot2<-ggplot(project, aes(x=state, y=backers)) +
  geom_boxplot()
```

2. Conversion of cartesian to polar in a single step. Converting a barchart to a pie chart (polar) using ggplot2 (coord_polar() is used for this purpose)

```
ggplot(data, aes(x="", y=value, fill=group)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void()
```

3. Installing packages for splitting and training the dataset

```
install.packages('caTools')
library(caTools)
set.seed(150)

#splitting the dataset into training 80% and remaining test dataset
split =sample.split(lr_dataset$state, SplitRatio = 0.8)
lr_training_set=subset(lr_dataset, split==TRUE)
lr_test_set=subset(lr_dataset, split=FALSE)
```

4. Packages for visualizing the training set results packages like caret, Rtools, caTools, ElemStatLearn are used.
5. Libraries like e1071, randomForest, rpart are used for the prediction of models.

5. Implementation of the Project

5.1. Doing the Visualizations Using Libraries

Pie chart, bar chart and box plot are plotted using ggplot2 library for data interpretation and visualization. Dplyr library used for manipulation.

5.2. Pre-Processing the Data

We carefully read the dataset and perform some basic pre-processing on the data. The preprocessing that we perform for our project includes

- ❖ Changing the class of required columns (eg. goal, backers and pledged were character by default which we changed to numeric using (as.numeric))
- ❖ Omitting rows with NA values (using na.omit)

5.3. Feature Selection

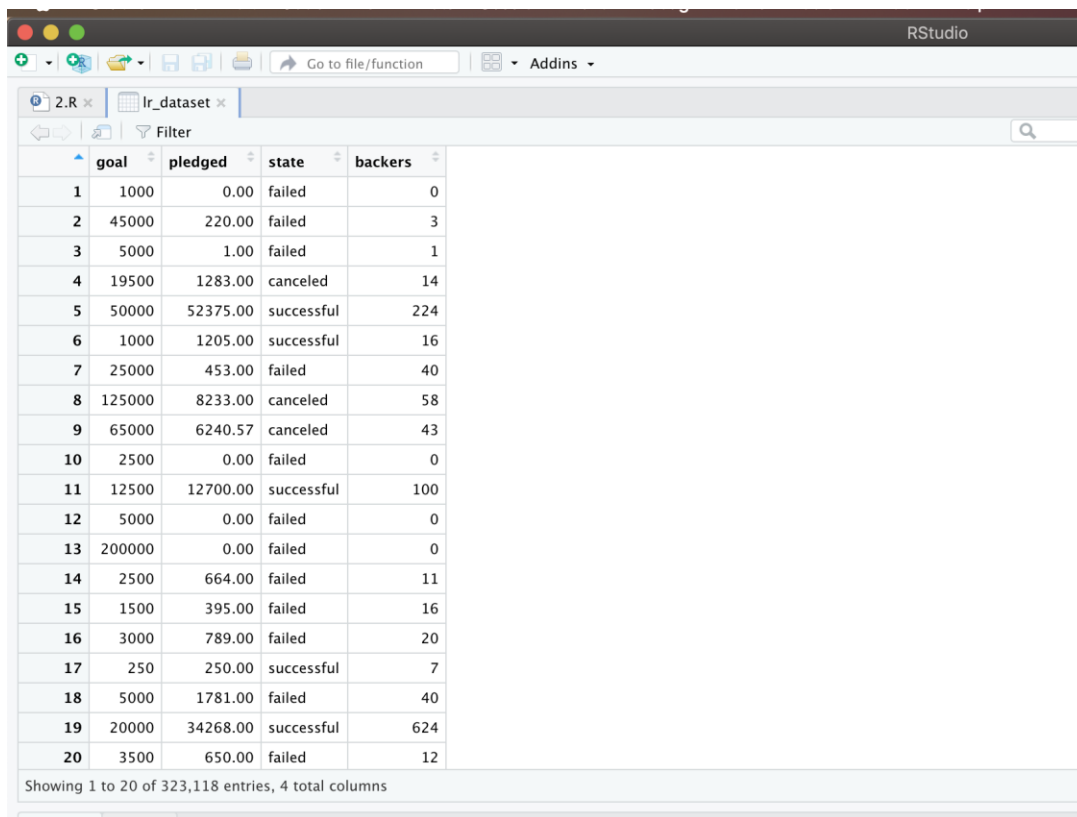
For predicting the success rate, we selected certain features (columns) from the table. The features which we thought would affect the success rate, we have chosen for training and testing.

The following are the features which we chose to predict the success rate

- ❖ Goal
- ❖ Pledged
- ❖ State
- ❖ backers

We predict the state with successful value using the other three features: goal, pledged and backers.

This is a screenshot of the selected features from the dataset.



The screenshot shows the RStudio interface with a dataset named 'lr_dataset' loaded. The dataset has 4 columns: 'goal', 'pledged', 'state', and 'backers'. The table displays 20 rows of data, showing the first 20 entries of a dataset with 323,118 total entries.

	goal	pledged	state	backers
1	1000	0.00	failed	0
2	45000	220.00	failed	3
3	5000	1.00	failed	1
4	19500	1283.00	canceled	14
5	50000	52375.00	successful	224
6	1000	1205.00	successful	16
7	25000	453.00	failed	40
8	125000	8233.00	canceled	58
9	65000	6240.57	canceled	43
10	2500	0.00	failed	0
11	12500	12700.00	successful	100
12	5000	0.00	failed	0
13	200000	0.00	failed	0
14	2500	664.00	failed	11
15	1500	395.00	failed	16
16	3000	789.00	failed	20
17	250	250.00	successful	7
18	5000	1781.00	failed	40
19	20000	34268.00	successful	624
20	3500	650.00	failed	12

Showing 1 to 20 of 323,118 entries, 4 total columns

5.4. Splitting the Data

We have split our dataset into training and testing sets with a proportion of 80% to train the data and 20% to test the data.

5.5. Models Used

We have used the Linear Regression model, Random forest model and Decision tree model to train our dataset and predict the success rate. We train the data and print the accuracy of the models as well.

Linear Regression Model:

The aim of linear regression is to model a continuous variable Y as a mathematical function of one or more X variable(s), so that we can use this regression model to predict the Y when only the X is known. This mathematical equation can be generalized as follows:

$$Y = \beta_1 + \beta_2 X + \epsilon$$

where, β_1 is the intercept and β_2 is the slope. Collectively, they are called regression coefficients. ϵ is the error term, the part of Y the regression model is unable to explain.

Random Forest Model:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that trees protect each from the individual error.

The random forest classifier is an algorithm consisting of many decision trees. It uses bagging and features randomness while building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than any of the individual trees.

Decision Tree Model:

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature. Each leaf node represents a class label and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning. Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision trees are a non-parametric supervised learning method used for both classification and regression tasks.

5.6. Output of the Models

Linear Regression Model:

Accuracy of the Linear regression model is 98.08%

Random Forest Model:

Accuracy of the Linear regression model is 89.69%

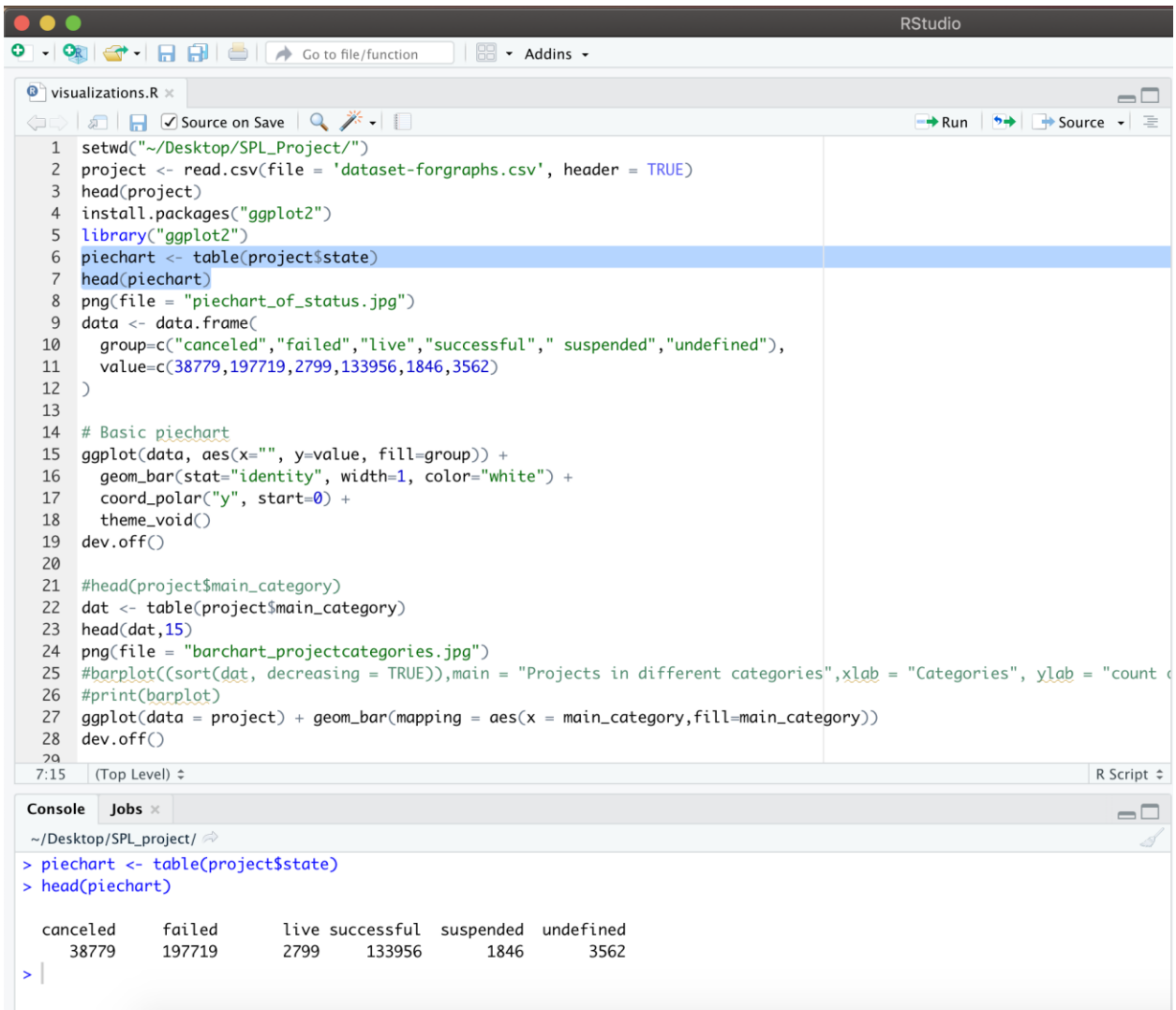
Decision Tree Model:

Accuracy of the Linear regression model is 94.25%

From the accuracies of the models, we can see that the Linear Regression model gave the best results with an accuracy of 98.08% for our dataset to predict the success rate.

6. Step by step project Execution with Screenshots

R performs Mathematical computation on the entire dataset very quickly. Below is the count operation for the no. of projects in each state (eg. successful, canceled etc) which is further used to draw a pie chart



The screenshot displays the RStudio interface with a script editor and a console. The script editor contains R code for reading a CSV file, creating a data frame, and generating a pie chart and a bar chart. The console shows the execution of the first two lines of the script, resulting in a table of project states.

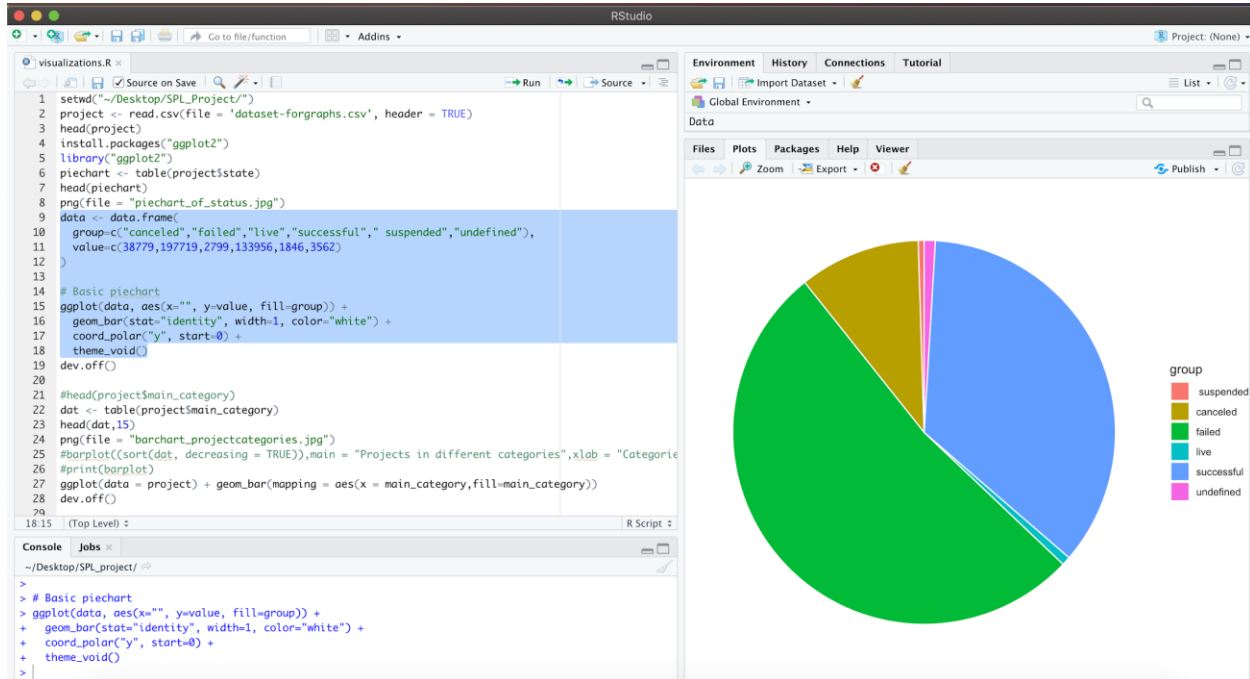
```
1 setwd("~/Desktop/SPL_Project/")
2 project <- read.csv(file = 'dataset-forgraphs.csv', header = TRUE)
3 head(project)
4 install.packages("ggplot2")
5 library("ggplot2")
6 piechart <- table(project$state)
7 head(piechart)
8 png(file = "piechart_of_status.jpg")
9 data <- data.frame(
10   group=c("canceled","failed","live","successful","suspended","undefined"),
11   value=c(38779,197719,2799,133956,1846,3562)
12 )
13
14 # Basic piechart
15 ggplot(data, aes(x="", y=value, fill=group)) +
16   geom_bar(stat="identity", width=1, color="white") +
17   coord_polar("y", start=0) +
18   theme_void()
19 dev.off()
20
21 #head(project$main_category)
22 dat <- table(project$main_category)
23 head(dat,15)
24 png(file = "barchart_projectcategories.jpg")
25 #barplot((sort(dat, decreasing = TRUE)),main = "Projects in different categories",xlab = "Categories", ylab = "count")
26 #print(barplot)
27 ggplot(data = project) + geom_bar(mapping = aes(x = main_category, fill=main_category))
28 dev.off()
29
```

Console output:

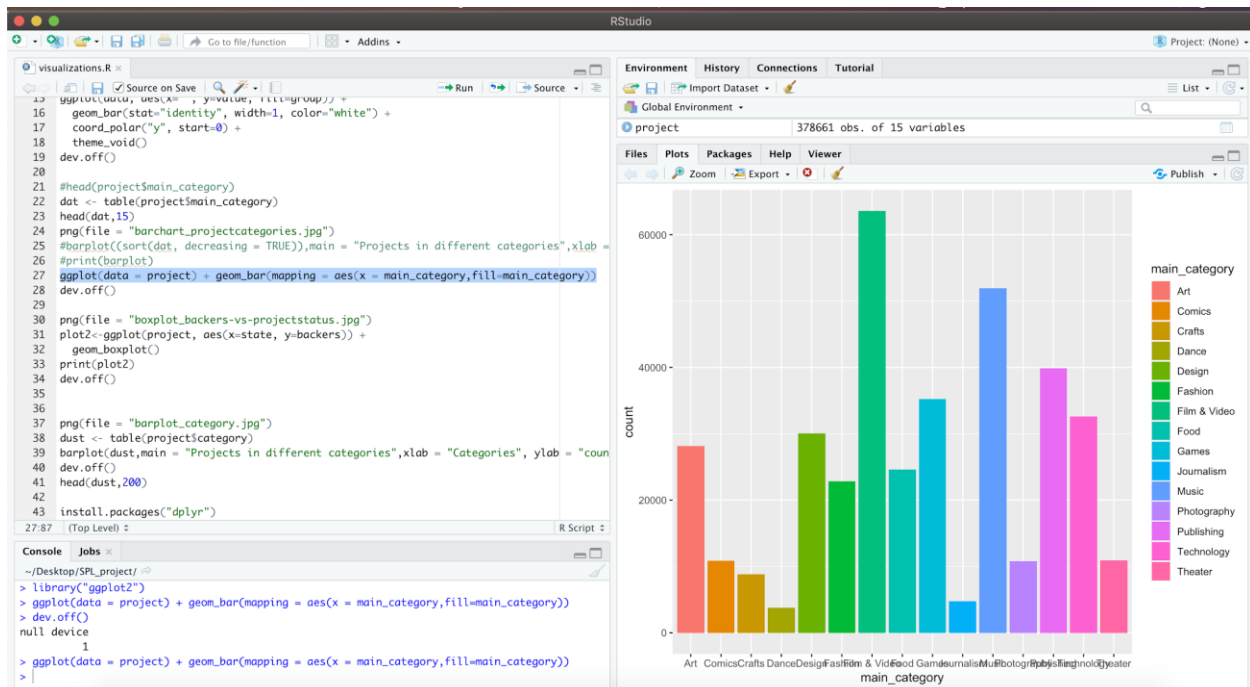
```
> piechart <- table(project$state)
> head(piechart)

  canceled    failed      live successful  suspended  undefined
   38779    197719     2799    133956     1846      3562
```

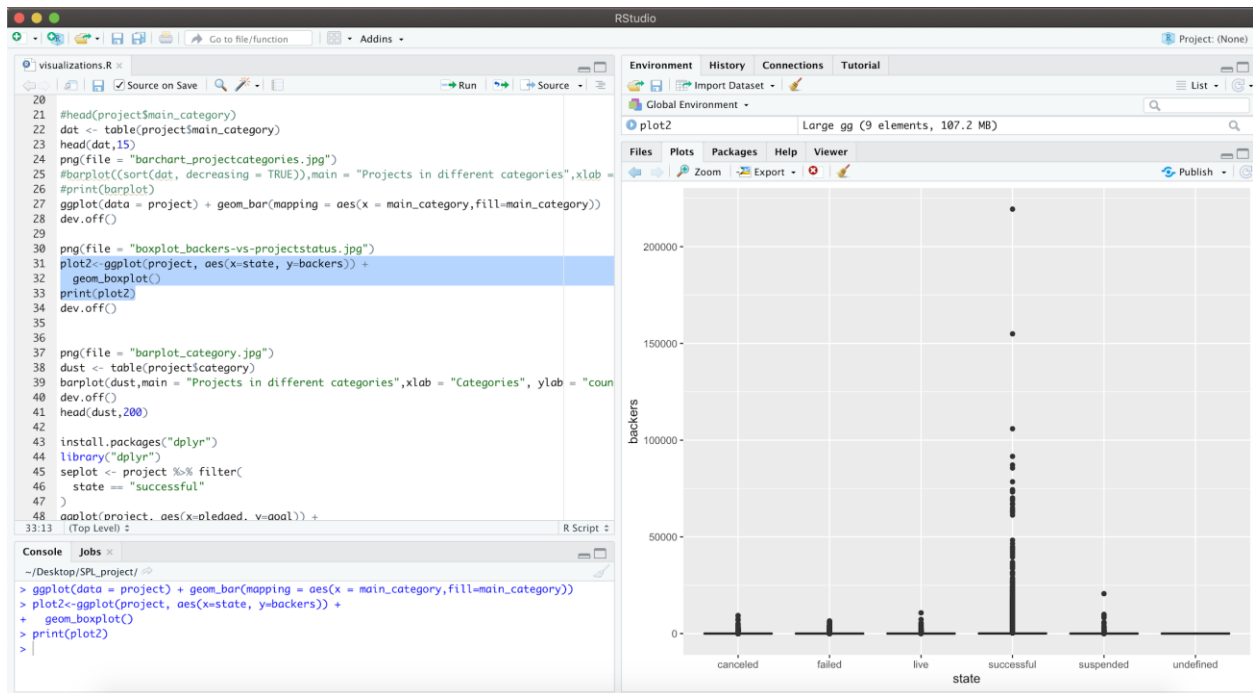
Pie chart for the different states of projects drawn using ggplot2



Bar chart for the different categories drawn using ggplot2



Boxplot for the backers vs state drawn using ggplot2



The pie chart shows that most of the projects have failed, and a good portion of the total projects have been successful as well.

The bar plot shows that Film & Video, Music are the most endorsed categories with maximum number of projects in that category followed by the other ones.

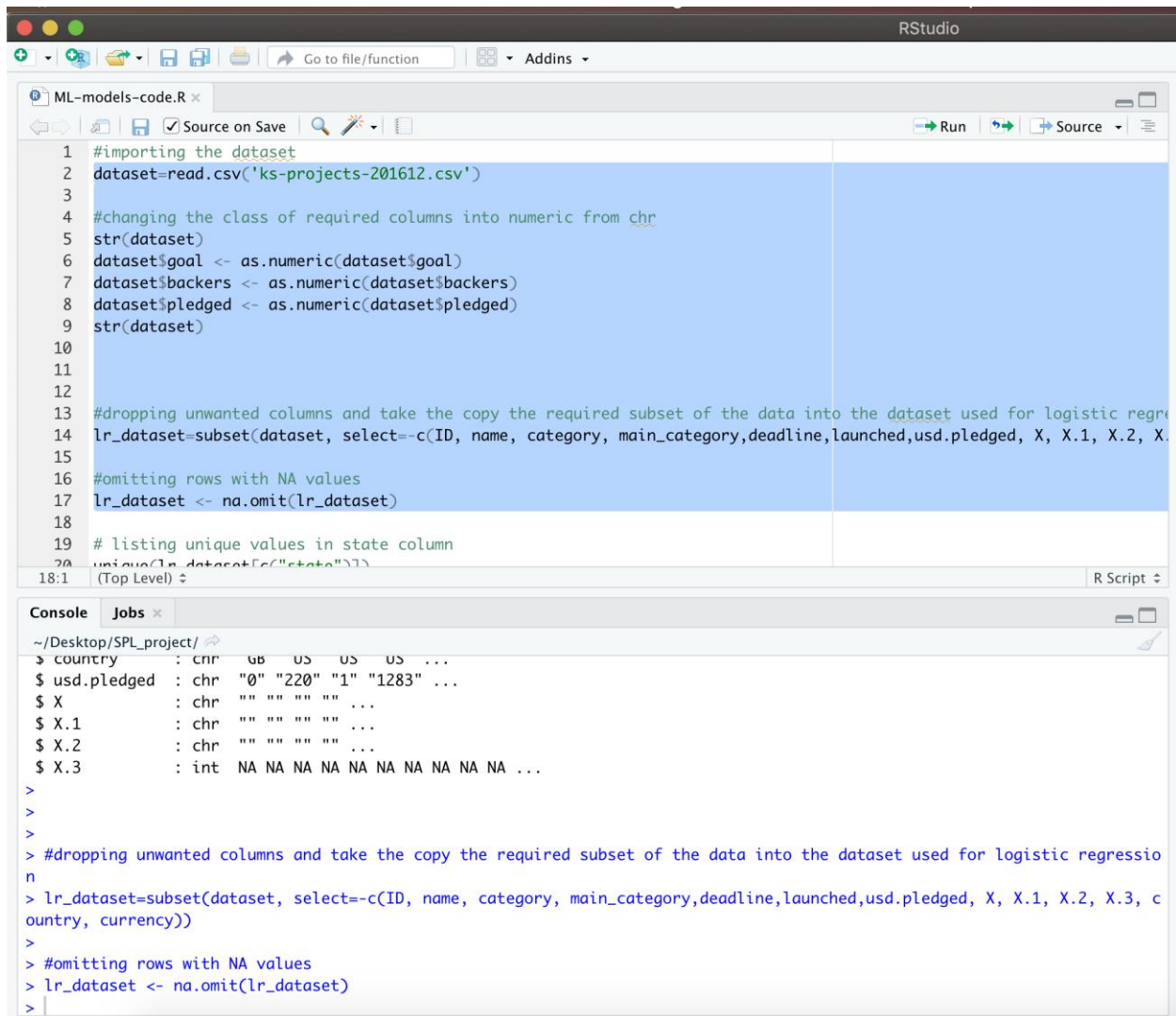
The boxplot shows that the backers influences the state of a project. More the number of backers for a project it increases the success rate of the project.

Machine Learning Algorithms on the data for prediction of success

Preprocessing data

We see some of the columns like goal, backers, pledged which have to be numeric are character by default. They are converted to numeric type.

The NA values are omitted in the chosen features for the dataset.



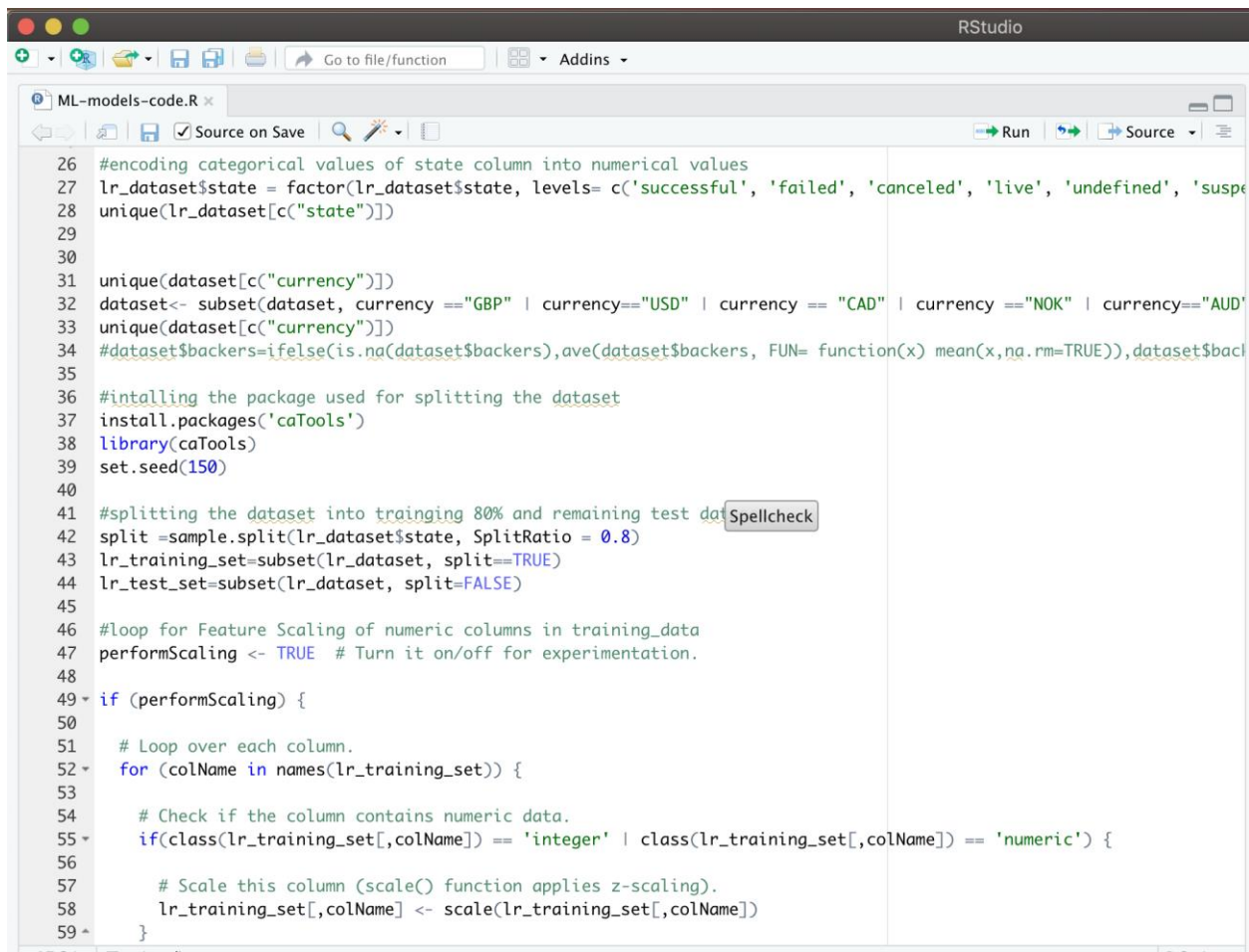
The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
1 #importing the dataset
2 dataset=read.csv('ks-projects-201612.csv')
3
4 #changing the class of required columns into numeric from chr
5 str(dataset)
6 dataset$goal <- as.numeric(dataset$goal)
7 dataset$backers <- as.numeric(dataset$backers)
8 dataset$pledged <- as.numeric(dataset$pledged)
9 str(dataset)
10
11
12
13 #dropping unwanted columns and take the copy the required subset of the data into the dataset used for logistic regression
14 lr_dataset=subset(dataset, select=-c(ID, name, category, main_category,deadline,launched,usd.pledged, X, X.1, X.2, X.3))
15
16 #omitting rows with NA values
17 lr_dataset <- na.omit(lr_dataset)
18
19 # listing unique values in state column
20 unique(lr_dataset[c("state")])
21
```

The console on the right shows the output of the code:

```
~/Desktop/SPL_project/ >
> country      : chr  GB  US  US  US  ...
$ usd.pledged  : chr  "0" "220" "1" "1283" ...
$ X            : chr  "" "" "" "" ...
$ X.1          : chr  "" "" "" "" ...
$ X.2          : chr  "" "" "" "" ...
$ X.3          : int  NA NA NA NA NA NA NA NA NA ...
>
>
> #dropping unwanted columns and take the copy the required subset of the data into the dataset used for logistic regression
> lr_dataset=subset(dataset, select=-c(ID, name, category, main_category,deadline,launched,usd.pledged, X, X.1, X.2, X.3, country, currency))
>
> #omitting rows with NA values
> lr_dataset <- na.omit(lr_dataset)
>
```

Encoding the categorical values of the state column that we are going to predict. We encode the success as 0 and any other kind of failure such as cancelled, failed, live, undefined as 1. We split the dataset to training and testing set with 80% of the data for training and 20% of the data for testing. We perform scaling on the training and testing sets.

The image shows a screenshot of the RStudio interface. The top bar has the RStudio logo and the text 'RStudio'. Below it is a toolbar with icons for file operations and a search bar. The main editor window shows a script titled 'ML-models-code.R'. The code is as follows:

```
26 #encoding categorical values of state column into numerical values
27 lr_dataset$state = factor(lr_dataset$state, levels= c('successful', 'failed', 'canceled', 'live', 'undefined', 'suspended'))
28 unique(lr_dataset[c("state")])
29
30
31 unique(dataset[c("currency")])
32 dataset<- subset(dataset, currency == "GBP" | currency=="USD" | currency == "CAD" | currency == "NOK" | currency=="AUD" | currency=="JPY")
33 unique(dataset[c("currency")])
34 #dataset$backers=ifelse(is.na(dataset$backers),ave(dataset$backers, FUN= function(x) mean(x,na.rm=TRUE)),dataset$backers)
35
36 #installing the package used for splitting the dataset
37 install.packages('caTools')
38 library(caTools)
39 set.seed(150)
40
41 #splitting the dataset into training 80% and remaining test data
42 split =sample.split(lr_dataset$state, SplitRatio = 0.8)
43 lr_training_set=subset(lr_dataset, split==TRUE)
44 lr_test_set=subset(lr_dataset, split==FALSE)
45
46 #loop for Feature Scaling of numeric columns in training_data
47 performScaling <- TRUE # Turn it on/off for experimentation.
48
49 if (performScaling) {
50
51   # Loop over each column.
52   for (colName in names(lr_training_set)) {
53
54     # Check if the column contains numeric data.
55     if(class(lr_training_set[,colName]) == 'integer' | class(lr_training_set[,colName]) == 'numeric') {
56
57       # Scale this column (scale() function applies z-scaling).
58       lr_training_set[,colName] <- scale(lr_training_set[,colName])
59     }
60   }
61 }
```

Linear Regression model

The logistic regression is fit to the training set. We create a matrix of the probability predicted from the dataset and the probability of the test set (`y_pred`, `lr_test_set[,3]`). We create the confusion matrix and print the output for the Linear regression model along with the accuracy and details.

The screenshot shows an R Studio interface with a script editor and a console. The script editor contains R code for fitting a logistic regression model and evaluating its performance. The console displays the output of the code, including a confusion matrix and various performance metrics.

```
81
82 #Fitting the logistic regression to the training set
83 classifier = glm(formula=state ~ ., family =binomial, data=lr_training_set, na.action = na.pass )
84
85 lr_test_set
86 lr_test_set[3]
87 unique(lr_test_set[3])
88
89 #Predicting the test set results
90 prob_pred = predict(classifier, type='response', newdata =lr_test_set[-3] )
91
92 #probability results
93 prob_pred
94 min(prob_pred)
95 max(prob_pred)
96 #converting the propabilitiy results to 0 to 1
97 #y_pred =ifelse(prob_pred < 0.167, 0, ifelse (0.167<=prob_pred < 0.33, 1,ifelse(0.33<=prob_pred<0.5,2,ifelse(0.5<= pr
98 y_pred <- ifelse(prob_pred<0.5,0,1)
99 unique(y_pred)
100 unique(lr_test_set[,3])
101 #making the confusion matrix
102 cm =table(y_pred,lr_test_set[,3])
103 cm
104 install.packages('e1071')
105 library(e1071)
106
107
108 result1 <- confusionMatrix(cm)
109 result1
```

109:8 (Top Level) ↕ R Script ↕

Console Jobs x

~/Desktop/SPL_project/ ↗

```
y_pred      0      1
0 111762  4872
1   1319 205165

      Accuracy : 0.9808
      95% CI   : (0.9804, 0.9813)
No Information Rate : 0.65
P-Value [Acc > NIR] : < 2.2e-16

      Kappa   : 0.9582

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

      Sensitivity : 0.9883
```

Random Forest model

We use the random forest library. The random forest is fit to the training set. We create a matrix of the probability predicted from the dataset and the probability of the test set (RF_prob_pred, RF_test_set[,3]). We create the confusion matrix and print the output for the RF model along with the accuracy and details.

```
ML-models-code.R x
Source on Save
Run
Source

166 #install.packages('randomForest')
167 library(randomForest)
168
169 RF_classifier = randomForest(x= RF_training_set[-3], y=RF_training_set$state, ntree= 100 )
170
171 RF_prob_pred =predict(RF_classifier, type='response', newdata =RF_test_set[-3] )
172 unique(RF_prob_pred)
173
174 RF_cm =table(RF_prob_pred,RF_test_set[,3])
175 RF_cm
176 install.packages('caret')
177 library(caret)
178 install.packages('e1071')
179 library(e1071)
180
181 result <- confusionMatrix(RF_cm)
182 result
183
184 #install.packages('rpart')
185 library(rpart)

182:7 (Top Level) R Script

Console Jobs
~/Desktop/SPL_project/

RF_prob_pred      0      1
0 113032 33286
1      17 176664

      Accuracy : 0.8969
      95% CI : (0.8958, 0.8979)
No Information Rate : 0.65
P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.7878

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

      Sensitivity : 0.9998
      Specificity : 0.8415
      Pos Pred Value : 0.7725
      Neg Pred Value : 0.9999
      Prevalence : 0.3500
      Detection Rate : 0.3499
      Detection Prevalence : 0.4530
      Balanced Accuracy : 0.9207

      'Positive' Class : 0
```

Decision Tree Model

We use the rpart library. The decision tree is fit to the training set. We create a matrix of the probability predicted from the decision tree and the RF test set (RF_test_set[,3],dt_ypred). We create the confusion matrix and print the output for the decision tree model along with the accuracy and details.

RStudio

ML-models-code.R

```
180
181 result <- confusionMatrix(RF_cm)
182 result
183
184 #install.packages('rpart')
185 library(rpart)
186
187 dt_classifier = rpart(formula = state ~ ., data = RF_training_set)
188 dt_ypred = predict(dt_classifier, newdata = RF_test_set[-3], type = 'class')
189
190 dt_cm = table(RF_test_set[,3], dt_ypred)
191 dt_cm
192 library(caret)
193 library(e1071)
194
195 result2 <- confusionMatrix(dt_cm)
196
196:8 (Top Level) R Script
```

Console

~/Desktop/SPL_project/

```
dt_ypred
  0    1
0 110430 2619
1 15948 194002

Accuracy : 0.9425
95% CI : (0.9417, 0.9433)
No Information Rate : 0.6087
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.877

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

Sensitivity : 0.8738
Specificity : 0.9867
Pos Pred Value : 0.9768
Neg Pred Value : 0.9240
Prevalence : 0.3913
```

7. Output

Screenshot of Linear Regression Model

```
Console  Jobs x
~/Desktop/SPL_project/

> result1 <- confusionMatrix(cm)
> result1
Confusion Matrix and Statistics

y_pred      0      1
0 111762  4872
1  1319 205165

      Accuracy : 0.9808
      95% CI   : (0.9804, 0.9813)
No Information Rate : 0.65
P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.9582

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

      Sensitivity : 0.9883
      Specificity : 0.9768
      Pos Pred Value : 0.9582
      Neg Pred Value : 0.9936
      Prevalence : 0.3500
      Detection Rate : 0.3459
      Detection Prevalence : 0.3610
      Balanced Accuracy : 0.9826

      'Positive' Class : 0
```

Screenshot of Random Forest Model

```
11F
182:7 (Top Level)

Console  Jobs x
~/Desktop/SPL_project/

> result <- confusionMatrix(RF_cm)
> result
Confusion Matrix and Statistics

RF_prob_pred      0      1
0 113028  33265
1    21 176685

      Accuracy : 0.8969
      95% CI   : (0.8959, 0.898)
No Information Rate : 0.65
P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.7879

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

      Sensitivity : 0.9998
      Specificity : 0.8416
      Pos Pred Value : 0.7726
      Neg Pred Value : 0.9999
      Prevalence : 0.3500
      Detection Rate : 0.3499
      Detection Prevalence : 0.4529
      Balanced Accuracy : 0.9207

      'Positive' Class : 0
```

Screenshot of Decision Tree Model

```
Console Jobs x
~/Desktop/SPL_project/
> result2 <- confusionMatrix(dt_cm)
> result2
Confusion Matrix and Statistics

      dt_ypred
      0      1
0 110430 2619
1 15948 194002

      Accuracy : 0.9425
      95% CI : (0.9417, 0.9433)
      No Information Rate : 0.6087
      P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.877

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

      Sensitivity : 0.8738
      Specificity : 0.9867
      Pos Pred Value : 0.9768
      Neg Pred Value : 0.9240
      Prevalence : 0.3913
      Detection Rate : 0.3419
      Detection Prevalence : 0.3500
      Balanced Accuracy : 0.9302

      'Positive' Class : 0
```

8. Tools and Technologies used

We installed the R packages, and R Studio platform to do the coding.

Downloaded and installed R <https://cran.r-project.org/>

Downloaded R studio <https://rstudio.com/products/rstudio/download/>

9. Conclusion

Kick Starter projects are well known for their success stories. In our project we have developed an industry level acceptable model for predicting the success of these projects based on various features. R programming Language made it very easy and effective for us to do the visualizations and predictions.

Combining all the information provided in the project in the form of graphs, plots, statistical and data analysis we can say that for any project to be successful the most important feature is the number of backers and the amount pledged for the project by the backers. But, we can't affirm it inversely because there is no convincing correlation between other states(failure, suspended) of the project with backers or any other feature. Our regression model and data tree classification has achieved an accuracy of 98.08% and 94.25% which is excellent for predicting the state of the future projects.

References:

1. <https://www.kaggle.com/kemical/kickstarter-projects>
2. <https://data-flair.training/blogs/pros-and-cons-of-r-programming-language/>
3. <https://techvidvan.com/tutorials/r-features/>
4. <http://r-statistics.co/Linear-Regression.html#Introduction>
5. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
6. <https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96>