Analysis on What Makes a Kickstarter Campaign Successful

1 Introduction

Kickstarter, founded in 2009, is one of the well-known global crowdfunding platforms that focus on bringing funding to creative projects. It is a good way to generate funds for projects created by individuals and small companies, as well as to advertise their businesses prior to opening. Kickstarter uses an "all-or-nothing" funding system, whereby a campaign is funded if only it meets the goal amount set by the creator. Otherwise, no money will be given by backers.

There are a huge variety of factors contributing to the success or failure of a campaign. In common sense, campaigns with a large number of backers are easier to be successful since the goal amount can be achieved more quickly. In this analysis, we first investigate what factors generally determine the outcome of a campaign, whether successful or failed. Then, we study the impact of the goal amount and the number of backers on the probability of making a successful campaign.

This analysis is intended to help a new community-based business in Detroit to get start-up funds from Kickstarter by determining the factors that lead to a successful campaign in the US and to predict whether they could get funds if they used Kickstarter. By a sub-analysis of the US, we found the goal amount, the length of the campaign name and the blurb, and the deadline year of the campaigns are significantly important to the campaign results. Moreover, it is very likely for a campaign to succeed if it has more backers with a large goal amount.

2 Methods

There are two main goals of our analysis. One is to figure out what attributes of a campaign can help it succeed on Kickstarter such as its category and features of the campaign's name and blurb. Another is to measure the chance of success of our clients if they would like to raise at least 25,000 dollars from at least 1000 backers. To address our goals, we conducted logistic regression and random forest models for this binary classification task. This aligned with the data and goals posed.

The past Kickstarter campaign data were collected from a public dataset, where each campaign's state was recorded, whether successful or not (1 or 0). This could be used as our dependent variable and our objective was to see what features impact the state of the campaign. For such a classification task where the dependent variable is discrete and binary, we would like to apply logistic regression and tree-based methods.

For a binary logistic regression, factor level 1 of the dependent variable (what we want to predict) represents the desired outcome which is a successful campaign in our dataset. Logistic regression works by measuring the relationship between the dependent variable and one or more independent variables (features). In addition, only meaningful variables should

be included in it as they might lead to errors. In our case, some JSON-structured variables and urls should be removed. We also need to eliminate multicollinearity in this model which requires that there should be no or less correlation between the independent variables. We would look deeper into the correlation for selecting variables in the next section. This model estimates the coefficient of selected variables and we could see whether it was statistically significant or not for the success of a campaign at the 95% confidence level.

Tree-based methods are a popular approach for classification tasks handling more complex and non-linear datasets. A basic model is the decision tree model. We go through each decision node (yes or no question) in the tree until we reach our final predictions (whether successful or not). This allows us to understand and interpret the model and outcomes can be easily explained. Moreover, decision trees are good at finding the most important features from all input features by creating a split on the one that separates the class labels the best in terms of entropy or Gini. However, singular decision tree models are prone to overfitting when there are many features but few observations. A slight change in the input dataset might greatly impact the final outcomes as well. To improve those, ensemble methods are proposed by building many trees and combining the results together. One powerful model is the random forest model.

A random forest algorithm consists of many decision trees. Bagging is the foundation of it that we build many decision trees at a time by randomly sampling with replacements from the original dataset. Based on this idea, on top of building many trees from sampled datasets, each node is only allowed to split on a random selection of the model's features. Then the random forest establishes the outcome by taking the majority votes from various trees. It eradicates the limitations of a decision tree model and reduces the risk of overfitting by ensuring variety in the trees.

Before we conducted the model, the standardization technique was applied in order to get faster convergence in the logistic regression model. Variable selection was conducted in order to find meaningful variables and we would go deeper into it in the next section. Moreover, there were several categorical variables and we used one-hot encoding to invert it to 0 or 1.

3 Results

3.1 Overview of the Data

The data set contains 20632 rows and 67 variables where each row represents one campaign and each column represents features of a certain campaign. 14141 campaigns were from the US. This subset of data is the focus of our analysis as it helps our clients to have an idea of their chance of making a successful campaign in the US. To compare the difference between campaigns from the US and those from other countries, we added a dummy variable indicating whether it's from the US or not.

Data Pre-processing First, we removed rows that provided litter information about the campaigns, where the 'id' are 11557, 17867, 17887, 11548, 11848, 6744 and 8722. Those rows had only the NA values or were marked as "canceled" / "test". Then, we dropped columns that were almost null, which are 'friends', 'is_starred', 'is_backing', 'permissions'.

We also dropped variables that were highly correlated, such as the weekday of the deadline and whether the deadline was on weekend (see more in the Appendix). Another example is the country of the campaign and whether this country is the US or Great Britain.

Datetime variables stored in Unix time were removed as well as we could use other variables extracted from the datetime ones. JSON-structured variables such as 'photo' and 'profile' were removed as they were meaningless for modeling. In addition, we dropped variables that had further information about the result of a campaign, such as the number of backers and the amount of pledged money, but we could still use those variables in the exploratory data analysis section. Finally, we had a cleaned dataset for modeling with 22 variables and 20625 rows where there were 14136 campaigns from the US.

Baseline Table The variables of interest are the state of campaign, the goal amount set by the creator, the length of a campaign's name and blurb, the year of the campaign's deadline, and the hour of the campaign's state changing and launching.

Below is the baseline table that displays the median and inner quartile range (pr percentage of categorical variables) for the variables of interest.

Variable	Median (IQR) or Percentage
the goal amount (in 1000$)	14(4, 50)
the cleaned length of campaign's name	5(3, 7)
the cleaned length of campaign's blurb	13(11, 15)
the year of the campaign's deadline	2015(2014, 2016)
the hour of the campaign's state changing	13(9, 17)
the hour of the campaign's launching	12(9, 16)
successful campaign (%)	29.18
the number of backers	12(2, 63)
the pledged amount (in 1000$)	0.72(0.025, 6)

Table 1: Baseline Table

3.2 Explanatory Data Analysis

We first have a look at the distribution of the campaign's state in different groups shown in Figure 1.

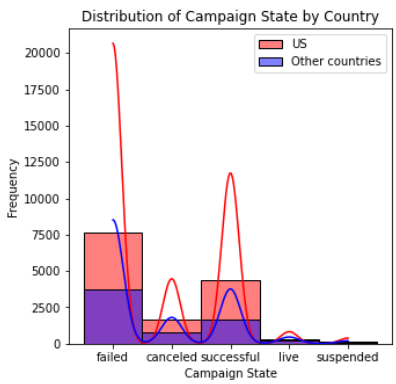


Figure 1: Comparing distributions of the campaign's state between the US and other countries

We can find there are more successful campaigns in the US compared to other countries while there is an approximate same chance of failure.

By looking at the histograms of the goal amount in the US and in the whole dataset (see in the Appendix), we find there are some extremely large goal amounts that are larger than 1 million dollars and 51 out of 55 are from US campaigns. In addition, we find there are some extremely large numbers of backers that are more than 20k, and 10 out of 11 are from US campaigns. All these indicate a sub-analysis of the US is necessary to better understand what makes a successful campaign in the US.

3.3 Results and Analysis

Firstly, we applied logistic regression and random forest models on the cleaned dataset. The outcome is whether the campaign is successful or not. We split the dataset where 70% of it is used as a training set and 30% as a test set. We compared the accuracy in the test set of those two models in Table 2.

	class 0	class 1	overall
logistic regression	0.8	0.36	0.57
random forest	0.71	0.80	0.78

Table 2: Model comparison in terms of accuracy

The random forest outperforms compared to the logistic regression. The logistic regression model performs worse when predicting the chance of successful campaigns. This may be caused by the non-linear relationship in our dataset and by a large number of dummy variables. We then looked deeper into the significant coefficients estimated by the logistic regression model as shown in Table 3.

Variable	exp(coef)	Confidence Interval
the goal amount	0.6835	(0.5770, 0.8097)
the cleaned length of campaign's name	1.2483	(1.2060, 1.2921)
the cleaned length of campaign's blurb	1.0669	(1.0313, 1.1038)
the year of the campaign's deadline	0.1833	(0.0998, 0.3368)
the hour of the campaign's state changing	1.1038	(1.0546, 1.1554)
the hour of the campaign's launching	0.8457	(0.8070, 0.8864)
US or other country	1.0909	(1.0530, 1.1303)

Table 3: Estimated coefficients of logistic regression

Obviously, we can find the country of campaigns is significant and the coefficient indicates that the campaigns from the US have 9.09% more odds of success than campaigns from other countries. Similarly, we can say that if the length of campaign's name increases per unit, there will be 24.83% more odds of success. Then, we conducted the logistic regression only on the campaigns from the US. The significant variables do not change.

Variable	exp(coef)	Confidence Interval
the goal amount	0.7068	(0.5770, 0.8782)
the cleaned length of campaign's name	1.2406	(1.1897, 1.2936)
the cleaned length of campaign's blurb	1.0647	(1.0215, 1.1097)
the year of the campaign's deadline	0.1849	(0.0773, 0.4424)
the hour of the campaign's state changing	1.0668	(1.0121, 1.1245)
the hour of the campaign's launching	0.8703	(0.8241, 0.9190)

Table 4: Estimated coefficients of logistic regression on the US campaigns

The coefficients do not change much either. We can say that if the length of campaign's name increases per unit, there will be 24.06% more odds of success. If the goal amount increases per unit, there will be 29.32% more adds of failure.

If our clients would like to raise at least 25,000 dollars (as goal) with at least 1000 backers, the probability of success is very highly that is 95.32% based on the campaign data in the US. Moreover, we find that campaigns with more backers tend to have a successful state. However, we cannot control the number of backers as what we expect.

4 Conclusion

In our analysis, we utilized the logistic regression and random forest models to determine the factors that contribute to a successful campaign and predict whether a campaign will succeed or not. We first find that the campaign's country (US or not) matters considering whether it is more likely to succeed. Focusing on the US data, we find if the campaign's goal amount increases per unit, there will be 29.32% more adds of failure. Then, we calculated the probability of success that is 95.32 % if our clients raises more than 25,000 dollars with more than 1000 backers.

We dropped rows that did not provide much information. However, we also found some rows corresponded to the same project but they sought funds in different countries, such as id=2416 and id=3352. We could look deeper into those projects and figure out what made them successful and fail before. We also dropped the column of campaign's category as about 1/3 rows were null. However, this feature could be important determining whether this campaign will be successful or not. If time permitted, we could go through the blurb and profile of the campaign to impute its category.

1 Load Data

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('https://query.data.world/s/lxnrwj5w73bsigranne42td54f54sm', index_col=0)
```

```
/var/folders/qx/wlpqnbz178962z833f_yq5gh0000gn/T/ipykernel_61643/1769345709.py:1: DtypeWarning: Columns (29,30,31,32) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('https://query.data.world/s/lxnrwj5w73bsigranne42td54f54sm', index_col=0)
```

In [3]:
```
df.head()
```

Out[3]:

	id	photo	name	blurb	goal	pledged	state	slug	disable_communication	country	...	launch_to_dea
0	1454391034	{"small":"https://ksr-ugc.imgix.net/assets/011...	Auntie Di's Music Time Sign ASL for Hearing an...	MTS ASL Curriculum Workbook is a reproducible ...	1500.0	0.0	failed	auntie-dis-music-time-sign-asl-for-hearing-and...	False	US	...	36 20:47:24.00000
1	1655206086	{"small":"https://ksr-ugc.imgix.net/assets/012...	Jump Start Kindergarten Toolkit	This kit teaches how to print, correct an ugly...	500.0	0.0	failed	jump-start-kindergarten-toolkit	False	US	...	60 00:00:00.00000
2	311581827	{"small":"https://ksr-ugc.imgix.net/assets/012...	Ojukwu Balewa Awolowo (O.B.A.) Public Library ...	Establishing a free, world-class, public libra...	100000.0	120.0	failed	ojukwu-balewa-awolowo-oba-public-library-of-ni...	False	US	...	60 00:00:00.00000
3	859724515	{"small":"https://ksr-ugc.imgix.net/assets/011...	MASTIZE - [mas-TAHYZ, MAS-tahyz] - to spread	Goal: Introducing a new word into the English ...	5000.0	0.0	failed	mastize-mas-tahyz-mas-tahyz-to-spread	False	US	...	30 00:00:00.00000
4	1613604977	{"small":"https://ksr-ugc.imgix.net/assets/012...	Synopse der EU-DSGVO - Artikel, Erwägungsgründ...	Zu den Artikeln der DSGVO sind die korrespondi...	3222.0	356.0	failed	synopse-der-eu-dsgvo-artikel-erwagungsgrunde-bdsg	False	DE	...	32 06:02:33.00000

5 rows × 67 columns

In [4]:
```
df.shape
```

Out[4]: (20632, 67)

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'photo', 'name', 'blurb', 'goal', 'pledged', 'state', 'slug',
       'disable_communication', 'country', 'currency', 'currency_symbol',
       'currency_trailing_code', 'deadline', 'state_changed_at', 'created_at',
       'launched_at', 'staff_pick', 'backers_count', 'static_usd_rate',
       'usd_pledged', 'creator', 'location', 'category', 'profile',
       'spotlight', 'urls', 'source_url', 'friends', 'is_starred',
       'is_backing', 'permissions', 'name_len', 'name_len_clean', 'blurb_len',
       'blurb_len_clean', 'deadline_weekday', 'state_changed_at_weekday',
       'created_at_weekday', 'launched_at_weekday', 'deadline_month',
       'deadline_day', 'deadline_yr', 'deadline_hr', 'state_changed_at_month',
       'state_changed_at_day', 'state_changed_at_yr', 'state_changed_at_hr',
       'created_at_month', 'created_at_day', 'created_at_yr', 'created_at_hr',
       'launched_at_month', 'launched_at_day', 'launched_at_yr',
       'launched_at_hr', 'create_to_launch', 'launch_to_deadline',
       'launch_to_state_change', 'create_to_launch_days',
       'launch_to_deadline_days', 'launch_to_state_change_days',
       'SuccessfulBool', 'USorGB', 'TOPCOUNTRY', 'LaunchedTuesday',
       'DeadlineWeekend'],
      dtype='object')
```

In [141]:
```
df[df['country']=='US'].shape
```

Out[141]: (14141, 67)

2 Clean Data

In [6]:
```
len(df[df.duplicated(subset='id')]) # no duplicate id
```
Out[6]: 0

2.1 Remove rows that do not provide much information

We can see some campaigns in the dataset were just used for testing and some were named with "Canceled" mark and no other infomation in `blurb` . Those rows cannot provide any meaningful information to analyze what makes a successful campaign. So, we can remove those rows.

In [7]:
```
len(df['name'].unique())
```
Out[7]: 20611

In [8]:

```
# find duplicate campaigns in names
columns  = ['name', 'country', 'name_len_clean', 'blurb_len_clean', 'state', 'SuccessfulBool', 'goal', 'blurb',
            'usd_pledged', 'backers_count', 'category', 'deadline', 'state_changed_at', 'created_at', 'launched_at']
(df[df.duplicated('name')].sort_values(by=['name']))[columns]
```

Out[8]:

	name	country	name_len_clean	blurb_len_clean	state	SuccessfulBool	goal	blurb	usd_pledged	backers_count	categor
2416	A Midsummer Night's Dream	US	4.0	14.0	failed	0	2500.0	This production is being put together by Wilso...	504.000000	10	Play
3352	A Midsummer Night's Dream	GB	4.0	12.0	successful	1	2000.0	Join us as we lead you into our enchanted worl...	4806.095430	84	Immersiv
1600	BEIRUT, LADY OF LEBANON	US	4.0	10.0	failed	0	30000.0	A Theatrical Production Celebrating the Lebane...	1225.000000	7	Play
2714	Beauty and the Beast	US	2.0	15.0	successful	1	1000.0	The 7th & 8th grade burgeoning actors, singers...	1001.000000	29	Music
3736	Born in Burnley, Made in Edinburgh	GB	4.0	14.0	successful	1	1000.0	Support a group of enthusiastic and talented y...	1688.512099	42	Festiva
11548	Cancelled. (Canceled)	US	2.0	4.0	canceled	0	250000.0	This project has been cancelled until further ...	25.000000	2	Softwar
8110	Christian DiLusso Watches	SE	3.0	4.0	failed	0	100000.0	It's time to move forward.	87.141577	1	Wearable
19146	FREE ENERGY	US	2.0	13.0	failed	0	100.0	THERE WILL BE FREE ENERGY! THE OVER-UNITY C...	33.000000	5	Na
20191	Fitness Buddy	GB	2.0	15.0	failed	0	10000.0	To create the world's first app that allows fi...	1077.553356	9	App
1789	Gruesome Playground Injuries	US	3.0	13.0	successful	1	5000.0	LA-based team of professional actors and direc...	5260.920000	82	Play
16746	Infinity	GB	1.0	13.0	failed	0	50000.0	Infinity consists of a personal eye view camer...	0.000000	0	Gadget
2114	Macbeth	US	1.0	14.0	successful	1	5500.0	Old Hat's new production explores the bleak cu...	5516.000000	79	Play
6744	N/A (Canceled)	AU	NaN	NaN	canceled	0	30000.0	NaN	454.353608	6	We
9240	Online Free Sound Effects/store (Canceled)	GB	5.0	10.0	canceled	0	50000.0	Making a online website that sells free sound ...	0.000000	0	Soun
17887	Project Canceled (Canceled)	US	3.0	1.0	canceled	0	80000.0	Canceled	1267.000000	12	Gadget
3411	Romeo & Juliet	GB	3.0	11.0	failed	0	5000.0	Set under the train tracks of a parallel Londo...	0.000000	0	Immersiv
7230	SoulDraft.com: Making Art Accessible to Everyo...	FR	6.0	14.0	canceled	0	50000.0	Soul Draft is a blog that puts forward artists...	1139.681118	2	We
2728	The 25th Annual Putnam County Spelling Bee	CA	7.0	14.0	successful	1	1210.0	A Tony and Drama Desk award-winning musical ki...	968.508787	41	Music
4251	Us, Bent (Canceled)	US	3.0	4.0	canceled	0	4000.0	Check us out! www.usbentshowcase.com	0.000000	0	Experiment
8722	test (Canceled)	US	2.0	1.0	canceled	0	49000.0	test	1333.000000	10	Na
16541	weSTAND: A Stand With a Mission	US	5.0	11.0	failed	0	7000.0	Following our campaign, a portion of each sale...	530.000000	19	Gadget

In [9]:
```
df[df['name'].str.contains('Canceled | canceled')]
```

Out[9]:

	id	photo	name	blurb	goal	pledged	state	slug	disable_communication	country	...	launch_to_deadline	l
11557	2135481078	{"small":"https://ksr-ugc.imgix.net/assets/011...	Canceled (Canceled)	Cancelled	2000.0	0.0	canceled	learn-2-develop-applications	False	US	...	51 days 08:28:13.000000000	
17867	1313241934	{"small":"https://ksr-ugc.imgix.net/assets/012...	Project Canceled (Canceled)	Project Canceled	10000.0	794.0	canceled	litelife-the-keychain-sized-mobile-charger	False	US	...	30 days 00:00:00.000000000	
17887	1415497849	{"small":"https://ksr-ugc.imgix.net/assets/012...	Project Canceled (Canceled)	Canceled	80000.0	1267.0	canceled	esl-and-brush-wireless-charging-solar-power-an...	False	US	...	45 days 14:04:49.000000000	

3 rows × 67 columns

In [10]:
```
(df[df['name'].str.contains('Canceled | canceled') &
    df['blurb'].str.contains('Canceled | canceled | cancelled')])[columns]
```

Out[10]:

name	country	name_len_clean	blurb_len_clean	state	SuccessfulBool	goal	blurb	usd_pledged	backers_count	category	deadline	state_changed_at	created_at	la

In [11]:
```
# id=6744 NA
# id=11557, 17867, 17887, 11548, 11848
# id=8722 test
df_transformed = df.drop([6774, 8722, 11557, 17867, 17887, 11548, 11848], axis=0)
```

In [12]:
```
df_transformed.shape
```

Out[12]: (20625, 67)

2.2 Drop null columns

In [13]:
```
df_transformed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20625 entries, 0 to 20631
Data columns (total 67 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           20625 non-null  int64
 1   photo                        20625 non-null  object
 2   name                         20625 non-null  object
 3   blurb                        20620 non-null  object
 4   goal                         20625 non-null  float64
 5   pledged                      20625 non-null  float64
 6   state                        20625 non-null  object
 7   slug                         20625 non-null  object
 8   disable_communication        20625 non-null  bool
 9   country                      20625 non-null  object
 10  currency                     20625 non-null  object
 11  currency_symbol              20625 non-null  object
 12  currency_trailing_code       20625 non-null  bool
 13  deadline                     20625 non-null  object
 14  state_changed_at             20625 non-null  object
 15  created_at                   20625 non-null  object
 16  launched_at                  20625 non-null  object
 17  staff_pick                   20625 non-null  bool
 18  backers_count                20625 non-null  int64
 19  static_usd_rate              20625 non-null  float64
 20  usd_pledged                  20625 non-null  float64
 21  creator                      20625 non-null  object
 22  location                     20581 non-null  object
 23  category                     18737 non-null  object
 24  profile                      20625 non-null  object
 25  spotlight                    20625 non-null  bool
 26  urls                         20625 non-null  object
 27  source_url                   20625 non-null  object
 28  friends                      60 non-null     object
 29  is_starred                   60 non-null     object
 30  is_backing                   60 non-null     object
 31  permissions                  60 non-null     object
 32  name_len                     20620 non-null  float64
 33  name_len_clean               20620 non-null  float64
 34  blurb_len                    20620 non-null  float64
 35  blurb_len_clean              20620 non-null  float64
 36  deadline_weekday             20625 non-null  object
 37  state_changed_at_weekday     20625 non-null  object
 38  created_at_weekday           20625 non-null  object
 39  launched_at_weekday          20625 non-null  object
 40  deadline_month               20625 non-null  int64
 41  deadline_day                 20625 non-null  int64
 42  deadline_yr                  20625 non-null  int64
 43  deadline_hr                  20625 non-null  int64
 44  state_changed_at_month       20625 non-null  int64
 45  state_changed_at_day         20625 non-null  int64
 46  state_changed_at_yr          20625 non-null  int64
 47  state_changed_at_hr          20625 non-null  int64
 48  created_at_month             20625 non-null  int64
 49  created_at_day               20625 non-null  int64
 50  created_at_yr                20625 non-null  int64
 51  created_at_hr                20625 non-null  int64
 52  launched_at_month            20625 non-null  int64
 53  launched_at_day              20625 non-null  int64
 54  launched_at_yr               20625 non-null  int64
 55  launched_at_hr               20625 non-null  int64
 56  create_to_launch             20625 non-null  object
 57  launch_to_deadline           20625 non-null  object
 58  launch_to_state_change       20625 non-null  object
 59  create_to_launch_days        20625 non-null  int64
 60  launch_to_deadline_days      20625 non-null  int64
 61  launch_to_state_change_days  20625 non-null  int64
 62  SuccessfulBool               20625 non-null  int64
 63  USorGB                       20625 non-null  int64
 64  TOPCOUNTRY                   20625 non-null  int64
 65  LaunchedTuesday              20625 non-null  int64
 66  DeadlineWeekend              20625 non-null  int64
dtypes: bool(4), float64(8), int64(26), object(29)
memory usage: 10.1+ MB
```

In [14]:
```
# drop columns that are mostly null
df_transformed.drop(['friends', 'is_starred', 'is_backing', 'permissions'], axis=1, inplace=True)
```

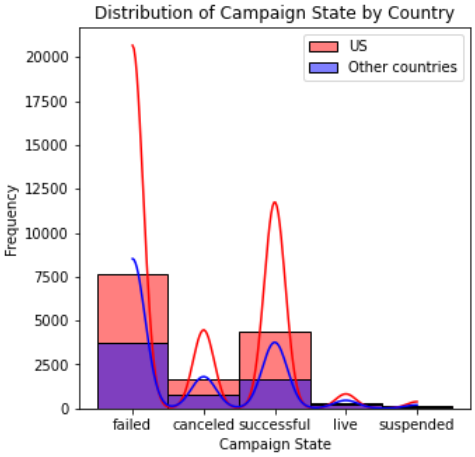
3 EDA

We focus on the difference between US campaigns and non-US campaigns, and also what makes a US campaign successful.

3.1 Variable distribution

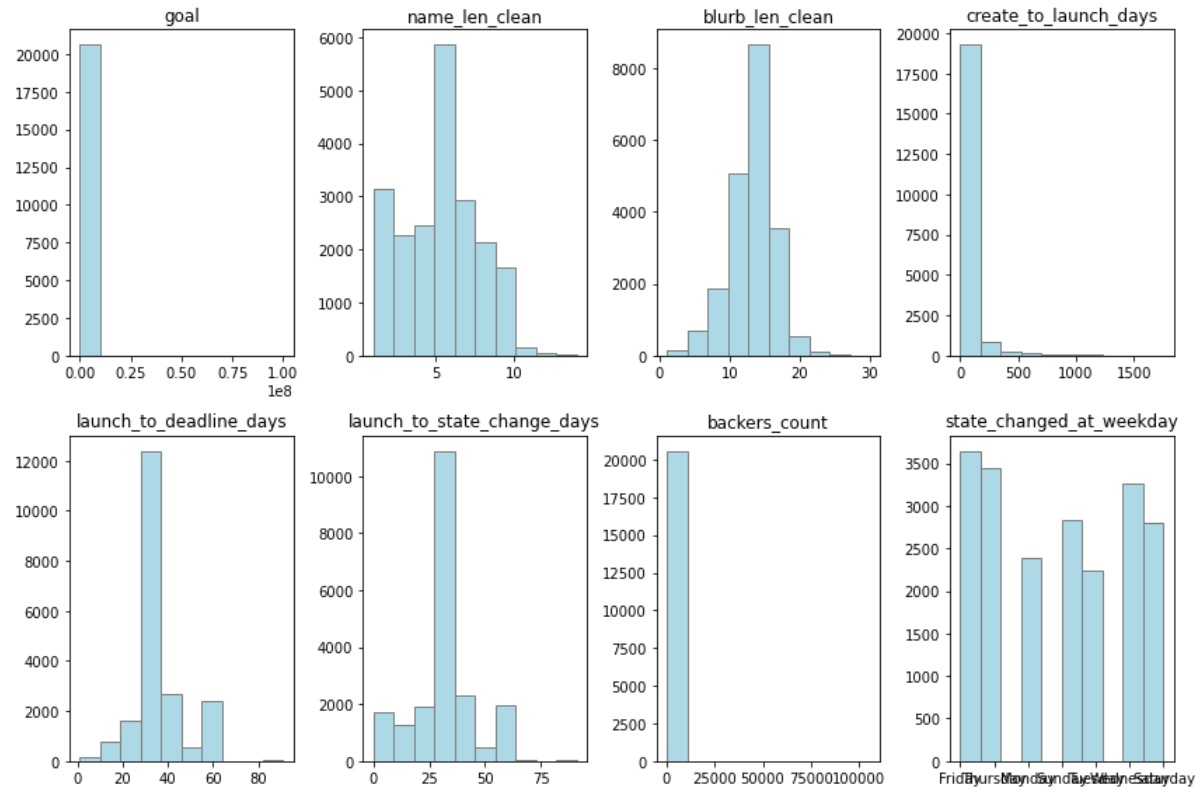
In [15]:
```
# dummy variable whether is US or not
df_transformed['USorNot'] = np.where(df_transformed['country'] == 'US', 1, 0)
```

In [16]:
```
import seaborn as sns
plt.rcParams['figure.figsize'] = [5, 5]
ax = sns.histplot(df_transformed[df_transformed.USorNot==1]['state'], label='US', kde=True, color='red')
sns.histplot(df_transformed[df_transformed.USorNot==0]['state'], label='Other countries', kde=True, color='blue')
ax.set(xlabel='Campaign State', ylabel='Frequency')
plt.title('Distribution of Campaign State by Country')
plt.legend()
plt.show()
```



3.2 Histograms

In [17]:
```
plt.rcParams['figure.figsize'] = [12, 8]
def draw_histogram(df, variables, n_rows, n_cols):
    fig = plt.figure()
    for i, var_name in enumerate(variables):
        ax = fig.add_subplot(n_rows,n_cols,i+1)
        df[var_name].hist(ax=ax, color='lightblue', ec='grey')
        ax.set_title(var_name)
        ax.grid(False)
    fig.tight_layout()
    plt.show()

cols = ['goal', 'name_len_clean', 'blurb_len_clean', 'create_to_launch_days',
        'launch_to_deadline_days', 'launch_to_state_change_days', 'backers_count', 'state_changed_at_weekday']
draw_histogram(df_transformed[cols], cols, 2, 4)
```



From first plot, we find there can be outliers where usd_pledged is too large.

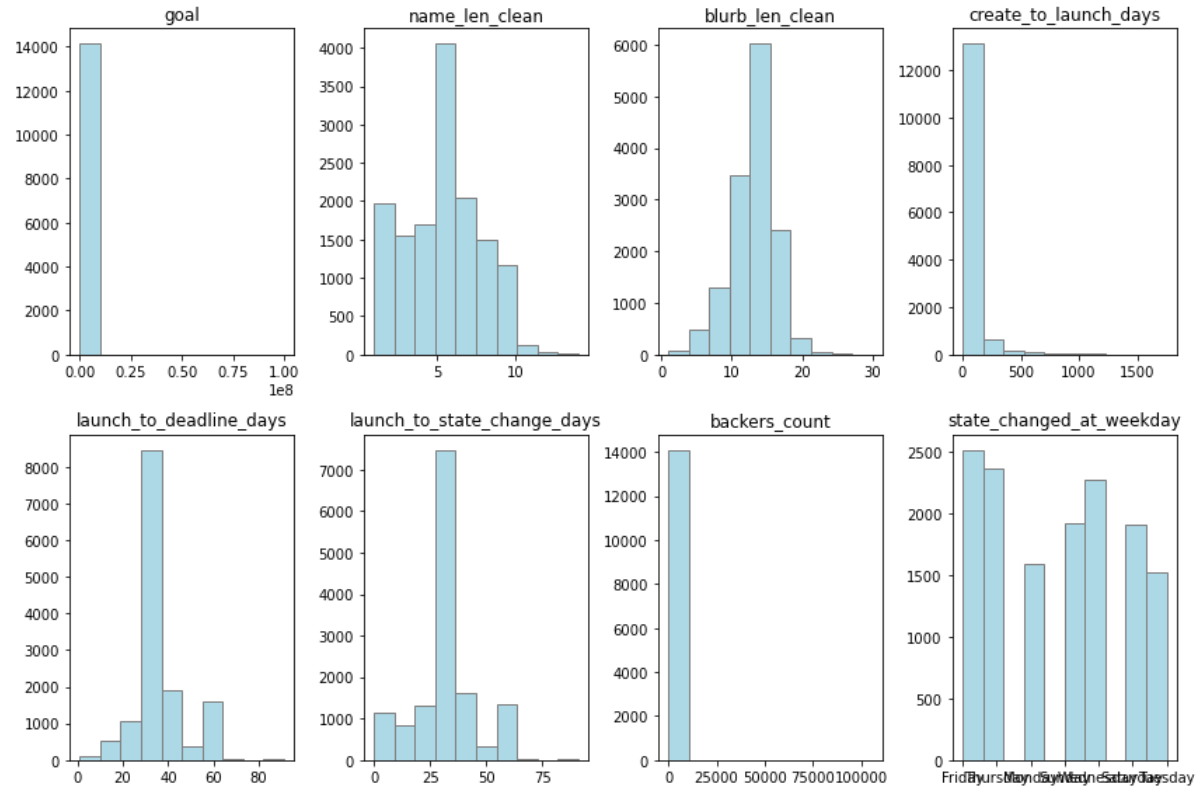
In [18]:
```
len(df_transformed[df_transformed['usd_pledged'] > 1000000])
```

Out[18]: 55

In [19]:
```
len(df_transformed[df_transformed['backers_count'] > 20000])
```

Out[19]: 11

In [20]:
```
# us only
draw_histogram((df_transformed[df_transformed['country']=='US'])[cols], cols, 2, 4)
```



In [21]:
```
len(df_transformed[(df_transformed['usd_pledged'] > 1000000) & (df_transformed['country']=='US')])
```

Out[21]: 51

In [22]:
```
len(df_transformed[(df_transformed['backers_count'] > 20000) & (df_transformed['country']=='US')])
```

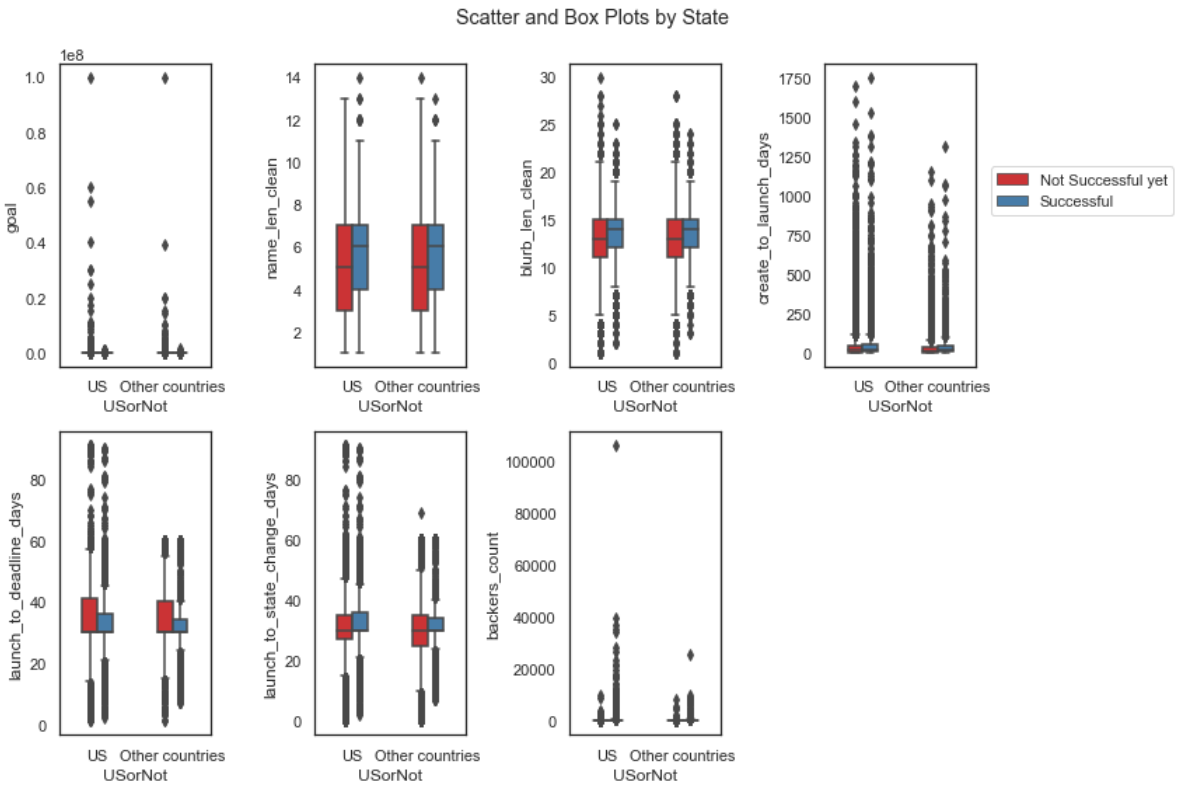
Out[22]: 10

3.3 Scatter and box plots

We first look at the varibles that of interest.

In [23]:
```
df1 = df_transformed.copy()
df1['SuccessfulBool'] = df1['SuccessfulBool'].map({0: 'Not Successful yet', 1: 'Successful'})
df1['USorNot'] = df1['USorNot'].map({0: 'Other countries', 1: 'US'})
```

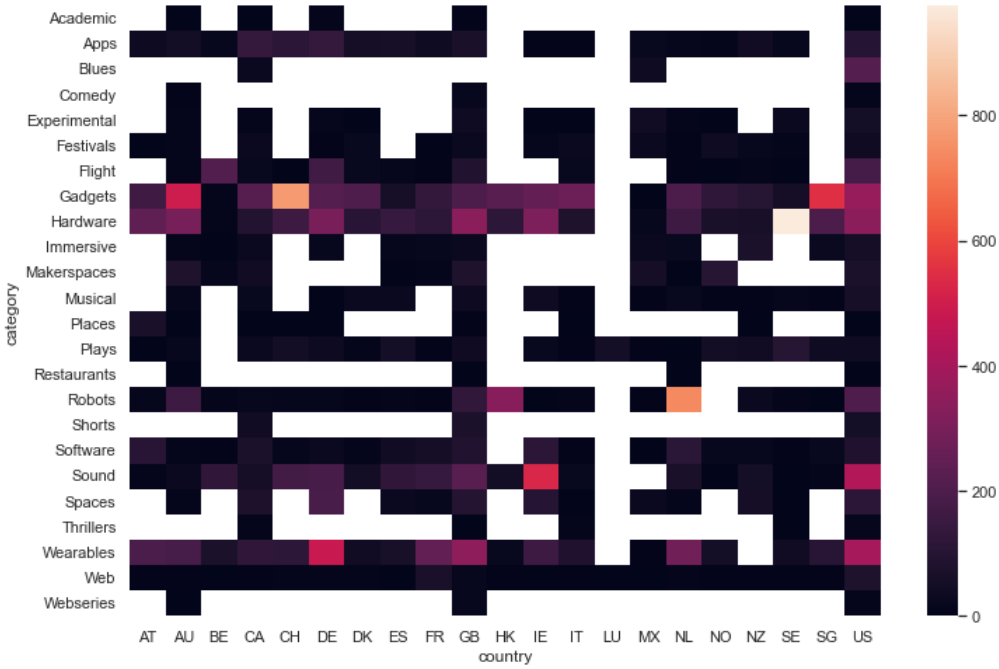
In [24]:
```
def draw_boxplot_bygroup(df, outcome, n_rows, n_cols, group):
    fig = plt.figure()
    variables = df.columns.drop([outcome, group])
    sns.set(style='white', palette='Set1')
    for i, var_name in enumerate(variables):
        ax = fig.add_subplot(n_rows, n_cols, i+1)
        if var_name == 'create_to_launch_days':
            sns.boxplot(x=outcome, y=var_name, hue=group, data=df, width=0.4, ax=ax)
            ax.legend(loc=(1.1, 0.5))
        # elif var_name == 'USorNot':
            # sns.boxplot(x=outcome, y=var_name, data=df, width=0.4, ax=ax)
        else:
            sns.boxplot(x=outcome, y=var_name, hue=group, data=df, width=0.4, ax=ax)
            ax.legend_.remove()
    fig.suptitle('Scatter and Box Plots by State')
    fig.tight_layout()
    plt.show()

cols = ['goal', 'name_len_clean', 'blurb_len_clean', 'create_to_launch_days',
        'launch_to_deadline_days', 'launch_to_state_change_days', 'backers_count', 'SuccessfulBool', 'USorNot']
draw_boxplot_bygroup(df1[cols], 'USorNot', 2, 4, 'SuccessfulBool')
```



3.4 Heatmap

In [25]:
```
# heat map of average backers by country and main_category
pivot_table = df_transformed.pivot_table(index='category', columns='country', values='backers_count', aggfunc='mean', )
sns.heatmap(pivot_table)
```

Out[25]: <AxesSubplot:xlabel='country', ylabel='category'>



Countries other than US have lots of missing value.

3.5 Varible Selection

3.5.1 Drop columns that are not useful

- id
- Natural language processing is beyond the scope of this project. So we also remove columns that contain urls and json.
- correlated varibles: pledged, usd_pledged, currency, static_usd_rate, currency_symbol, currency_trailing_code. We only obtain `used_pledged` for EDA and remove others since all currency is converted to dollars.
- datetime columns: deadline, state_changed_at, created_at and launched_at are stored in unix time. Also for create_to_launch, launch_to_deadline, launch_to_state_change.
- Our analysis focuses on the US and other countries. So we can remove country, USorGB and create a dummy variable indicating it is US or not.
- `spotlight` : projects can only be highlighted after they are already successful. This is entirely correlated to successful campaigns.
- name_len and name_len_clean, blurb_len and blurb_len_clean are correlated. We leave the cleaned one.

In [26]:
```
cols_drop = ['id', 'photo', 'name', 'blurb', 'pledged', 'slug', 'currency', 'currency_symbol',
       'currency_trailing_code', 'deadline', 'state_changed_at', 'created_at',
       'launched_at', 'static_usd_rate', 'creator', 'location','profile',
       'spotlight', 'urls', 'source_url', 'name_len', 'blurb_len',
       'create_to_launch', 'launch_to_deadline', 'launch_to_state_change', 'USorGB', 'country', 'state']
df_transformed.drop(cols_drop, axis=1, inplace=True)
```

3.5.2 Remove correlated variables

In [27]:
```
# pariwise correlation
df_transformed.corr()
```

```
/var/folders/qx/wlpqnbz178962z833f_yq5gh0000gn/T/ipykernel_61643/2866852891.py:2: FutureWarning: The default value of numer
ic_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  df_transformed.corr()
```

Out[27]:

	goal	disable_communication	staff_pick	backers_count	usd_pledged	name_len_clean	blurb_len_clean	deadline_month	deadline_day
goal	1.000000	-0.003438	-0.010754	0.003837	0.009605	-0.014570	-0.008108	0.000600	-0.015332
disable_communication	-0.003438	1.000000	-0.036555	0.004582	-0.005347	0.032947	-0.007948	-0.000651	0.015122
staff_pick	-0.010754	-0.036555	1.000000	0.201952	0.267046	0.076695	0.018859	0.036313	-0.004432
backers_count	0.003837	0.004582	0.201952	1.000000	0.757999	0.061755	0.008511	0.006400	-0.009522
usd_pledged	0.009605	-0.005347	0.267046	0.757999	1.000000	0.090190	0.014214	0.005653	-0.009594
name_len_clean	-0.014570	0.032947	0.076695	0.061755	0.090190	1.000000	0.215821	0.018115	0.000168
blurb_len_clean	-0.008108	-0.007948	0.018859	0.008511	0.014214	0.215821	1.000000	0.000427	0.006716
deadline_month	0.000600	-0.000651	0.036313	0.006400	0.005653	0.018115	0.000427	1.000000	0.013810
deadline_day	-0.015332	0.015122	-0.004432	-0.009522	-0.009594	0.000168	0.006716	0.013810	1.000000
deadline_yr	0.006820	0.029245	-0.121580	-0.021360	-0.009376	0.013044	0.013759	-0.252904	-0.019461
deadline_hr	0.001145	-0.007547	-0.010892	-0.025266	-0.037357	-0.018665	-0.010194	-0.017411	0.029209
state_changed_at_month	0.002375	-0.001546	0.036884	0.006225	0.005641	0.016515	-0.000566	0.957330	0.015372
state_changed_at_day	-0.011216	-0.004376	-0.002802	-0.008927	-0.009135	-0.001090	0.000487	0.016478	0.874029
state_changed_at_yr	0.005983	0.025211	-0.120282	-0.020601	-0.008432	0.012579	0.014725	-0.243799	-0.017548
state_changed_at_hr	-0.001998	0.007999	-0.005153	-0.022921	-0.034411	-0.022093	-0.008538	-0.015264	0.027201
created_at_month	0.002297	0.010492	0.013079	0.003861	-0.000194	0.016620	0.004946	0.256255	-0.005813
created_at_day	-0.000069	-0.010295	0.002628	0.002436	0.000288	0.006704	-0.005301	0.003688	0.093800
created_at_yr	0.006350	0.029481	-0.127629	-0.029302	-0.018132	-0.006908	0.012491	-0.086751	0.001155
created_at_hr	0.002119	-0.013244	-0.007246	-0.004089	-0.013473	-0.007686	-0.011017	-0.023952	-0.001663
launched_at_month	0.002633	-0.002540	0.024115	0.010491	0.006815	0.028031	0.000519	0.548679	0.006630
launched_at_day	0.000478	0.014777	0.005957	0.007561	0.001467	0.000308	0.007562	0.023875	0.463009
launched_at_yr	0.004970	0.029633	-0.119307	-0.023283	-0.010914	0.010981	0.013854	-0.150869	-0.007682
launched_at_hr	0.005361	0.005559	-0.061801	-0.049281	-0.066690	-0.045270	-0.020092	-0.024179	0.003692
create_to_launch_days	-0.005612	-0.010426	0.050072	0.032603	0.037047	0.084547	0.002155	0.012924	0.003962
launch_to_deadline_days	0.044366	0.009654	-0.020380	0.021233	0.039301	-0.001413	-0.002052	-0.034185	0.018068
launch_to_state_change_days	0.024841	-0.146464	0.042729	0.043725	0.064925	-0.029205	0.018281	0.049098	0.008773
SuccessfulBool	-0.035053	-0.068163	0.344295	0.194105	0.232220	0.132860	0.060052	0.025638	-0.011326
TOPCOUNTRY	-0.018735	-0.015168	0.059635	0.031375	0.037515	0.028315	-0.007517	0.000975	-0.013015
LaunchedTuesday	0.001786	0.009104	0.040073	0.028455	0.045573	0.026374	-0.003547	0.019026	0.007464
DeadlineWeekend	-0.007185	0.003882	-0.031936	-0.006848	-0.011135	-0.021386	-0.002076	-0.019732	0.020517
USorNot	-0.006025	-0.015549	0.055449	0.039818	0.051540	0.041679	-0.029306	0.003411	-0.006709

31 rows × 31 columns

We can see backers_count and usd_pledged are highly positive correlated.
launch_to_state_change_days and launch_to_deadline_days are highly positive correlated.
TOPCOUNTRY and USorNot highly positive correlated.
We may remove launch_to_deadline_days and TOPCOUNTRY to eliminate multicollinearity.

In [28]:
```
df_transformed.drop(['TOPCOUNTRY', 'launch_to_deadline_days'], axis=1, inplace=True)
df_transformed.shape
```

Out[28]: (20625, 34)

In [29]:
```
# get categorical variable
cat_vars = df_transformed.select_dtypes(exclude=np.number)
cat_vars.drop('category', axis=1, inplace=True)
df_num = df_transformed.copy()
dic = {'Monday':1, 'Tuesday':2, 'Wednesday':3, 'Thursday':4, 'Friday':5, 'Saturday':6, 'Sunday':7}
for var in cat_vars:
    df_num[var] = df_transformed[var].map(dic)
df_num.corr().round(4)
```

```
/var/folders/qx/wlpqnbz178962z833f_yq5gh0000gn/T/ipykernel_61643/3231429754.py:8: FutureWarning: The default value of nume
ric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or spec
ify the value of numeric_only to silence this warning.
  df_num.corr().round(4)
```

Out[29]:

	goal	disable_communication	staff_pick	backers_count	usd_pledged	name_len_clean	blurb_len_clean	deadline_weekday	state_chang
goal	1.0000	NaN	NaN	0.0038	0.0096	-0.0146	-0.0081	-0.0142	
disable_communication	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
staff_pick	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
backers_count	0.0038	NaN	NaN	1.0000	0.7580	0.0618	0.0085	0.0019	
usd_pledged	0.0096	NaN	NaN	0.7580	1.0000	0.0902	0.0142	0.0005	
name_len_clean	-0.0146	NaN	NaN	0.0618	0.0902	1.0000	0.2158	-0.0219	
blurb_len_clean	-0.0081	NaN	NaN	0.0085	0.0142	0.2158	1.0000	-0.0066	

positively highly correlated:

- deadline_weekday and state_changed_at_weekday
- DeadlineWeekend and deadline_weekday
- DeadlineWeekend and state_changed_at_weekday
- launched_at_hr and deadline_hr
- deadline_month and launched_at_month

In [30]:
```
df_transformed.drop(['deadline_weekday', 'DeadlineWeekend', 'deadline_hr', 'deadline_month'], axis=1, inplace=True)
```

In [31]:
```
# Checking the proportions of each category
df_transformed.disable_communication.value_counts(normalize=True)
```

Out[31]:
```
False    0.988848
True     0.011152
Name: disable_communication, dtype: float64
```

98.9% of project owners did not disable communication with their backers (unsurprisingly). Because nearly all projects have the same value for this variable, it will be dropped as it does not provide much information.

In [32]:
```
df_transformed.drop('disable_communication', axis=1, inplace=True)
```

3.5.3 Remove columns that have further information

Our goal is to find what factors make a successful campaign and to predict whether or not a Kickstarter campaign will be successful or fail. We may leave those variables for EDA, but need to remove them before buliding our models.

- usd_pledged: This is the amount of money that was funded after.
- backers_count: the number of people who backed the campaign. This column contains "future information" and could act as a proxy for our target variable.

In [33]:
```
df_transformed.isna().sum()
```

Out[33]:
```
goal                              0
staff_pick                        0
backers_count                     0
usd_pledged                       0
category                       1888
name_len_clean                    5
blurb_len_clean                   5
state_changed_at_weekday          0
created_at_weekday                0
launched_at_weekday               0
deadline_day                      0
deadline_yr                       0
state_changed_at_month            0
state_changed_at_day              0
state_changed_at_yr               0
state_changed_at_hr               0
created_at_month                  0
created_at_day                    0
created_at_yr                     0
created_at_hr                     0
launched_at_month                 0
launched_at_day                   0
launched_at_yr                    0
launched_at_hr                    0
create_to_launch_days             0
launch_to_state_change_days       0
SuccessfulBool                    0
LaunchedTuesday                   0
USorNot                           0
dtype: int64
```

In [34]:
```
# replace null values
df_transformed.name_len_clean.fillna(0, inplace=True)
df_transformed.blurb_len_clean.fillna(0, inplace=True)
```

In [35]:
```
df_transformed.drop(['usd_pledged', 'backers_count', 'category'], axis=1, inplace=True)
```

In [36]:
```
df_transformed.isna().sum()
```

Out[36]:
```
goal                          0
staff_pick                    0
name_len_clean                0
blurb_len_clean               0
state_changed_at_weekday      0
created_at_weekday            0
launched_at_weekday           0
deadline_day                  0
deadline_yr                   0
state_changed_at_month        0
state_changed_at_day          0
state_changed_at_yr           0
state_changed_at_hr           0
created_at_month              0
created_at_day                0
created_at_yr                 0
created_at_hr                 0
launched_at_month             0
launched_at_day               0
launched_at_yr                0
launched_at_hr                0
create_to_launch_days         0
launch_to_state_change_days   0
SuccessfulBool                0
LaunchedTuesday               0
USorNot                       0
dtype: int64
```

3.6 Baseline Table

In [147]:
```
df.describe().round(2).transpose()
```

Out[147]:

	count	mean	std	min	25%	50%	75%	max
id	20632.0	1.071156e+09	6.154929e+08	164555.00	5.472185e+08	1.069882e+09	1.601801e+09	2.147388e+09
goal	20632.0	9.410497e+04	1.335511e+06	1.00	4.000000e+03	1.400000e+04	5.000000e+04	1.000000e+08
pledged	20632.0	2.139268e+04	1.204972e+05	0.00	2.500000e+01	6.950000e+02	5.954250e+03	6.225355e+06
backers_count	20632.0	1.836800e+02	1.222010e+03	0.00	2.000000e+00	1.200000e+01	6.300000e+01	1.058570e+05
static_usd_rate	20632.0	1.040000e+00	2.300000e-01	0.05	1.000000e+00	1.000000e+00	1.000000e+00	1.720000e+00
usd_pledged	20632.0	2.091591e+04	1.154717e+05	0.00	2.500000e+01	7.163000e+02	6.004630e+03	6.225355e+06
name_len	20627.0	5.940000e+00	2.830000e+00	1.00	4.000000e+00	6.000000e+00	8.000000e+00	1.600000e+01
name_len_clean	20627.0	5.290000e+00	2.420000e+00	1.00	3.000000e+00	5.000000e+00	7.000000e+00	1.400000e+01
blurb_len	20627.0	1.899000e+01	4.630000e+00	1.00	1.700000e+01	2.000000e+01	2.200000e+01	3.500000e+01
blurb_len_clean	20627.0	1.308000e+01	3.280000e+00	1.00	1.100000e+01	1.300000e+01	1.500000e+01	3.000000e+01
deadline_month	20632.0	6.710000e+00	3.410000e+00	1.00	4.000000e+00	7.000000e+00	1.000000e+01	1.200000e+01
deadline_day	20632.0	1.570000e+01	9.030000e+00	1.00	8.000000e+00	1.500000e+01	2.300000e+01	3.100000e+01
deadline_yr	20632.0	2.014830e+03	1.270000e+00	2009.00	2.014000e+03	2.015000e+03	2.016000e+03	2.017000e+03
deadline_hr	20632.0	1.293000e+01	6.040000e+00	0.00	9.000000e+00	1.300000e+01	1.700000e+01	2.300000e+01
state_changed_at_month	20632.0	6.690000e+00	3.450000e+00	1.00	4.000000e+00	7.000000e+00	1.000000e+01	1.200000e+01
state_changed_at_day	20632.0	1.565000e+01	9.000000e+00	1.00	8.000000e+00	1.500000e+01	2.300000e+01	3.100000e+01
state_changed_at_yr	20632.0	2.014820e+03	1.270000e+00	2009.00	2.014000e+03	2.015000e+03	2.016000e+03	2.017000e+03
state_changed_at_hr	20632.0	1.286000e+01	6.020000e+00	0.00	9.000000e+00	1.300000e+01	1.700000e+01	2.300000e+01
created_at_month	20632.0	6.470000e+00	3.350000e+00	1.00	4.000000e+00	7.000000e+00	9.000000e+00	1.200000e+01
created_at_day	20632.0	1.554000e+01	8.780000e+00	1.00	8.000000e+00	1.500000e+01	2.300000e+01	3.100000e+01
created_at_yr	20632.0	2.014620e+03	1.270000e+00	2009.00	2.014000e+03	2.015000e+03	2.016000e+03	2.017000e+03
created_at_hr	20632.0	1.267000e+01	5.950000e+00	0.00	9.000000e+00	1.300000e+01	1.700000e+01	2.300000e+01
launched_at_month	20632.0	6.540000e+00	3.380000e+00	1.00	4.000000e+00	7.000000e+00	9.000000e+00	1.200000e+01
launched_at_day	20632.0	1.532000e+01	8.800000e+00	1.00	8.000000e+00	1.500000e+01	2.300000e+01	3.100000e+01
launched_at_yr	20632.0	2.014750e+03	1.260000e+00	2009.00	2.014000e+03	2.015000e+03	2.016000e+03	2.017000e+03
launched_at_hr	20632.0	1.242000e+01	5.570000e+00	0.00	9.000000e+00	1.200000e+01	1.600000e+01	2.300000e+01
create_to_launch_days	20632.0	4.958000e+01	1.110900e+02	0.00	3.000000e+00	1.400000e+01	4.500000e+01	1.754000e+03
launch_to_deadline_days	20632.0	3.472000e+01	1.187000e+01	1.00	3.000000e+01	3.000000e+01	4.000000e+01	9.100000e+01
launch_to_state_change_days	20632.0	3.117000e+01	1.428000e+01	0.00	2.800000e+01	3.000000e+01	3.500000e+01	9.100000e+01
SuccessfulBool	20632.0	2.900000e-01	4.500000e-01	0.00	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00
USorGB	20632.0	8.100000e-01	4.000000e-01	0.00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
TOPCOUNTRY	20632.0	8.200000e-01	3.900000e-01	0.00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
LaunchedTuesday	20632.0	2.300000e-01	4.200000e-01	0.00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
DeadlineWeekend	20632.0	2.900000e-01	4.500000e-01	0.00	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00

In [37]:
```
df_transformed.describe().round(2).transpose()
```

Out[37]:

	count	mean	std	min	25%	50%	75%	max
goal	20625.0	94117.39	1335736.97	1.0	4000.0	14000.0	50000.0	100000000.0
name_len_clean	20625.0	5.29	2.42	0.0	3.0	5.0	7.0	14.0
blurb_len_clean	20625.0	13.08	3.28	0.0	11.0	13.0	15.0	30.0
deadline_day	20625.0	15.70	9.03	1.0	8.0	15.0	23.0	31.0
deadline_yr	20625.0	2014.83	1.27	2009.0	2014.0	2015.0	2016.0	2017.0
state_changed_at_month	20625.0	6.68	3.45	1.0	4.0	7.0	10.0	12.0
state_changed_at_day	20625.0	15.65	9.00	1.0	8.0	15.0	23.0	31.0
state_changed_at_yr	20625.0	2014.83	1.27	2009.0	2014.0	2015.0	2016.0	2017.0
state_changed_at_hr	20625.0	12.85	6.02	0.0	9.0	13.0	17.0	23.0
created_at_month	20625.0	6.47	3.35	1.0	4.0	7.0	9.0	12.0
created_at_day	20625.0	15.54	8.78	1.0	8.0	15.0	23.0	31.0
created_at_yr	20625.0	2014.62	1.27	2009.0	2014.0	2015.0	2016.0	2017.0
created_at_hr	20625.0	12.67	5.95	0.0	9.0	13.0	17.0	23.0
launched_at_month	20625.0	6.54	3.38	1.0	4.0	7.0	9.0	12.0
launched_at_day	20625.0	15.32	8.80	1.0	8.0	15.0	23.0	31.0
launched_at_yr	20625.0	2014.75	1.26	2009.0	2014.0	2015.0	2016.0	2017.0
launched_at_hr	20625.0	12.42	5.57	0.0	9.0	12.0	16.0	23.0
create_to_launch_days	20625.0	49.59	111.11	0.0	3.0	14.0	45.0	1754.0
launch_to_state_change_days	20625.0	31.18	14.28	0.0	28.0	30.0	35.0	91.0
SuccessfulBool	20625.0	0.29	0.45	0.0	0.0	0.0	1.0	1.0
LaunchedTuesday	20625.0	0.23	0.42	0.0	0.0	0.0	0.0	1.0
USorNot	20625.0	0.69	0.46	0.0	0.0	1.0	1.0	1.0

In [38]:
```
def get_categorical_percentages(df):
    df_cat = df.select_dtypes(exclude=np.number)
    for var in df_cat.columns:
        perc = df[var].value_counts() / df[var].count()
        print(var)
        print(perc)

get_categorical_percentages(df_transformed)
```

```
staff_pick
False    0.894061
True     0.105939
Name: staff_pick, dtype: float64
state_changed_at_weekday
Friday       0.176970
Thursday     0.166642
Wednesday    0.158206
Sunday       0.137406
Saturday     0.136048
Monday       0.116218
Tuesday      0.108509
Name: state_changed_at_weekday, dtype: float64
created_at_weekday
Tuesday      0.175127
Monday       0.168242
Wednesday    0.161164
Thursday     0.152970
Friday       0.130812
Sunday       0.110352
Saturday     0.101333
Name: created_at_weekday, dtype: float64
launched_at_weekday
Tuesday      0.225067
Monday       0.203539
Wednesday    0.181333
Thursday     0.150012
Friday       0.136630
Saturday     0.052121
Sunday       0.051297
Name: launched_at_weekday, dtype: float64
```

4 MODEL EXPLORATION & TUNING

4.1 Data Pre-processing

In [39]:
```
# categorical to dummy
df_transformed['staff_pick'] = df_transformed['staff_pick'].astype(str)
df_dummy = pd.get_dummies(df_transformed)
df_dummy.shape
```

Out[39]:
```
(20625, 45)
```

In [42]:
```
x_unscaled = df_dummy.drop('SuccessfulBool', axis=1)
y = df_dummy['SuccessfulBool']
y.value_counts() # roughly equal size, no need to balance
```

Out[42]:
```
0    14607
1     6018
Name: SuccessfulBool, dtype: int64
```

In [44]:
```
# use standardization to converge faster
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = pd.DataFrame(scaler.fit_transform(x_unscaled), columns=x_unscaled.columns)
```

In [45]:
```
x_scaled.head()
```

Out[45]:

	goal	name_len_clean	blurb_len_clean	deadline_day	deadline_yr	state_changed_at_month	state_changed_at_day	state_changed_at_yr	state_changed_at_hr	crea
0	-0.069340	1.532720	0.888643	0.808942	0.131581	-1.648733	0.815877	0.137885	-0.473941	
1	-0.070088	-0.534095	0.584177	-1.627970	0.131581	-0.488674	-1.627408	0.137885	0.522105	
2	0.004404	1.119357	-0.938155	1.141248	0.131581	-1.068703	1.149052	0.137885	-0.805957	
3	-0.066719	0.292631	-0.024756	-1.074126	-0.655960	0.961401	-1.072116	-0.651423	-2.134019	
4	-0.068051	0.705994	1.497575	1.252017	0.919123	-0.198659	1.260110	0.927193	-0.141926	

5 rows × 44 columns

In [46]:
```
# Splitting into train and test sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.3, random_state=504)
```

4.2 Decision Tree

In [59]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=504)
dt.fit(x_train, y_train)
```

Out[59]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=504)
```

In [60]:
```
y_pred_dt = dt.predict(x_test)
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score

conf_matrix = pd.DataFrame(confusion_matrix(y_test, y_pred_dt),
                           index = ['actual 0', 'actual 1'],
                           columns = ['predicted 0', 'predicted 1'])
conf_matrix
```

Out[60]:

	predicted 0	predicted 1
actual 0	3499	923
actual 1	868	898

In [61]:
```
print('Accuracy score of basic decision tree model: {}%'.format(accuracy_score(y_test, y_pred_dt)*100))
print('Precision score of basic decision Tree Model: {}%'.format(precision_score(y_test, y_pred_dt)*100))
```

```
Accuracy score of basic decision tree model: 71.05688429217841%
Precision score of basic decision Tree Model: 49.313563975837454%
```

4.3 Random Forest

In [74]:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 400, random_state = 504)
rf.fit(x_train, y_train)
```

Out[74]:

```
RandomForestClassifier
RandomForestClassifier(n_estimators=400, random_state=504)
```

In [75]:

```
y_pred_rf = rf.predict(x_test)
conf_matrix1 = pd.DataFrame(confusion_matrix(y_test, y_pred_rf),
                            index = ['actual 0', 'actual 1'],
                            columns = ['predicted 0', 'predicted 1'])
conf_matrix1
```

Out[75]:

	predicted 0	predicted 1
actual 0	4130	292
actual 1	1047	719

In [76]:

```
print('Accuracy of Random Forest Model: {}%'.format(accuracy_score(y_test, y_pred_rf)*100))
print('Precision of Random Forest Model: {}%'.format(precision_score(y_test, y_pred_rf)*100))
```

```
Accuracy of Random Forest Model: 78.36134453781513%
Precision of Random Forest Model: 71.11770524233432%
```

4.4 Logistic Regression

In [77]:

```
import statsmodels.api as sm
log_reg=sm.Logit(list(y_train),x_train).fit(maxiter=10000)
log_reg.summary()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.625257
         Iterations: 10000

/opt/anaconda3/envs/504/lib/python3.10/site-packages/statsmodels/base/model.py:604: ConvergenceWarning: Maximum Likelihood
optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Out[77]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	14437
Model:	Logit	**Df Residuals:**	14398
Method:	MLE	**Df Model:**	38
Date:	Sun, 09 Oct 2022	**Pseudo R-squ.:**	-0.03152
Time:	16:22:52	**Log-Likelihood:**	-9026.8
converged:	False	**LL-Null:**	-8751.0
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
goal	-0.4634	0.105	-4.422	0.000	-0.669	-0.258
name_len_clean	0.2005	0.018	10.943	0.000	0.165	0.236
blurb_len_clean	0.0619	0.018	3.422	0.001	0.026	0.097
deadline_day	-0.0424	0.038	-1.129	0.259	-0.116	0.031
deadline_yr	-1.5754	0.330	-4.768	0.000	-2.223	-0.928
state_changed_at_month	0.8726	2.652	0.329	0.742	-4.325	6.070
state_changed_at_day	0.0992	0.231	0.430	0.667	-0.353	0.552
state_changed_at_yr	5.1682	11.724	0.441	0.659	-17.810	28.147
state_changed_at_hr	0.0693	0.025	2.795	0.005	0.021	0.118
created_at_month	-1.5175	2.798	-0.542	0.588	-7.002	3.967
created_at_day	-0.1318	0.241	-0.546	0.585	-0.605	0.341
created_at_yr	-6.8117	12.796	-0.532	0.595	-31.892	18.268
created_at_hr	-0.0005	0.020	-0.026	0.979	-0.040	0.039
launched_at_month	0.6841	3.296	0.208	0.836	-5.777	7.145
launched_at_day	0.0621	0.282	0.220	0.825	-0.490	0.614
launched_at_yr	3.1028	14.779	0.210	0.834	-25.864	32.070
launched_at_hr	-0.1117	0.025	-4.414	0.000	-0.161	-0.062
create_to_launch_days	-1.6681	3.057	-0.546	0.585	-7.659	4.323
launch_to_state_change_days	-0.0206	0.360	-0.057	0.954	-0.727	0.686
LaunchedTuesday	0.0210	nan	nan	nan	nan	nan
USorNot	0.0620	0.019	3.281	0.001	0.025	0.099
staff_pick_False	-0.4194	nan	nan	nan	nan	nan
staff_pick_True	0.4160	nan	nan	nan	nan	nan
state_changed_at_weekday_Friday	0.0137	nan	nan	nan	nan	nan
state_changed_at_weekday_Monday	-0.0130	nan	nan	nan	nan	nan
state_changed_at_weekday_Saturday	0.0225	nan	nan	nan	nan	nan
state_changed_at_weekday_Sunday	0.0118	nan	nan	nan	nan	nan
state_changed_at_weekday_Thursday	-0.0105	nan	nan	nan	nan	nan
state_changed_at_weekday_Tuesday	-0.0268	nan	nan	nan	nan	nan
state_changed_at_weekday_Wednesday	0.0015	nan	nan	nan	nan	nan
created_at_weekday_Friday	-0.0054	nan	nan	nan	nan	nan
created_at_weekday_Monday	0.0122	nan	nan	nan	nan	nan
created_at_weekday_Saturday	-0.0233	nan	nan	nan	nan	nan
created_at_weekday_Sunday	-0.0040	nan	nan	nan	nan	nan
created_at_weekday_Thursday	0.0177	nan	nan	nan	nan	nan
created_at_weekday_Tuesday	0.0046	nan	nan	nan	nan	nan

created_at_weekday_Wednesday	-0.0066	nan	nan	nan	nan	nan
launched_at_weekday_Friday	-0.0319	4.48e+05	-7.14e-08	1.000	-8.77e+05	8.77e+05
launched_at_weekday_Monday	-0.0057	5.25e+05	-1.08e-08	1.000	-1.03e+06	1.03e+06
launched_at_weekday_Saturday	-0.0026	2.9e+05	-9.01e-09	1.000	-5.68e+05	5.68e+05
launched_at_weekday_Sunday	0.0523	2.87e+05	1.82e-07	1.000	-5.63e+05	5.63e+05
launched_at_weekday_Thursday	-0.0111	4.65e+05	-2.39e-08	1.000	-9.12e+05	9.12e+05
launched_at_weekday_Tuesday	0.0210	nan	nan	nan	nan	nan
launched_at_weekday_Wednesday	-0.0068	5.02e+05	-1.35e-08	1.000	-9.84e+05	9.84e+05

We can find the mle does not converge and this may be caused by many dummy variables in the data.

In [78]:
```
np.diag(log_reg.cov_params()) # the negative caused nans above
```

Out[78]:
```
array([ 1.09816575e-02,  3.35631311e-04,  3.27349593e-04,  1.40839825e-03,
        1.09156373e-01,  7.03251569e+00,  5.32883589e-02,  1.37449169e+02,
        6.14354539e-04,  7.83148563e+00,  5.81978389e-02,  1.63742175e+02,
        4.14228369e-04,  1.08667148e+01,  7.93585731e-02,  2.18427751e+02,
        6.40534859e-04,  9.34418478e+00,  1.29896647e-01, -1.21878576e+14,
        3.56790539e-04, -2.52321450e+12, -2.52075981e+12, -5.30155412e+12,
       -3.78144034e+12, -4.16833807e+12, -4.26383963e+12, -5.07964029e+12,
       -3.28767504e+12, -4.84437577e+12, -1.87741726e+12, -2.30099043e+12,
       -1.48815823e+12, -1.66145680e+12, -2.13776953e+12, -2.35801687e+12,
       -2.20313586e+12,  2.00266416e+11,  2.75218149e+11,  8.38748207e+10,
        8.26202095e+10,  2.16472822e+11, -1.28332763e+14,  2.52028147e+11])
```

In [84]:
```
# if we drop created_at_weekday, launched_at_weekday and staff_pick, might be correlated
df_transformed1 = df_transformed.drop(['created_at_weekday', 'state_changed_at_weekday', 'staff_pick',
                                       'LaunchedTuesday', ], axis=1)
```

In [145]:
```
df_transformed1.shape
```

Out[145]:
```
(20625, 22)
```

In [146]:
```
df_transformed1.columns
```

Out[146]:
```
Index(['goal', 'name_len_clean', 'blurb_len_clean', 'launched_at_weekday',
       'deadline_day', 'deadline_yr', 'state_changed_at_month',
       'state_changed_at_day', 'state_changed_at_yr', 'state_changed_at_hr',
       'created_at_month', 'created_at_day', 'created_at_yr', 'created_at_hr',
       'launched_at_month', 'launched_at_day', 'launched_at_yr',
       'launched_at_hr', 'create_to_launch_days',
       'launch_to_state_change_days', 'SuccessfulBool', 'USorNot'],
      dtype='object')
```

In [85]:
```
df_dummy1 = pd.get_dummies(df_transformed1)
df_dummy1.shape
```

Out[85]:
```
(20625, 28)
```

In [86]:
```
x_unscaled1 = df_dummy1.drop('SuccessfulBool', axis=1)
y1 = df_dummy1['SuccessfulBool']
x_scaled1 = pd.DataFrame(scaler.fit_transform(x_unscaled1), columns=x_unscaled1.columns)
x_train1, x_test1, y_train1, y_test1 = train_test_split(x_scaled1, y1, test_size=0.3, random_state=504)
```

In [96]:

```
log_reg1=sm.Logit(list(y_train1),x_train1).fit(maxiter=10000)
log_reg1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.674140
         Iterations 12
```

Out[96]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	14437
Model:	Logit	**Df Residuals:**	14411
Method:	MLE	**Df Model:**	25
Date:	Sun, 09 Oct 2022	**Pseudo R-squ.:**	-0.1122
Time:	16:40:22	**Log-Likelihood:**	-9732.6
converged:	True	**LL-Null:**	-8751.0
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
goal	-0.3805	0.086	-4.402	0.000	-0.550	-0.211
name_len_clean	0.2218	0.018	12.611	0.000	0.187	0.256
blurb_len_clean	0.0648	0.017	3.736	0.000	0.031	0.099
deadline_day	-0.0576	0.036	-1.585	0.113	-0.129	0.014
deadline_yr	-1.6964	0.310	-5.469	0.000	-2.304	-1.088
state_changed_at_month	2.4900	2.533	0.983	0.326	-2.474	7.454
state_changed_at_day	0.2511	0.221	1.138	0.255	-0.181	0.683
state_changed_at_yr	12.3556	11.197	1.104	0.270	-9.589	34.301
state_changed_at_hr	0.0988	0.023	4.241	0.000	0.053	0.144
created_at_month	-2.4060	2.649	-0.908	0.364	-7.597	2.785
created_at_day	-0.2107	0.228	-0.922	0.356	-0.658	0.237
created_at_yr	-10.9071	12.111	-0.901	0.368	-34.644	12.829
created_at_hr	-0.0136	0.019	-0.702	0.483	-0.052	0.024
launched_at_month	0.0040	3.123	0.001	0.999	-6.117	6.125
launched_at_day	0.0072	0.267	0.027	0.978	-0.516	0.530
launched_at_yr	0.0635	14.002	0.005	0.996	-27.381	27.508
launched_at_hr	-0.1675	0.024	-6.999	0.000	-0.214	-0.121
create_to_launch_days	-2.6121	2.893	-0.903	0.367	-8.282	3.058
launch_to_state_change_days	-0.2283	0.344	-0.663	0.507	-0.903	0.446
USorNot	0.0870	0.018	4.813	0.000	0.052	0.122
launched_at_weekday_Friday	-0.0509	1.03e+06	-4.92e-08	1.000	-2.03e+06	2.03e+06
launched_at_weekday_Monday	-0.0115	1.21e+06	-9.51e-09	1.000	-2.38e+06	2.38e+06
launched_at_weekday_Saturday	-0.0184	6.69e+05	-2.74e-08	1.000	-1.31e+06	1.31e+06
launched_at_weekday_Sunday	0.0415	6.64e+05	6.25e-08	1.000	-1.3e+06	1.3e+06
launched_at_weekday_Thursday	-0.0162	1.08e+06	-1.51e-08	1.000	-2.11e+06	2.11e+06
launched_at_weekday_Tuesday	0.0530	1.26e+06	4.21e-08	1.000	-2.47e+06	2.47e+06
launched_at_weekday_Wednesday	0.0018	1.16e+06	1.55e-09	1.000	-2.27e+06	2.27e+06

In [106]:

```
y_pred_all = log_reg1.predict(x_test1)
predicted_choice = (y_pred_all > 0.5).astype(int)
conf_matrix2 = pd.DataFrame(confusion_matrix(y_test1,predicted_choice),
                            index = ['actual 0', 'actual 1'],
                            columns = ['predicted 0', 'predicted 1'])
conf_matrix2
```

Out[106]:

	predicted 0	predicted 1
actual 0	2368	2054
actual 1	592	1174

In [108]:

```
print('Accuracy of logistic regression Model: {}%'.format(accuracy_score(y_test1, predicted_choice)*100))
print('Precision of logistic regression Model: {}%'.format(precision_score(y_test1, predicted_choice)*100))
```

```
Accuracy of logistic regression Model: 57.23981900452488%
Precision of logistic regression Model: 36.36926889714994%
```

In [140]:

```
odds_ratios = pd.DataFrame(
    {
        "OR": log_reg1.params,
        "Lower CI": log_reg1.conf_int()[0],
        "Upper CI": log_reg1.conf_int()[1],
    }
)
odds_ratios = round(np.exp(odds_ratios),4)
odds_ratios
```

```
/opt/anaconda3/envs/504/lib/python3.10/site-packages/pandas/core/internals/blocks.py:352: RuntimeWarning: overflow encounte
red in exp
  result = func(self.values, **kwargs)
```

Out[140]:

	OR	Lower CI	Upper CI
goal	0.6835	0.5770	8.097000e-01
name_len_clean	1.2483	1.2060	1.292100e+00
blurb_len_clean	1.0669	1.0313	1.103800e+00
deadline_day	0.9441	0.8792	1.013700e+00
deadline_yr	0.1833	0.0998	3.368000e-01
state_changed_at_month	12.0614	0.0842	1.726810e+03
state_changed_at_day	1.2854	0.8343	1.980500e+00
state_changed_at_yr	232265.8168	0.0001	7.882003e+14
state_changed_at_hr	1.1038	1.0546	1.155400e+00
created_at_month	0.0902	0.0005	1.620340e+01
created_at_day	0.8100	0.5177	1.267400e+00
created_at_yr	0.0000	0.0000	3.730113e+05
created_at_hr	0.9865	0.9496	1.024700e+00
launched_at_month	1.0040	0.0022	4.573063e+02
launched_at_day	1.0073	0.5969	1.699800e+00
launched_at_yr	1.0656	0.0000	8.839066e+11
launched_at_hr	0.8457	0.8070	8.864000e-01
create_to_launch_days	0.0734	0.0003	2.128760e+01
launch_to_state_change_days	0.7959	0.4054	1.562700e+00
USorNot	1.0909	1.0530	1.130300e+00
launched_at_weekday_Friday	0.9504	0.0000	inf
launched_at_weekday_Monday	0.9885	0.0000	inf
launched_at_weekday_Saturday	0.9818	0.0000	inf
launched_at_weekday_Sunday	1.0424	0.0000	inf
launched_at_weekday_Thursday	0.9839	0.0000	inf
launched_at_weekday_Tuesday	1.0544	0.0000	inf
launched_at_weekday_Wednesday	1.0018	0.0000	inf

4.4.1 Only us

In [91]:

```
df_us_dummy = (df_dummy1[df_dummy1['USorNot'] == 1]).drop('USorNot', axis=1)
x_us = df_us_dummy.drop('SuccessfulBool', axis=1)
y_us = df_us_dummy['SuccessfulBool']
x_us_scaled = pd.DataFrame(scaler.fit_transform(x_us), columns=x_us.columns)
x_train2, x_test2, y_train2, y_test2 = train_test_split(x_us_scaled, y_us, test_size=0.3, random_state=504)
```

In [144]:
```
df_us_dummy
```

Out[144]:

	goal	name_len_clean	blurb_len_clean	deadline_day	deadline_yr	state_changed_at_month	state_changed_at_day	state_changed_at_yr	state_changed_at_hr
0	1500.0	9.0	16.0	23	2015	1	23	2015	10
1	500.0	4.0	15.0	1	2015	5	1	2015	16
2	100000.0	8.0	10.0	26	2015	3	26	2015	8
3	5000.0	6.0	13.0	6	2014	10	6	2014	0
5	13000.0	8.0	15.0	20	2015	11	20	2015	10
...	...	...	...	...	...	...	...	...	...
20621	6000.0	7.0	15.0	7	2015	4	7	2015	17
20622	1000.0	3.0	15.0	7	2015	6	7	2015	20
20624	40000.0	1.0	8.0	16	2015	4	16	2015	4
20629	10000.0	3.0	17.0	14	2015	4	14	2015	12
20630	2500.0	1.0	6.0	20	2015	5	20	2015	13

14136 rows × 27 columns

In [103]:

```
log_reg2=sm.Logit(list(y_train2),x_train2).fit(maxiter=10000)
log_reg2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.673327
         Iterations 9
```

Out[103]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	9895
Model:	Logit	**Df Residuals:**	9870
Method:	MLE	**Df Model:**	24
Date:	Sun, 09 Oct 2022	**Pseudo R-squ.:**	-0.08965
Time:	16:44:43	**Log-Likelihood:**	-6662.6
converged:	True	**LL-Null:**	-6114.4
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
goal	-0.3470	0.111	-3.132	0.002	-0.564	-0.130
name_len_clean	0.2156	0.021	10.086	0.000	0.174	0.257
blurb_len_clean	0.0627	0.021	2.965	0.003	0.021	0.104
deadline_day	-0.0843	0.045	-1.876	0.061	-0.172	0.004
deadline_yr	-1.6877	0.445	-3.793	0.000	-2.560	-0.816
state_changed_at_month	1.8074	2.998	0.603	0.547	-4.069	7.684
state_changed_at_day	0.1949	0.264	0.737	0.461	-0.323	0.713
state_changed_at_yr	9.9532	14.358	0.693	0.488	-18.187	38.094
state_changed_at_hr	0.0647	0.027	2.407	0.016	0.012	0.117
created_at_month	-5.5263	3.169	-1.744	0.081	-11.737	0.685
created_at_day	-0.4943	0.275	-1.796	0.072	-1.034	0.045
created_at_yr	-27.2241	15.730	-1.731	0.083	-58.054	3.605
created_at_hr	0.0220	0.023	0.965	0.334	-0.023	0.067
launched_at_month	3.8189	3.707	1.030	0.303	-3.446	11.084
launched_at_day	0.3155	0.320	0.987	0.323	-0.311	0.942
launched_at_yr	18.5653	18.065	1.028	0.304	-16.841	53.972
launched_at_hr	-0.1389	0.028	-4.998	0.000	-0.193	-0.084
create_to_launch_days	-6.5172	3.755	-1.736	0.083	-13.876	0.842
launch_to_state_change_days	-0.1260	0.411	-0.307	0.759	-0.931	0.679
launched_at_weekday_Friday	-0.0771	5.18e+05	-1.49e-07	1.000	-1.01e+06	1.01e+06
launched_at_weekday_Monday	-0.0054	6.05e+05	-8.96e-09	1.000	-1.19e+06	1.19e+06
launched_at_weekday_Saturday	-0.0255	3.3e+05	-7.71e-08	1.000	-6.47e+05	6.47e+05
launched_at_weekday_Sunday	0.0357	3.24e+05	1.1e-07	1.000	-6.35e+05	6.35e+05
launched_at_weekday_Thursday	-0.0272	5.33e+05	-5.12e-08	1.000	-1.04e+06	1.04e+06
launched_at_weekday_Tuesday	0.0831	6.29e+05	1.32e-07	1.000	-1.23e+06	1.23e+06
launched_at_weekday_Wednesday	0.0040	5.75e+05	6.96e-09	1.000	-1.13e+06	1.13e+06

In [139]:

```
# calculate odd ratios
odds_ratios = pd.DataFrame(
    {
        "OR": log_reg2.params,
        "Lower CI": log_reg2.conf_int()[0],
        "Upper CI": log_reg2.conf_int()[1],
    }
)
odds_ratios = round(np.exp(odds_ratios),4)
odds_ratios
```

```
/opt/anaconda3/envs/504/lib/python3.10/site-packages/pandas/core/internals/blocks.py:352: RuntimeWarning: overflow encounte
red in exp
  result = func(self.values, **kwargs)
```

Out[139]:

	OR	Lower CI	Upper CI
goal	7.068000e-01	0.5689	8.782000e-01
name_len_clean	1.240600e+00	1.1897	1.293600e+00
blurb_len_clean	1.064700e+00	1.0215	1.109700e+00
deadline_day	9.192000e-01	0.8417	1.003800e+00
deadline_yr	1.849000e-01	0.0773	4.424000e-01
state_changed_at_month	6.094700e+00	0.0171	2.172671e+03
state_changed_at_day	1.215200e+00	0.7237	2.040400e+00
state_changed_at_yr	2.101969e+04	0.0000	3.499154e+16
state_changed_at_hr	1.066800e+00	1.0121	1.124500e+00
created_at_month	4.000000e-03	0.0000	1.983400e+00
created_at_day	6.100000e-01	0.3557	1.046100e+00
created_at_yr	0.000000e+00	0.0000	3.679460e+01
created_at_hr	1.022200e+00	0.9776	1.068800e+00
launched_at_month	4.555440e+01	0.0319	6.510854e+04
launched_at_day	1.371000e+00	0.7329	2.564600e+00
launched_at_yr	1.155588e+08	0.0000	2.752466e+23
launched_at_hr	8.703000e-01	0.8241	9.190000e-01
create_to_launch_days	1.500000e-03	0.0000	2.321000e+00
launch_to_state_change_days	8.816000e-01	0.3942	1.971600e+00
launched_at_weekday_Friday	9.258000e-01	0.0000	inf
launched_at_weekday_Monday	9.946000e-01	0.0000	inf
launched_at_weekday_Saturday	9.748000e-01	0.0000	inf
launched_at_weekday_Sunday	1.036300e+00	0.0000	inf
launched_at_weekday_Thursday	9.731000e-01	0.0000	inf
launched_at_weekday_Tuesday	1.086600e+00	0.0000	inf
launched_at_weekday_Wednesday	1.004000e+00	0.0000	inf

In [107]:

```
y_pred_us = log_reg2.predict(x_test2)
predicted_choice1 = (y_pred_us > 0.5).astype(int)
conf_matrix3 = pd.DataFrame(confusion_matrix(y_test2, predicted_choice1),
                            index = ['actual 0', 'actual 1'],
                            columns = ['predicted 0', 'predicted 1'])
conf_matrix3
```

Out[107]:

	predicted 0	predicted 1
actual 0	1597	1335
actual 1	447	862

In [109]:

```
print('Accuracy of logistic regression Model: {}%'.format(accuracy_score(y_test2, predicted_choice1)*100))
print('Precision of logistic regression Model: {}%'.format(precision_score(y_test2, predicted_choice1)*100))
```

```
Accuracy of logistic regression Model: 57.981608111294506%
Precision of logistic regression Model: 39.23532089212563%
```

In [120]:

```
df_us = df[df['country']=='US']
df_condition = df_us[(df_us['goal'] >= 25000) & (df_us['backers_count'] >= 1000)]
# prob of successful
print('The probability of a successful campaign in such condition is {}%'.format(
    round(len(df_condition[df_condition['SuccessfulBool']==1])/df_condition.shape[0], 4)*100))
```

```
The probability of a successful campaign in such condition is 95.32000000000001%
```

In [131]:

```
df_condition['USorNot'] = np.where(df_condition['country'] == 'US', 1, 0)
cols = df_transformed.columns.values.tolist()
df_condition[cols+['usd_pledged', 'backers_count']]
```

```
/var/folders/qx/wlpqnbz178962z833f_yq5gh0000gn/T/ipykernel_61643/1073711085.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_condition['USorNot'] = np.where(df_condition['country'] == 'US', 1, 0)
```

Out[131]:

	goal	staff_pick	name_len_clean	blurb_len_clean	state_changed_at_weekday	created_at_weekday	launched_at_weekday	deadline_day	deadline_yr	state_cha
300	150000.0	True	5.0	12.0	Monday	Friday	Thursday	10	2012	
301	30000.0	True	8.0	17.0	Wednesday	Wednesday	Monday	15	2015	
304	50000.0	True	9.0	16.0	Thursday	Monday	Tuesday	6	2016	
306	300000.0	True	6.0	7.0	Friday	Tuesday	Wednesday	11	2016	
308	50000.0	True	2.0	13.0	Saturday	Tuesday	Thursday	27	2015	
...	...	...	...	...	...	...	...	...	...	
19737	38500.0	False	8.0	17.0	Monday	Friday	Friday	20	2015	
19746	39500.0	False	6.0	7.0	Thursday	Monday	Tuesday	19	2016	
19829	40000.0	False	8.0	15.0	Tuesday	Tuesday	Tuesday	28	2015	
19929	50000.0	False	9.0	11.0	Saturday	Thursday	Wednesday	16	2016	
19940	35000.0	False	4.0	12.0	Wednesday	Sunday	Wednesday	9	2016	

449 rows × 28 columns

In [ ]: