Predicting Success for Kickstarter Campaigns

Abstract

Kickstarter is an online funding platform for people trying to get their personal projects off the ground. Projects fall into a variety of categories and usually offer a reward system for donors that reflects their pledge amount.

The twist is that funding is all or nothing. If a project falls short of its funding goal at the pre-selected deadline, the project fails and the creator receives NO funding (all pledges are returned to their respective donors).

Therefore both campaign creators AND donors have a strong incentive to make sure that their projects have the best chance of success possible!

Research Questions

1. If I'm looking for campaigns to support, how can I determine which are most likely to succeed?
   - Are some categories more successful than others?
2. If I'm starting a campaign, which factors should I focus on to give it the best chance of success?
   - Does it matter where I'm located?
   - Does it matter in which month my campaign is launched?

Data Sources

(dataset 1) https://www.kaggle.com/wood2174/mapkickstarter (https://www.kaggle.com/wood2174/mapkickstarter)
(dataset 2) https://www.kaggle.com/kemical/kickstarter-projects (https://www.kaggle.com/kemical/kickstarter-projects)

In [3]:
```
# Import necessary libraries

import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import plotly
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.offline import iplot as plot, init_notebook_mode
init_notebook_mode(connected=True)
```

In [4]:
```
filepath = "data/mapkickstarter/"
df = pd.read_csv(filepath + "MasterKickstarter.csv")
```

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

Out[5]:
```
(99036, 57)
```

In [6]:
```
df.keys()
```

Out[6]:
```
Index(['Unnamed: 0', 'X1', 'X1_1', 'Country', 'City', 'id', 'name', 'blurb',
       'goal', 'pledged', 'status', 'slug', 'disable_communication',
       'currency', 'currency_symbol', 'currency_trailing_code', 'deadline',
       'state_changed_at', 'created_at', 'launched_at', 'staff_pick',
       'backers_count', 'deadlineTime', 'state_changed_atTime',
       'created_atTime', 'launched_atTime', 'Categories', 'spotLight',
       'pledgedUSD', 'Ex_USd', 'deadlineYM', 'state_changed_atYM',
       'created_atYM', 'launched_atYM', 'deadlineY', 'state_changed_atY',
       'created_atY', 'launched_atY', 'Pledge_per_person', 'Prct_goal',
       'Length_of_kick', 'City_Pop', 'Latitude', 'Longitude', 'County',
       'State', 'Backers_as_Prct_of_Pop', 'Backers_as_Prct_of_Pop_YM',
       'Backers_as_Prct_of_Pop_Y', 'Days_spent_making_campign',
       'Days_inception_to_Deadline', 'Backers_in_city_Y', 'Backers_in_city_YM',
       'All_Time_Backers_city', 'Mean_Pledge_City', 'Mean_pledge_city_Y',
       'Mean_pledge_city_YM'],
      dtype='object')
```

Data Filtering

For now we will only use campaigns with USD currency, since there aren't any features that normalize/convert other currencies. Effectively this limits the study to campaigns launched in the United States.

Additionally, we will select the features that we want to consider for our study.

In [8]:
```
# Filter by currency to extract only USA-based Kickstarters
df_filter = df.copy()
df_filter = df_filter[df_filter['currency'] == 'USD']

# Feature selection
features = ['id','goal','status','Categories','City','State','launched_atYM','Length_of_kick','Days_spent_making_campign','City_P
op','staff_pick']
df_select = df_filter.copy()[features]
df_select.head()
```

Out[8]:

	id	goal	status	Categories	City	State	launched_atYM	Length_of_kick	Days_spent_making_campign	City_Pop
4344	1365968739	4500.0	failed	music	Abilene	texas	16-01	31	7	114247
4345	730363660	1860.0	failed	photography	Abilene	texas	14-08	30	10	114247
4346	982082961	6000.0	successful	music	Abilene	texas	14-02	30	492	114247
4347	1880062664	35551.0	failed	film%20&%20video	Abilene	texas	16-07	30	76	114247
4348	1068762173	10000.0	canceled	film%20&%20video	Abilene	texas	14-07	30	1	114247

Data Cleaning

A few items to note in the data cleaning steps below:

1. We will consider 'canceled' projects to have 'failed', since this study is meant to indicate a project's chance of success from the perspective of an outside donor. Therefore we will keep 'successful' as '1' and 'failed/canceled' as '0', dropping all other statuses.
2. For now we will only use campaigns with USD currency, since there aren't any features that normalize/convert other currencies. Effectively this limits the study to campaigns launched in the United States.
3. We'll need to create a parallel feature to 'Categories' that converts categories to numbers, in order for the DecisionTreeClassifier model that we'll use later on to work. Same for 'City' and 'State'.

In [9]:
```
# Clean up values for later calculations

# Change Statuses to binary 1/0 for Success/Fail; remove all other entries
df_select['status'] = df_select['status'].replace('successful', 1)
df_select['status'] = df_select['status'].replace('failed', 0)
df_select['status'] = df_select['status'].replace('canceled', 0)
df_select = df_select[(df_select['status'] == 1) | (df_select['status'] == 0)]
df_select['status'] = df_select['status'].astype(str).astype(int)

# Convert Staff pick to numerical binary
df_select['staff_pick'] = df_select['staff_pick']*1

df_select['Categories'] = df_select['Categories'].replace('film%20&%20video', 'film+video')
df_select['launched_atYM'] = df['launched_atYM'].str.extract('.*-(.*)')

# Create parallel categories for City, State and Category with numbers only
cities = list(set(df_filter['City']))
states = list(set(df_filter['State']))
categories = list(set(df_select['Categories']))

df_select['Cat-Nums'] = df_select['Categories'].replace(categories, list(range(len(categories))))
df_select['City-Nums'] = df_select['City'].replace(cities, list(range(len(cities))))
df_select['State-Nums'] = df_select['State'].replace(cities, list(range(len(cities))))


# Remove rows with lingering null values
df_select = df_select.dropna()

df_select.head()
```

Out[9]:

	id	goal	status	Categories	City	State	launched_atYM	Length_of_kick	Days_spent_making_campign	City_Pop	staff_pick
4344	1365968739	4500.0	0	music	Abilene	texas	01	31	7	114247	0
4345	730363660	1860.0	0	photography	Abilene	texas	08	30	10	114247	0
4346	982082961	6000.0	1	music	Abilene	texas	02	30	492	114247	0
4347	1880062664	35551.0	0	film+video	Abilene	texas	07	30	76	114247	0
4348	1068762173	10000.0	0	film+video	Abilene	texas	07	30	1	114247	0

First, let's see what the overall success rate is for projects in this dataset:

In [20]:
```
success_rate = df_select['status'].mean()
success_rate
```

Out[20]: 0.4192610697166722

About **42%** of projects overall are successfully funded.

Venturing a hypothesis, let's see if there is are different success rates among the main categories:

In [11]:
```
# Create DataFrame copy grouped by Categories
df_select_cats = df_select.copy()
df_select_cats = df_select_cats.groupby(['Categories'], as_index=False).mean()
df_select_cats
```

Out[11]:

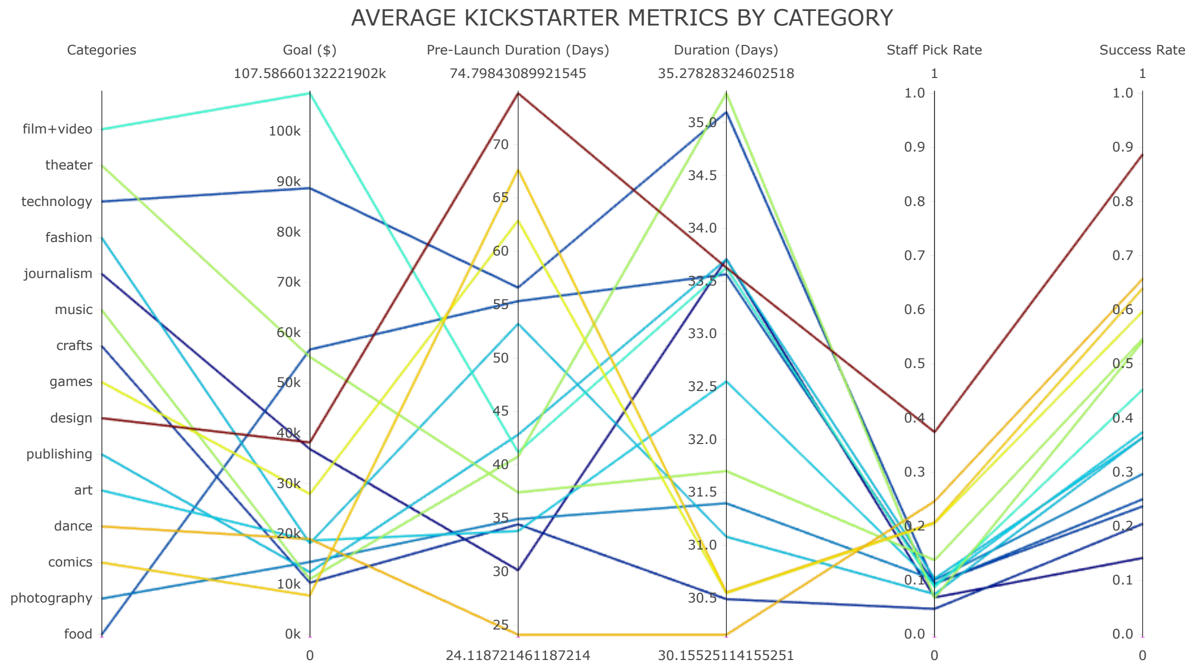
	Categories	id	goal	status	Length_of_kick	Days_spent_making_campign	City_Pop	staff_pick	Cat-Nums	City-Nun
0	art	1.077749e+09	18710.927331	0.374497	32.551914	33.826492	1.095829e+06	0.092331	4.0	1200.1344
1	comics	1.060169e+09	7783.690871	0.640041	30.556017	67.634855	1.099350e+06	0.207469	2.0	1268.4823
2	crafts	1.091693e+09	10338.137931	0.204981	30.492337	34.455939	5.418500e+05	0.047893	8.0	1259.8505
3	dance	1.028531e+09	19019.698630	0.657534	30.155251	24.118721	1.763997e+06	0.246575	3.0	1088.7442
4	design	1.102715e+09	38218.805069	0.887749	33.628244	74.798431	1.434374e+06	0.374170	6.0	1174.6312
5	fashion	1.035238e+09	18153.502658	0.364442	31.082693	53.228588	1.280850e+06	0.075015	11.0	1189.0897
6	film+video	1.066851e+09	107586.601322	0.453147	33.628485	41.200489	1.793613e+06	0.087525	14.0	1147.9212
7	food	1.078181e+09	56667.390687	0.250330	33.565390	55.338177	6.427551e+05	0.093791	0.0	1236.2120
8	games	1.070333e+09	27971.366096	0.597201	30.548367	62.896112	8.229019e+05	0.205910	7.0	1230.5452
9	journalism	1.083949e+09	36894.259831	0.141854	33.710674	30.153090	1.082486e+06	0.068820	10.0	1202.4887
10	music	1.079030e+09	11077.422939	0.542276	35.278283	40.803419	1.298518e+06	0.067549	9.0	1184.4841
11	photography	1.055361e+09	14456.145351	0.297250	31.398952	34.941947	1.135248e+06	0.101266	1.0	1240.7826
12	publishing	1.083332e+09	12418.899630	0.363826	33.706925	42.853195	9.879979e+05	0.103851	5.0	1208.2764
13	technology	1.072904e+09	88732.675021	0.237258	35.100505	56.610765	1.014257e+06	0.094786	12.0	1224.7988
14	theater	1.061051e+09	55209.404489	0.546633	31.703741	37.440898	1.953753e+06	0.137656	13.0	1092.3885

In [12]:
```
df_select_cats['Cat-Nums'] = df_select_cats['Categories'].replace(categories, list(range(15)))

data = [
    go.Parcoords(
        line = dict(
                  color = df_select_cats['status'],
                  colorscale = 'Jet'
                   ),
        dimensions = [
            dict(range = [0,15],
                tickvals = list(range(15)),
                ticktext = categories,
                label = 'Categories',
                 values = df_select_cats['Cat-Nums']
                ),
            dict(range = [0,df_select_cats['goal'].max()],
                 label = 'Goal ($)',
                 values = df_select_cats['goal']
                ),
            dict(range = [df_select_cats['Days_spent_making_campign'].min(),df_select_cats['Days_spent_making_campign'].max()],
                 label = 'Pre-Launch Duration (Days)',
                 values = df_select_cats['Days_spent_making_campign']
                ),
            dict(range = [df_select_cats['Length_of_kick'].min(),df_select_cats['Length_of_kick'].max()],
                 label = 'Duration (Days)',
                 values = df_select_cats['Length_of_kick']
                ),
            dict(range = [0,1],
                 label = 'Staff Pick Rate',
                 values = df_select_cats['staff_pick']
                ),
            dict(range = [0,1],
                 label = 'Success Rate',
                 values = df_select_cats['status']
                )
        ]
    )
]

layout = go.Layout(
    title='Average Kickstarter Metrics by Category'.upper(),
    autosize=False,
    width=960,
    height=600,
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename = 'kickstarter-categories_parallel-coordinates')
```



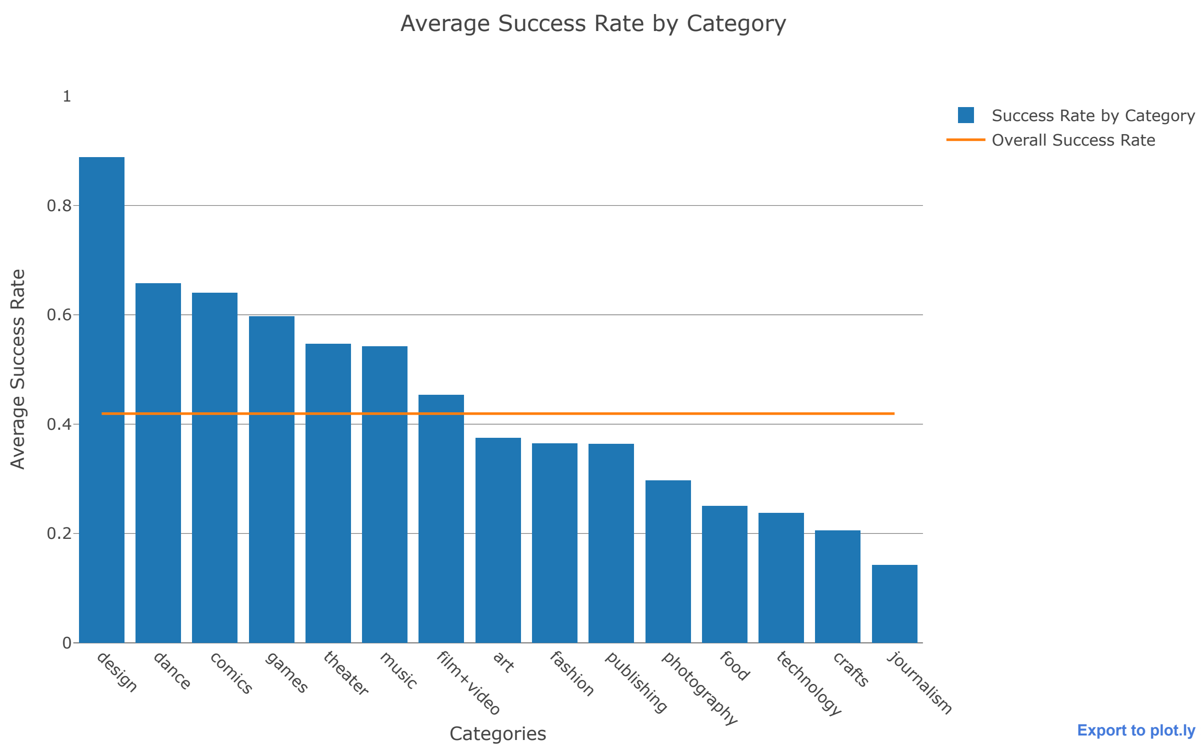
Export to plot.ly »

```
In [25]: df_select_cats = df_select_cats.sort_values(by=['status'], ascending=False)

data = [
    go.Bar(
        x=df_select_cats['Categories'],
        y=df_select_cats['status'],
        name='Success Rate by Category'
    ),
    go.Scatter(
        x=df_select_cats['Categories'],
        y=[success_rate for cat in df_select_cats['Categories']],
        mode='lines',
        name='Overall Success Rate'
    )
]

layout = go.Layout(
    title='Average Success Rate by Category',
    yaxis={'title': 'Average Success Rate', 'range': [0,1]},
    xaxis={'title': 'Categories', 'tickangle': 45},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='avg-success-rate-by-category')
```



Export to plot.ly »

Finally, let's see how much of a boost a campaign receives if labeled as a "Staff Pick":

```
In [14]: df_staff = df_select.copy()
df_staff = df_staff.groupby(['staff_pick'], as_index=False).mean()
df_staff
```

Out[14]:

	staff_pick	id	goal	status	Length_of_kick	Days_spent_making_campign	City_Pop	Cat-Nums	City-Nums
0	0	1.075390e+09	36891.891039	0.372683	33.949108	42.895471	1.107842e+06	7.840833	1205.229543
1	1	1.080529e+09	21865.249617	0.835791	32.747198	63.601864	1.763350e+06	7.433054	1135.564940

Staff picks alone are a tremendous indicator of success for a campaign. As shown above, a whopping **84%** of projects marked as "Staff pick" have been successfully funded, compared to just 37% of all other campaigns. Note that overall about 42% of projects get funded (though I've seen 36-44% according to various other sources).

Decision Tree Classifier

Split the data into training and test groups, and use a Decision Tree Classifier from Scikit-Learn to predict the success of a Kickstarter campaign.

In [17]:
```
# Set up X and y variables for test/train/split Decision Tree Classifier model

test_features = ['goal','Cat-Nums','City-Nums','launched_atYM','Length_of_kick','Days_spent_making_campign','City_Pop','staff_pic
k']
X = df_select[test_features].copy()
y = df_select[['status']].copy()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=324)
```

In [18]:
```
success_classifier = DecisionTreeClassifier()
success_classifier.fit(X_train, y_train)
predictions = success_classifier.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
```

Out[18]:
```
0.6392823757819803
```

The classifier is able to predict a campaign's success about **64%** of the time. It isn't clear yet if this accuracy score is strong or weak, but there is definitely room to improve.

However, there are additional metrics that aren't included in this dataset, but which I believe have the potential to move the needle. Additionally, we can begin to tune the parameters of our classifier(s).

Alternative Dataset

Let's check out an alternative dataset that includes sub-categories to see if this additional categorical granularity helps improve the predictive success rate.

In [220]:
```
filepath2 = "data/kickstarter-projects/"
df2 = pd.read_csv(filepath2 + "ks-projects-201801.csv", encoding = 'ISO-8859-1', low_memory=False)
```

In [221]:
```
df2.keys()
```

Out[221]:
```
Index(['ID', 'name', 'category', 'main_category', 'currency', 'deadline',
       'goal', 'launched', 'pledged', 'state', 'backers', 'country',
       'usd pledged', 'usd_pledged_real', 'usd_goal_real'],
      dtype='object')
```

In [222]:
```
df2.shape
```

Out[222]:
```
(378661, 15)
```

In [223]:
```
df2.rename(columns={'ID':'id'}, inplace=True)
```

In [224]:
```
df2.head()
```

Out[224]:

	id	name	category	main_category	currency	deadline	goal	launched	pledged	state	backers	country	usd pledged	usd_
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	2015-10-09	1000.0	2015-08-11 12:12:28	0.0	failed	0	GB	0.0	0.0
1	1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD	2017-11-01	30000.0	2017-09-02 04:43:57	2421.0	failed	15	US	100.0	2421.
2	1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26	45000.0	2013-01-12 00:20:50	220.0	failed	3	US	220.0	220.0
3	1000007540	ToshiCapital Rekordz Needs Help to Complete Album	Music	Music	USD	2012-04-16	5000.0	2012-03-17 03:24:11	1.0	failed	1	US	1.0	1.0
4	1000011046	Community Film Project: The Art of Neighborhoo...	Film & Video	Film & Video	USD	2015-08-29	19500.0	2015-07-04 08:35:03	1283.0	canceled	14	US	1283.0	1283.

Merge Datasets

This second dataset appears to have valuable features such as 'usd_pledged_real', 'usd_goal_real', and sub-categories.
Let's merge this in with the first dataset to see if the 'id' categories overlap, and we're able to augment the data dimensionality.

In [367]:
```
df_master = df2.merge(df_select, on='id', how='outer')
```

In [368]:
```
df_master.shape
```

Out[368]:
```
(378823, 25)
```

In [369]:
```
df_master = df_master.dropna()
df_master.shape
```

Out[369]:
```
(94460, 25)
```

In [370]:
```
df_master.head()
```

Out[370]:

	id	name	category	main_category	currency	deadline	goal_x	launched	pledged	state	...	goal_y	status	Ca
1	1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD	2017-11-01	30000.0	2017-09-02 04:43:57	2421.0	failed	...	30000.0	live	film%20&%
2	1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26	45000.0	2013-01-12 00:20:50	220.0	failed	...	45000.0	failed	film%20&%
5	1000014025	Monarch Espresso Bar	Restaurants	Food	USD	2016-04-01	50000.0	2016-02-26 13:38:27	52375.0	successful	...	50000.0	successful	food
11	100005484	Lisa Lim New CD!	Indie Rock	Music	USD	2013-04-08	12500.0	2013-03-09 06:42:58	12700.0	successful	...	12500.0	successful	music
17	1000068480	Notes From London: Above & Below	Art Books	Publishing	USD	2015-05-10	3000.0	2015-04-10 21:20:54	789.0	failed	...	3000.0	failed	publishing

5 rows × 25 columns

In [371]:
```
df_master.keys()
```

Out[371]:
```
Index(['id', 'name', 'category', 'main_category', 'currency', 'deadline',
       'goal_x', 'launched', 'pledged', 'state', 'backers', 'country',
       'usd pledged', 'usd_pledged_real', 'usd_goal_real', 'goal_y', 'status',
       'Categories', 'City', 'State', 'launched_atYM', 'Length_of_kick',
       'Days_spent_making_campign', 'City_Pop', 'staff_pick'],
      dtype='object')
```

In [372]:
```
merge_features = ['id', 'category', 'main_category',
       'state', 'country', 'usd_goal_real',
       'City', 'State', 'launched_atYM', 'Length_of_kick',
       'Days_spent_making_campign', 'City_Pop', 'staff_pick']
```

In [385]:
```
df_master_select = df_master.copy()
df_master_select = df_master_select[merge_features]
```

In [386]:
```
df_master_select.shape
```

Out[386]:
```
(94460, 13)
```

In [387]:
```
df_master_select.rename(columns={'state': 'Status', 'category': 'sub_category', 'launched_atYM': 'month_launched'}, inplace=True)
df_master_select.head()
```

Out[387]:

	id	sub_category	main_category	Status	country	usd_goal_real	City	State	month_launched	Length_of_kick	Day:
1	1000003930	Narrative Film	Film & Video	failed	US	30000.0	Los Angeles	california	17-09	60.0	21.0
2	1000004038	Narrative Film	Film & Video	failed	US	45000.0	Tucson	arizona	13-01	45.0	4.0
5	1000014025	Restaurants	Food	successful	US	50000.0	Tuscaloosa	alabama	16-02	35.0	39.0
11	100005484	Indie Rock	Music	successful	US	12500.0	Washington	district of columbia	13-03	30.0	5.0
17	1000068480	Art Books	Publishing	failed	US	3000.0	Pittsburgh	pennsylvania	15-04	30.0	2.0

In [388]:
```
# Clean up values for later calculations

# Change Statuses to binary 1/0 for Success/Fail; remove all other entries
df_master_select['Status'] = df_master_select['Status'].replace('successful', 1)
df_master_select['Status'] = df_master_select['Status'].replace('failed', 0)
df_master_select['Status'] = df_master_select['Status'].replace('canceled', 0)
df_master_select = df_master_select[(df_master_select['Status'] == 1) | (df_master_select['Status'] == 0)]
df_master_select['Status'] = df_master_select['Status'].astype(str).astype(int)

# Convert Staff pick to numerical binary
df_master_select['staff_pick'] = df_master_select['staff_pick']*1
df_master_select['staff_pick'] = df_master_select['staff_pick'].astype(str).astype(int)

df_master_select['month_launched'] = df_master_select['month_launched'].str.extract('.*-(.*)').astype(int)

# Create parallel categories for City, State, Country, Main_Category, and Sub_Category with numbers only
cities = list(set(df_master_select['City']))
states = list(set(df_master_select['State']))
countries = list(set(df_master_select['country']))
main_categories = list(set(df_master_select['main_category']))
sub_categories = list(set(df_master_select['sub_category']))

df_master_select['Main_Cat_Nums'] = df_master_select['main_category'].replace(main_categories, list(range(len(main_categories))))
df_master_select['Sub_Cat_Nums'] = df_master_select['sub_category'].replace(sub_categories, list(range(len(sub_categories))))
df_master_select['City-Nums'] = df_master_select['City'].replace(cities, list(range(len(cities))))
df_master_select['State-Nums'] = df_master_select['State'].replace(states, list(range(len(states))))
df_master_select['Country-Nums'] = df_master_select['country'].replace(countries, list(range(len(countries))))
```

In [389]:
```
# Remove rows with lingering null values
df_master_select = df_master_select.dropna()

df_master_select.head()
```

Out[389]:

	id	sub_category	main_category	Status	country	usd_goal_real	City	State	month_launched	Length_of_kick	Days_s
1	1000003930	Narrative Film	Film & Video	0	US	30000.0	Los Angeles	california	9	60.0	21.0
2	1000004038	Narrative Film	Film & Video	0	US	45000.0	Tucson	arizona	1	45.0	4.0
5	1000014025	Restaurants	Food	1	US	50000.0	Tuscaloosa	alabama	2	35.0	39.0
11	100005484	Indie Rock	Music	1	US	12500.0	Washington	district of columbia	3	30.0	5.0
17	1000068480	Art Books	Publishing	0	US	3000.0	Pittsburgh	pennsylvania	4	30.0	2.0

Calculating Prediction Accuracy

Now that the data is structured to our liking, we will test a couple classification models from the Scikit-Learn library to see how high we can push our accuracy for predicting the success of a Kickstarter campaign.

First, we will establish variables for our test features, and split into our X and Y groups for training and testing.

In [613]:
```
# Set up X and y variables for test/train/split Decision Tree Classifier model

test_features = ['usd_goal_real', 'Main_Cat_Nums', 'Sub_Cat_Nums', 'City-Nums', 'State-Nums', 'month_launched', 'Length_of_kick',
 'Days_spent_making_campign', 'City_Pop','staff_pick']
X = df_master_select[test_features].copy()
y = df_master_select[['Status']].copy()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=324)
```

Decision Tree Classifier

Begin with a simple Decision Tree Classifier with default, out-of-the-box settings.

In [614]:
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
predictions = dtc.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
```

Out[614]: 0.6563445567501448

An initial, out-of-the-box classifier test yielded a **65%** prediction accuracy. Remember that our previous accuracy score was **64%** before merging in the second dataset. This is not a huge improvement over the original dataset that didn't have subcategories, and so it seems like the additional categorical granularity is not a major factor.

However, while playing around with some of the DecisionTreeClassifier parameters (specifically, "max_leaf_nodes"), I noticed that the accuracy score varied from about **0.65** to **0.72**. In order to find the "best" paramater value to use, I used the GridSearchCV to test a wide range of max_leaf_node values.

In [615]:
```
dtc = DecisionTreeClassifier()
param_grid = {'max_leaf_nodes': np.arange(2,50000, 1000)}
CV_dtc = GridSearchCV(dtc, param_grid)
CV_dtc.fit(X_train, y_train)
CV_dtc.best_params_
```

Out[615]: {'max_leaf_nodes': 1002}

The 'max_leaf_nodes' parameter yields optimal accuracy at a value of **1002**.
With that value, let's re-build the model, re-calculate prediction accuracy, and find out which features play the biggest role.

In [627]:
```
dtc = DecisionTreeClassifier(max_leaf_nodes=1002)
dtc.fit(X_train, y_train)
predictions = dtc.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
```

Out[627]: 0.7184059743771326

In [628]:
```
# Which features play the biggest role in predicting campaign success:

dtc_features = sorted(list(zip(test_features, dtc.feature_importances_)), key=lambda x: x[1], reverse=True)
dtc_features
```

Out[628]:
```
[('staff_pick', 0.21400074480567696),
 ('usd_goal_real', 0.213321908442561),
 ('Sub_Cat_Nums', 0.1422118134367124),
 ('Main_Cat_Nums', 0.12482740089358024),
 ('Days_spent_making_campign', 0.11063687630724438),
 ('Length_of_kick', 0.06092865651283583),
 ('City_Pop', 0.05280036627303274),
 ('City-Nums', 0.03771322685397257),
 ('State-Nums', 0.024073984843382114),
 ('month_launched', 0.019485021631001853)]
```

In [618]:
```
# Further increase confidence of accuracy score through cross-validation of X and y data

dtc = DecisionTreeClassifier(max_leaf_nodes=1002)
scores = cross_val_score(dtc, X, np.ravel(y,order='C'), cv=10)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.72 (+/- 0.01)
```

So a simple Decision Tree classifier is able to predict a campaign's success about **72%** of the time.
Let's see if a **Random Forest** model can do any better.

Random Forest Classifier

Begin with a simple Random Forest Classifier with default, out-of-the-box settings.

In [619]:
```
rfc = RandomForestClassifier()
rfc.fit(X_train, np.ravel(y_train,order='C'))
predictions = rfc.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
```

Out[619]: 0.7132234597308955

The default RandomForestClassifier yields a **71%** prediction accuracy. This is definitely higher than the Decision Tree with default settings, but lower than the tuned Decision Tree model.

As with the Decision Tree, let's try to find some optimal settings. Here, I don't have the expertise yet to know which parameters to focus on tuning, so I relied on Google and StackOverflow, where I found a number of suggestions that the following features are important tune:

- n_estimators
- max_features
- max_depth

In [620]:
```
# WARNING: THIS CELL TAKES A LONG TIME TO CALCULATE! DO NOT RUN UNLESS NECESSARY!

# rfc = RandomForestClassifier(n_jobs=-1)
# param_grid = {'n_estimators': [10,100,1000], 'max_features': list(range(2,len(test_features)+1,2)), 'max_depth': [10,100,1000]}
# CV_rfc = GridSearchCV(rfc, param_grid)
# CV_rfc.fit(X_train, np.ravel(y_train,order='C'))
# CV_rfc.best_params_
```

From code cell output above:
{'max_depth': 100, 'max_features': 2, 'n_estimators': 1000}

With that optimized combination of parameters, let's re-build the model:

In [625]:
```
rfc = RandomForestClassifier(n_estimators=1000, max_depth=100, max_features=2, n_jobs=-1)
rfc.fit(X_train, np.ravel(y_train,order='C'))
predictions = rfc.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
```

Out[625]:
```
0.7411961630077899
```

In [626]:
```
# Which features play the biggest role in predicting campaign success:

rfc_features = sorted(list(zip(test_features, rfc.feature_importances_)), key=lambda x: x[1], reverse=True)
rfc_features
```

Out[626]:
```
[('usd_goal_real', 0.1632654172975585),
 ('Days_spent_making_campign', 0.13074576873369936),
 ('Sub_Cat_Nums', 0.12433641358038665),
 ('City_Pop', 0.09422942016052813),
 ('Length_of_kick', 0.08990126082289077),
 ('City-Nums', 0.0883109927562802),
 ('month_launched', 0.08722720899469728),
 ('Main_Cat_Nums', 0.07667362892926581),
 ('staff_pick', 0.07654743030687486),
 ('State-Nums', 0.06876245841781886)]
```

In [623]:
```
# Verify accuracy score through cross-validation of X and y data

rfc = RandomForestClassifier(n_estimators=1000, max_depth=100, max_features=2, n_jobs=-1)
scores = cross_val_score(rfc, X, np.ravel(y,order='C'), cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

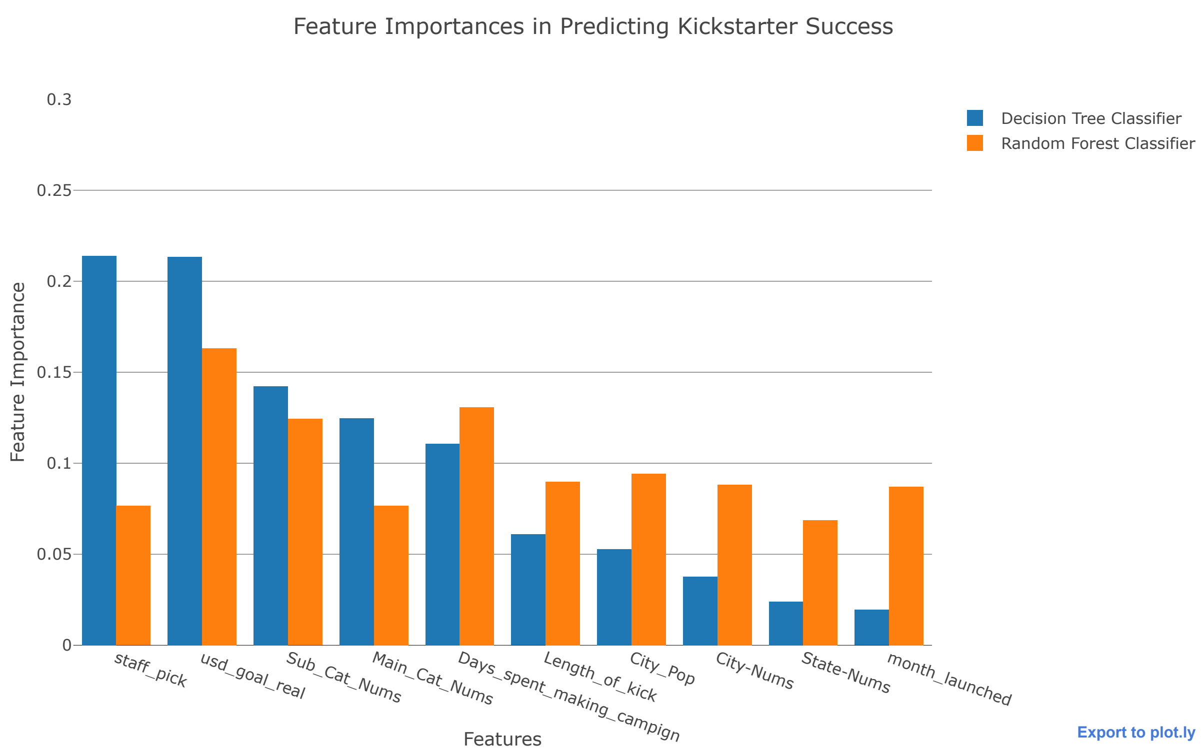
```
Accuracy: 0.74 (+/- 0.01)
```

The Random Forest approach brings our prediction accuracy up 2%, to **74%**.
Interestingly, notice that the feature importances differs between the DTC and RFC. Let's take a look in a graph:

In [635]:
```
data = [
    go.Bar(
        x=[x[0] for x in dtc_features],
        y=[x[1] for x in dtc_features],
        name='Decision Tree Classifier'
    ),
    go.Bar(
        x=[x[0] for x in rfc_features],
        y=[x[1] for x in rfc_features],
        name='Random Forest Classifier'
    )
]

layout = go.Layout(
    title='Feature Importances in Predicting Kickstarter Success',
    yaxis={'title': 'Feature Importance', 'range': [0,.3]},
    xaxis={'title': 'Features', 'tickangle': 20},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='feat-importances')
```



Export to plot.ly »

I am not going to include, in this report, a study into why the feature importances vary between the two classifiers, but needless to say it is something to examine going forward.

However, taking into account both results, it seems clear that the **monetary goal** plays a large role in the success of a campaign, as does the **sub-category** and **days spent making campaign**. *Staff pick* remains inconclusive for now; it seems odd that, when 84% of staff picks get funded, it would have such a small feature importance in the Random Forest Classifier.

Next Steps

There are additional data features that I would like to include in the future to move this study to the next level:

- Suggested pledge amounts
  - Number of different suggestions
  - Pledge suggestion as percent of goal
- Rewards offered / reward thresholds (not sure the best way to quantify/normalize this)
- Campaign page contents (videos, photos, charts, graphics, description, etc.....also not sure how to quantify/measure this)

From Kickstarter blog:
[https://www.kickstarter.com/blog/trends-in-pricing-and-duration (https://www.kickstarter.com/blog/trends-in-pricing-and-duration)](https://www.kickstarter.com/blog/trends-in-pricing-and-duration)