KICK STARTER

Predicting Campaign Success

Brian McMahon

21 February 2018

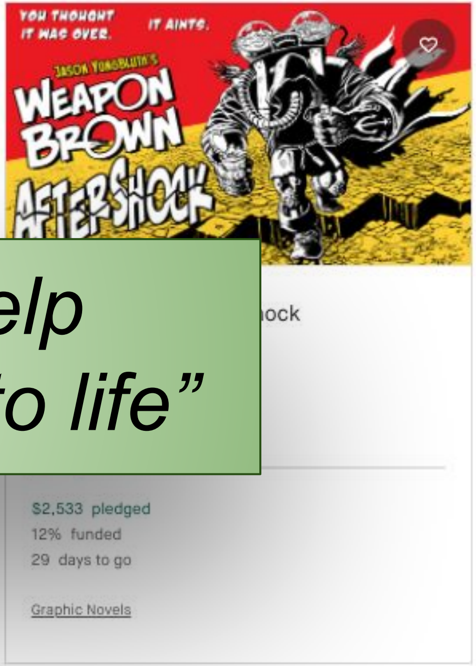
What is Kickstarter?

Leading crowdfunding platform, raising small amounts of money from large amounts of people

Since 2009:

- $3.5 bi
- 139,000+ projects funded
- 14 million project backers

"Our mission is to help bring creative projects to life"



Example Campaign

Model Overview

- Dataset scraped from Kickstarter*
- From January 2016 to today
- 8 -> ~160 features (using “one-hot” dummy variables)
- Analysis:
  - Initial screen of several models
    - *GridSearchCV to optimize parameters*
  - Deep dive on two high performers

** Kickstarter scraped dataset courtesy of webrobots.io.*

Dataset Features

X Features
Category
Subcategory
Staff Pick
Goal (US$)
Country
Currency
Campaign Length (Days)
Blurb Length

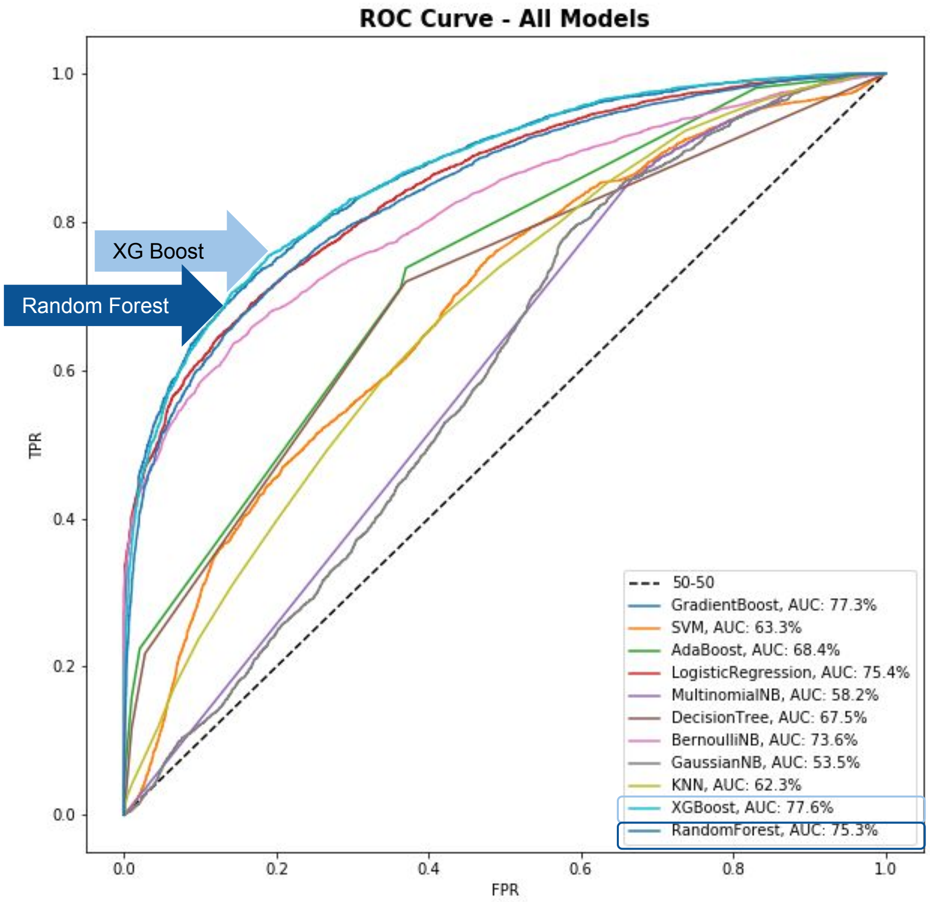
Y Targets
Backers Count
Pledged (US$)
State (“Success”, “Fail”)

Categorical; our supervised ML model response

KICKSTARTER

ROC Curve: All Models

	accuracy	auc	f1_f	f1_s	precision_f	precision_s	recall_f	recall_s
XGBoost	77.6%	77.6%	75.9%	79.1%	73.9%	81.0%	78.0%	77.3%
GradientBoost	77.2%	77.3%	75.6%	78.7%	73.3%	80.9%	78.0%	76.6%
LogisticRegression	75.3%	75.4%	73.5%	76.9%	71.3%	79.0%	75.9%	74.9%
RandomForest	75.5%	75.3%	73.0%	77.5%	72.6%	77.9%	73.4%	77.2%
BernoulliNB	73.3%	73.6%	72.1%	74.3%	68.1%	78.6%	76.7%	70.5%
AdaBoost	68.9%	68.4%	64.7%	72.2%	66.4%	70.8%	63.0%	73.8%
DecisionTree	67.9%	67.5%	63.9%	71.1%	64.9%	70.3%	63.0%	71.9%
SVM	64.7%	63.3%	55.8%	70.6%	64.1%	65.0%	49.4%	77.2%
KNN	63.4%	62.3%	55.5%	69.0%	61.6%	64.5%	50.6%	74.0%
MultinomialNB	61.4%	58.2%	37.4%	72.1%	69.7%	59.7%	25.5%	90.9%
GaussianNB	57.9%	53.5%	15.4%	72.0%	82.5%	56.7%	8.5%	98.5%



Business Case: What we care about

Campaign Creator

Cares about: **Precision**
(low False Positive, Type I Error)

- Decrease risk of creator's campaign in fact failing *(when they believed it would succeed)*
- Cost: campaign investment

Campaign Backer

Cares about: **Recall**
(low False Negative, Type II Error)

- Decrease risk of backer missing a success
- Cost: missed success

∴ Maximize both precision and recall (with F1 Score)

Random Forests

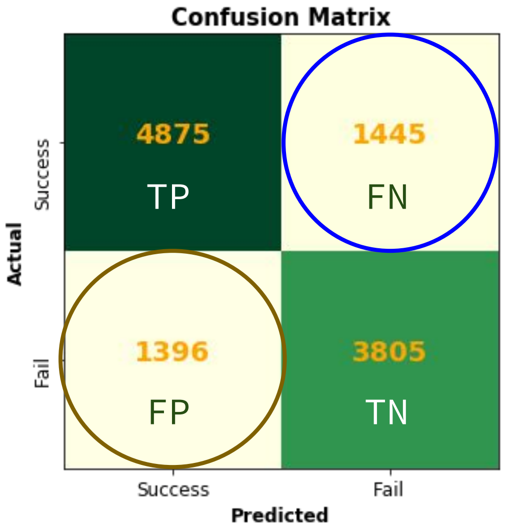
	accuracy	auc	f1_f	f1_s	precision_f	precision_s	recall_f	recall_s
RandomForest	75.4%	75.2%	72.9%	77.4%	72.4%	77.8%	73.3%	77.1%

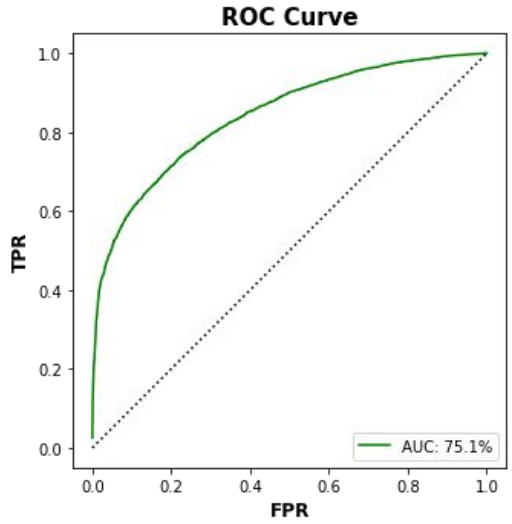
High Precision **Creator**

High Recall **Backer**

High F1 ✔

High AUC ✔





Random Forests: Feature Importance

importances	features
0.243471	usd_goal
0.154323	blurb_length
0.119610	campaign_length
0.073467	staff_pick
0.015714	category_name_Apparel
0.013365	category_name_Children's Books
0.012575	category_name_Video Games
0.012474	category_main_food
0.009060	category_name_Nonfiction
0.008936	category_name_Illustration

Keys to campaign success

Category matters!

XG Boost

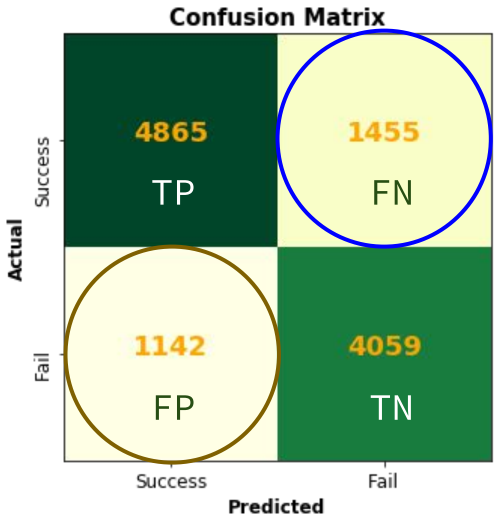
	accuracy	auc	f1_f	f1_s	precision_f	precision_s	recall_f	recall_s
		✔		✔	✔			✔
XGBoost	77.5%	77.5%	75.8%	78.9%	73.6%	81.0%	78.0%	77.0%

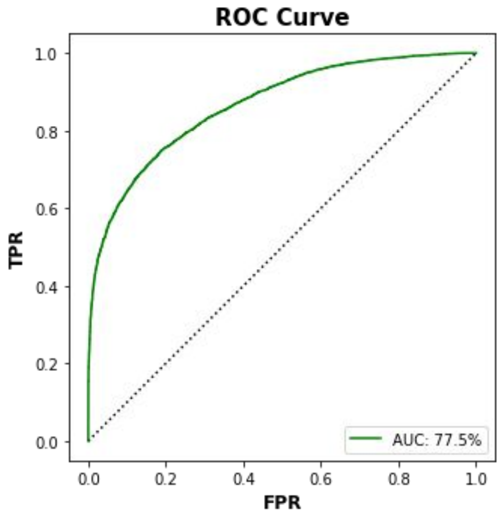
High Precision **Creator**

High Recall **Backer**

High F1 ✔

High AUC ✔





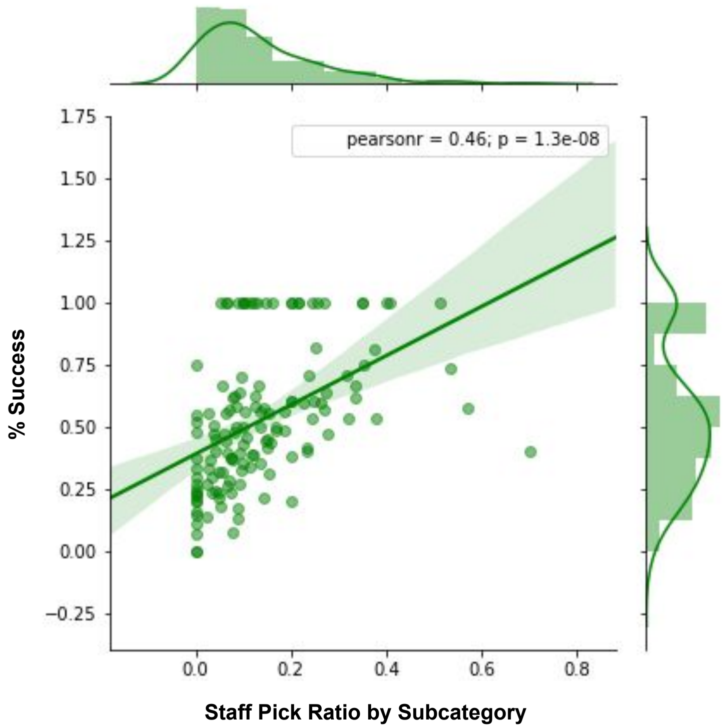
Feature Exploration

160 subcategories plotted to determine **whether a recommendation by kickstarter staff ("staff pick") affects the success of a campaign***

- x axis: staff picks as % of subcategory total
- y axis: success rate as % of subcategory total

Click on chart to visualize via Flask / D3 web app…

Staff Pick vs. Success Rate by Subcategory



** data from July 2017 to today*

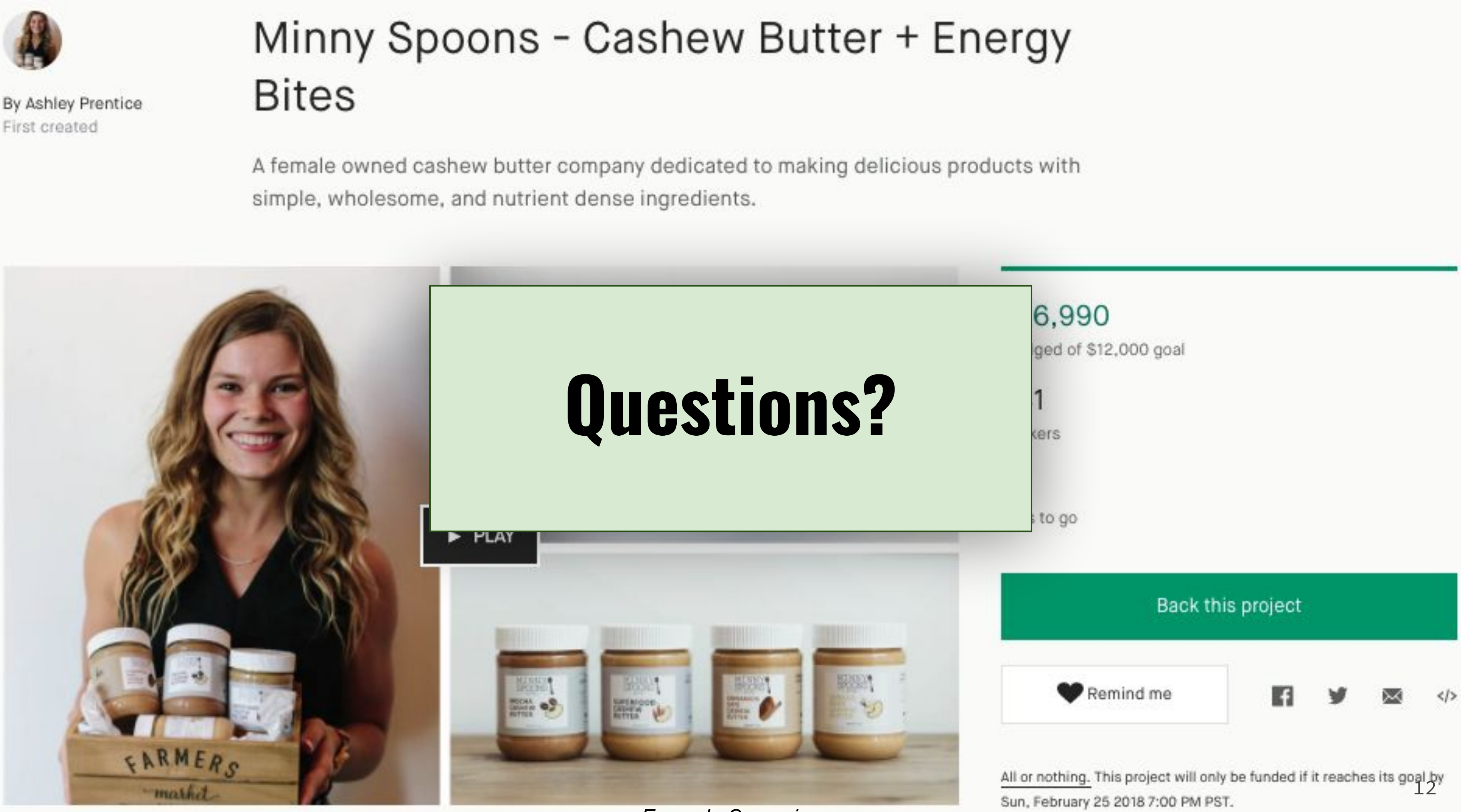
Key Takeaways

Remember!

When planning a Kickstarter Campaign:

1. Set a **low US$ goal**
2. Be **recommended** by staff
3. Make your **campaign short**
4. **Concise** description
5. Category **matters**

Example Campaign



Questions?

Example Campaign

APPENDIX

Agenda

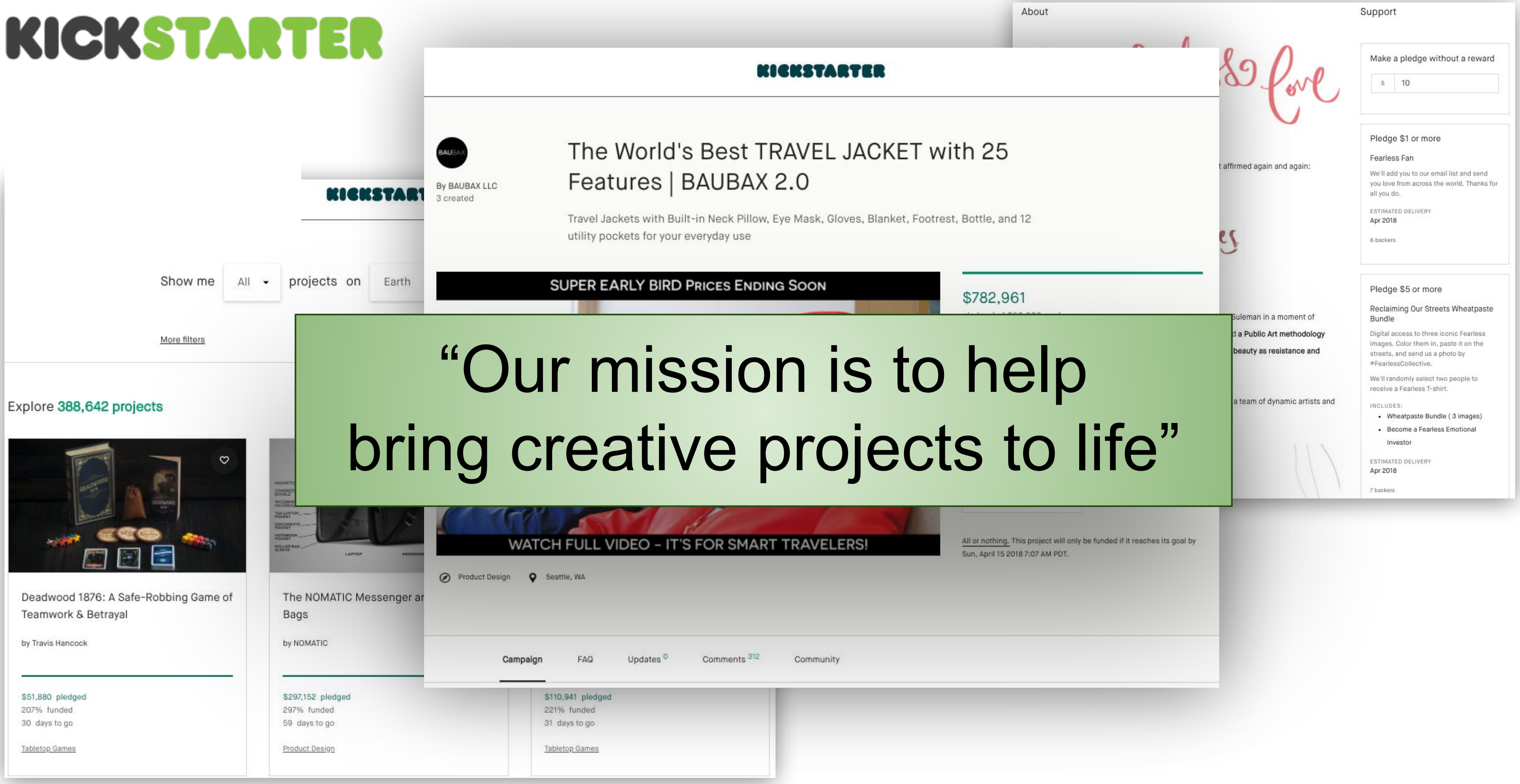
Intro to Kickstarter

Prediction Model Overview

Data Exploration
with Flask/D3 visualization

Key Takeaways

Example Campaign



"Our mission is to help bring creative projects to life"

Model Methodology

- Dataset scraped from Kickstarter website*
- 160k -> 40k rows, from January 2016 to January 2018
- 8 -> ~160 features (with “one-hot” dummy variables)
- Model refinement procedure:
  - Initial screen, several ML models preliminarily run against data
    - Preprocessing: StandardScaler for SVM, KNN, LogisticRegression
    - *Utilize GridSearchCV to optimize parameters*
  - Deep dive on highest performing models: Random Forests and XG Boost

** Kickstarter scraped dataset courtesy of webrobots.io.*

Preprocessing

State to Success, fail (dropped live, suspended, cancelled)

StandardScaler for SVM, KNN, Logistic Regression

It all starts with SQL...

```
Select * from kickstarter_data2;
```

Dataset Features

id	163425	non-null	int64	Unique campaign ID
name	163424	non-null	object	Project Name
state	163425	non-null	object	*Response, categorical.* **Filtered to "Success" or "Fail"**
category_main	163425	non-null	object	**15 main categories**
category_name	163425	non-null	object	**~160 subcategories**
backers_count	163425	non-null	int64	*Response, numerical.* **# backers**
pct_goal_achieved	147802	non-null	float64	*Response, numerical.* US$ pledged / US$ goal
usd_pledged	163425	non-null	float64	*Response, numerical.* **US$ pledged**
usd_goal	147802	non-null	float64	**US$ goal**
country	163425	non-null	object	**Country**
currency	163425	non-null	object	**Currency**
campaign_length	163425	non-null	int64	**Length of campaign (days)**
deadline	163425	non-null	object	Project end data
launched	163425	non-null	object	Project launch date
created	163425	non-null	object	Project creation date
staff_pick	163425	non-null	int64	**Recommended by Kickstarter staff**
creator_name	163425	non-null	object	Name of creator
blurb_length	163425	non-null	int64	**Length of intro blurb**

Y Value

X Values

Features

Green: Independent Variables

Orange: Response Variables

- State: Categorical
  - Success/Fail defined as pledged > goal
- USD Pledged: Numerical
- # Backers: Numerical

*We will predict **State***

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 163425 entries, 543 to 980
Data columns (total 18 columns):
id                    163425 non-null int64
name                  163424 non-null object
state                 163425 non-null object
category_main         163425 non-null object
category_name         163425 non-null object
backers_count         163425 non-null int64
pct_goal_achieved     147802 non-null float64
usd_pledged           163425 non-null float64
usd_goal              147802 non-null float64
country               163425 non-null object
currency              163425 non-null object
campaign_length       163425 non-null int64
deadline              163425 non-null object
launched              163425 non-null object
created               163425 non-null object
staff_pick            163425 non-null int64
creator_name          163425 non-null object
blurb_length          163425 non-null int64
dtypes: float64(3), int64(5), object(10)
memory usage: 23.7+ MB
```

Dataset Stats

	Dataset 1	Dataset 2
Source	Kaggle	WebRobots.io
Initial Shape	(378661, 17)	(192716, 37)
Key features	Category, subcategory, country, currency, goal, campaign length, pledged*, backers*	Category, subcategory, country, currency, goal, **staff pick**, campaign length, **blurb length**, pledged*, backers*
Time period	2012 - Present	2016 - Present
** removed from prediction dataset*		

KICKSTARTER

MVP TODO

\# observations, # features/variables

ROC chart add AUC value in legend

Candlestick for pledges, backers by success/fail

Easy nuggets with probability - key takeaways

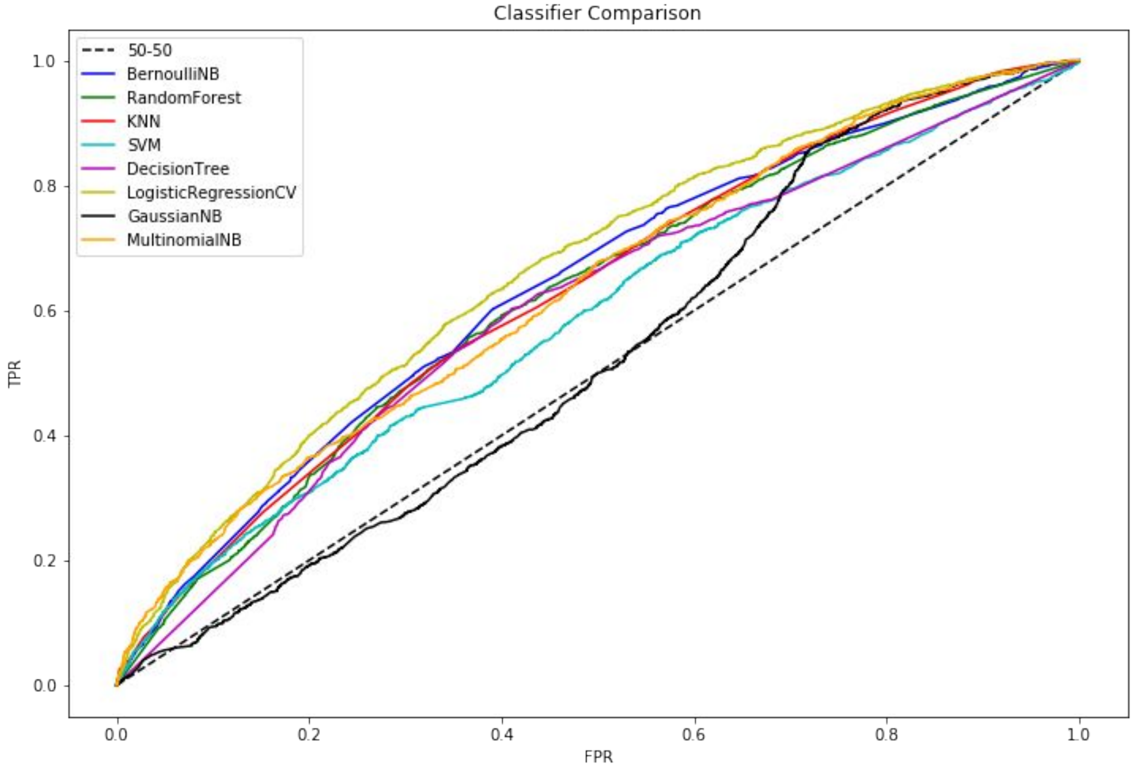
Logit / LR models provide log odds

ROCO *Performance stagnates at AUC of ~0.60*

Initial dataset from Kaggle

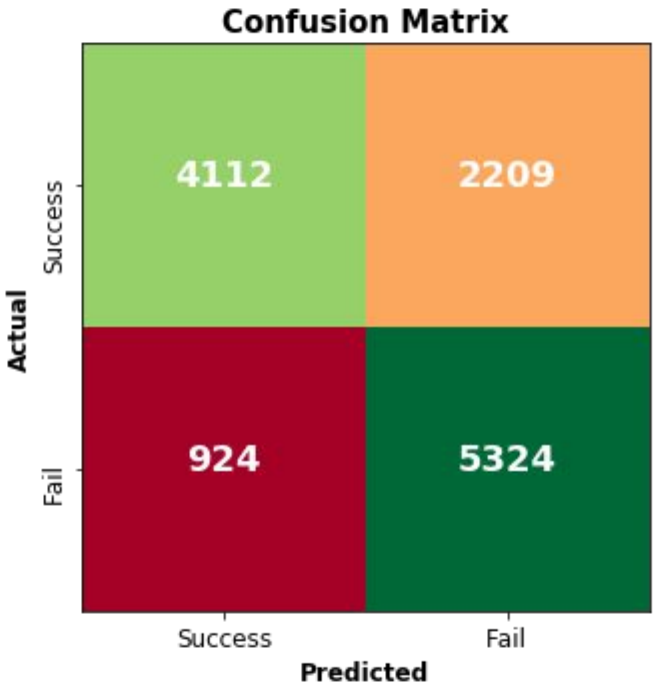
Limited predictive features:

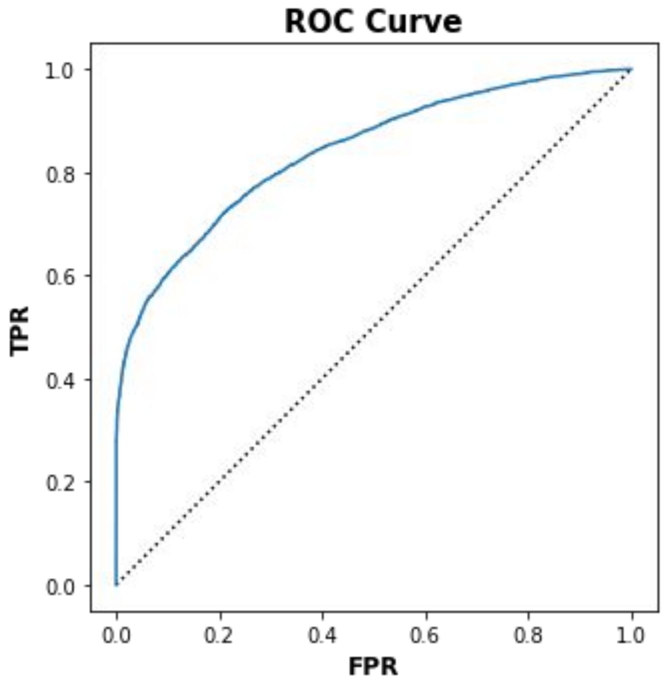
- Category
- Country
- Currency
- Campaign Length
- Goal Amount



KICKSTARTER

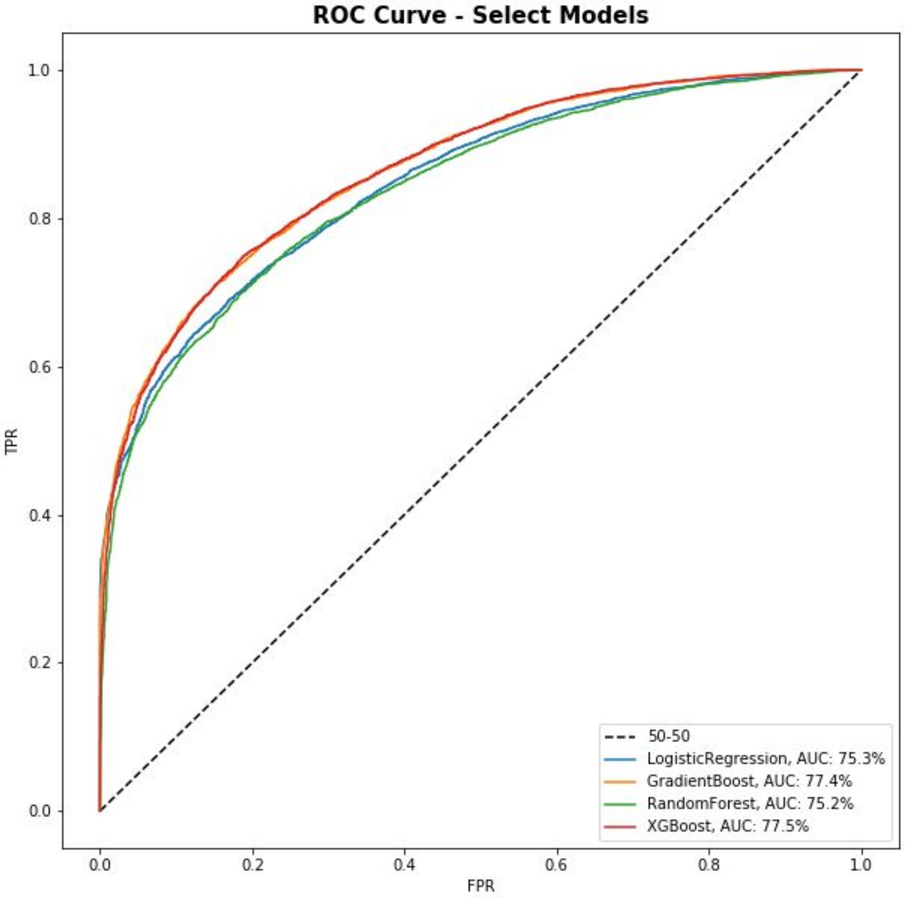
Gradient Boost





ROC Curve: Select Models

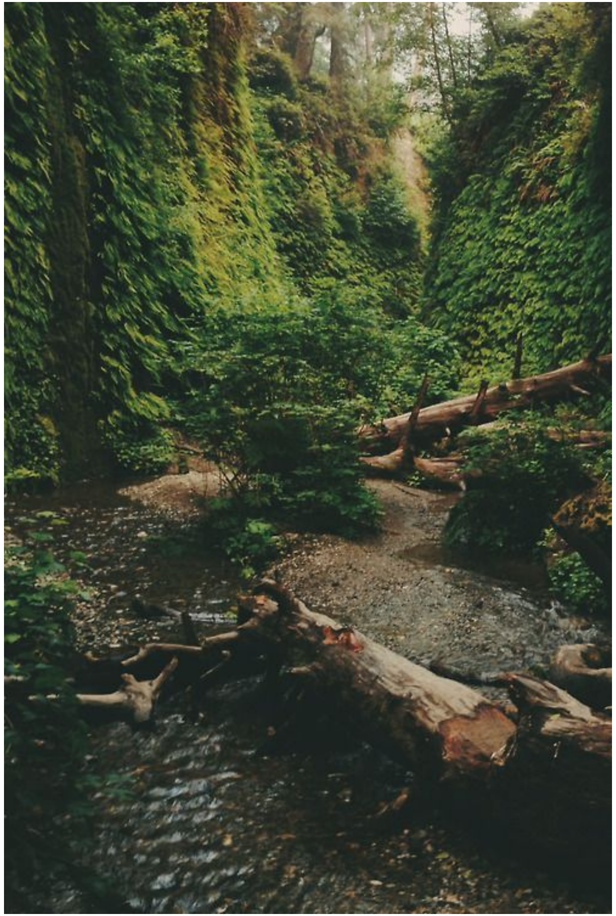
[insert dict]



KICKSTARTER

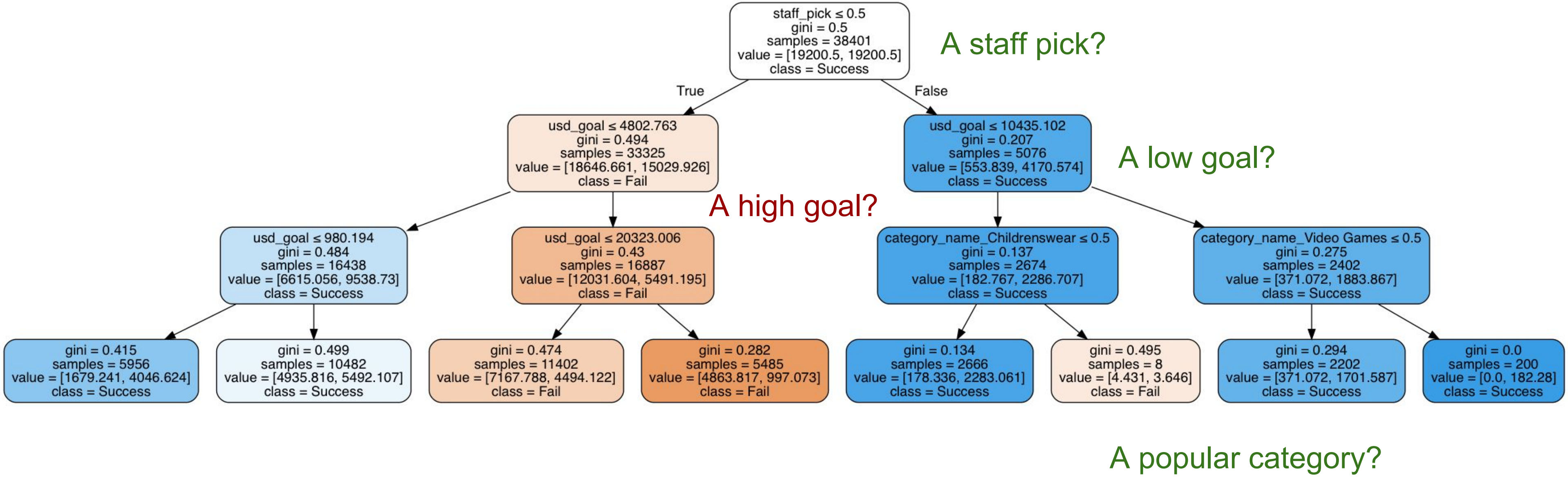
Random Forests: What it is

- Ensemble learning method
- Trains on a multitude of decision trees
- Outputs class that is the mean/mode of individual trees
- Corrects overfitting, a weakness of individual decision trees



A random forest..

Random Forests: Decision Tree



XG Boost: What it is

- “Extreme Gradient Boosting” is a gradient boosting framework proposed in Friedman’s *“Greedy Function Approximation: A Gradient Boosting Machine”*
- Designed and optimized for boosted tree algorithms
- Popular, highly competitive algorithm for ML competitions

ROC1

Performance up to AUC of ~0.76

New dataset from WebRobots

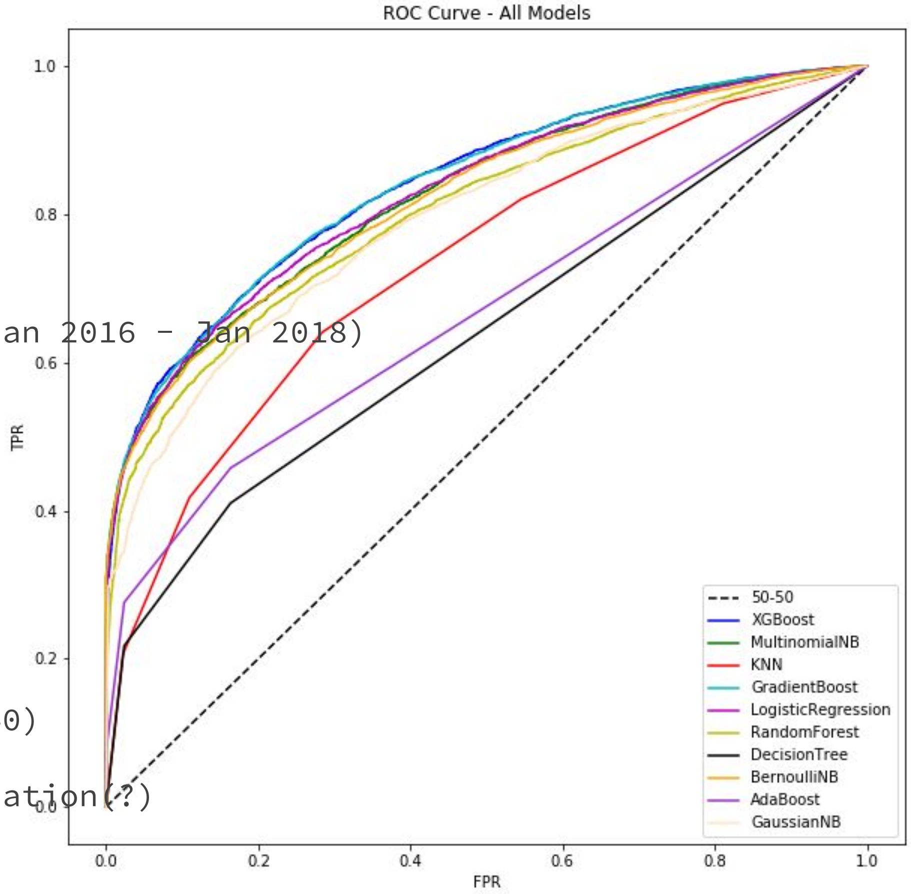
- ~200,000 -> 40,000 datapoints (Jan 2016 - Jan 2018)

Added predictive features:

- Staff pick
- Blurb length

Additional modifications:

- Main category (15) -> sub category (140)
- Parameter optimization (GridSearchCV)
- Removed currency due to country correlation(?)

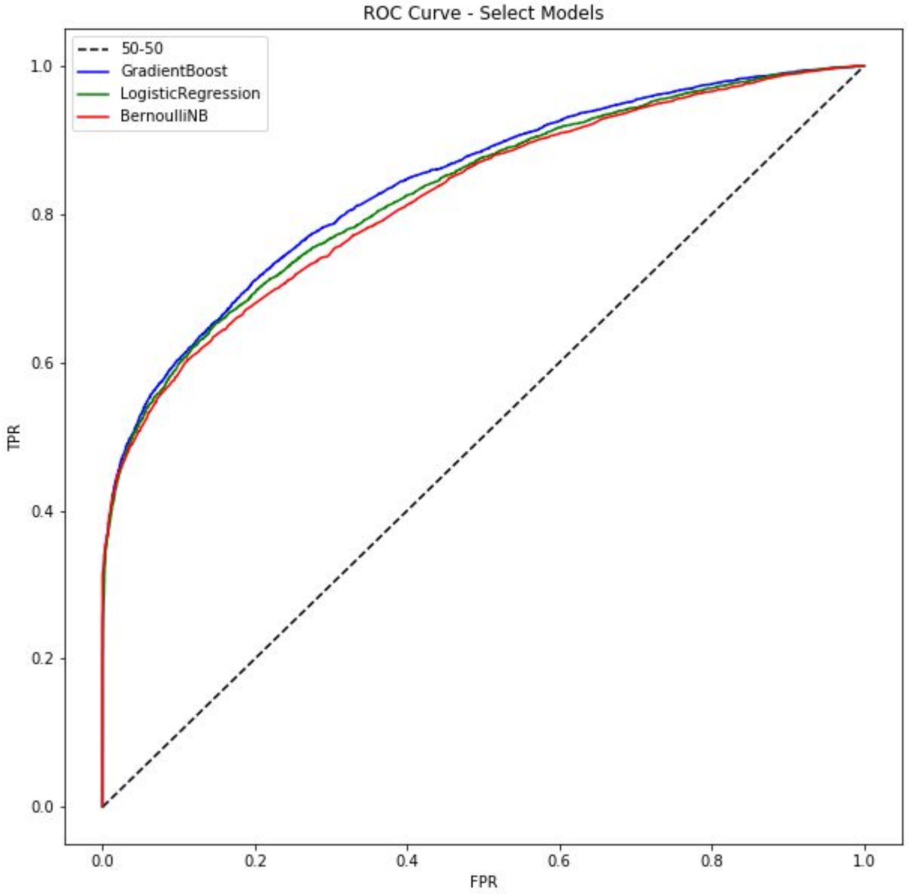


KICKSTARTER

ROC2

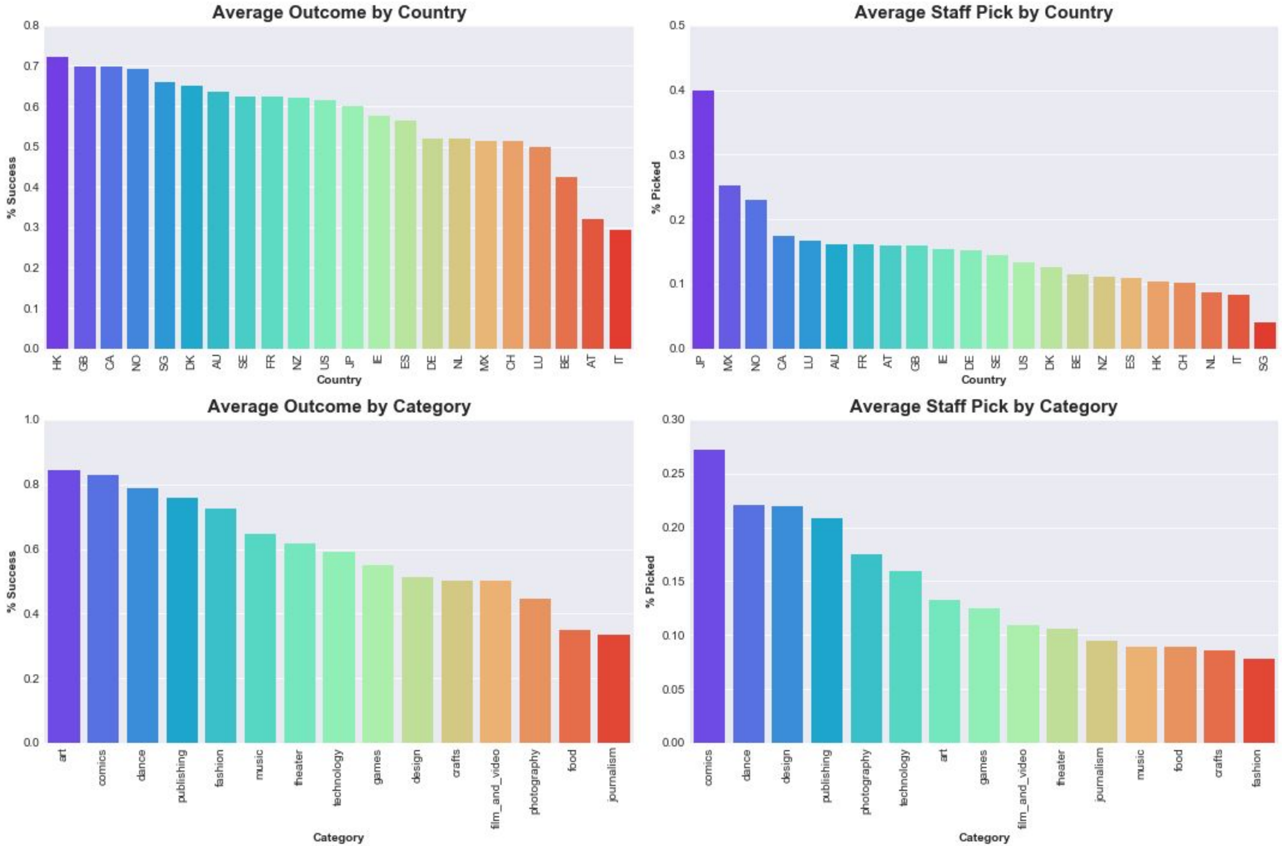
Best performing tests include:

- Gradient Boost
- Logistic Regression
- Naive Bayes (Bernoulli)

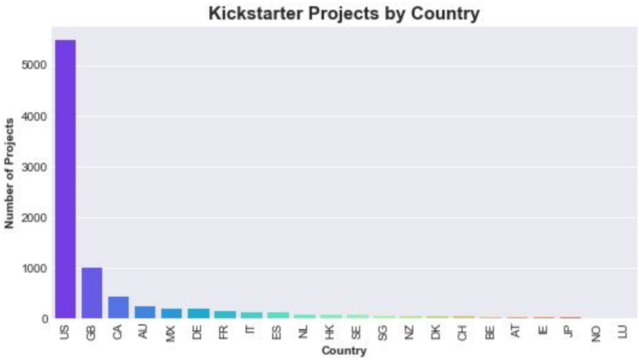


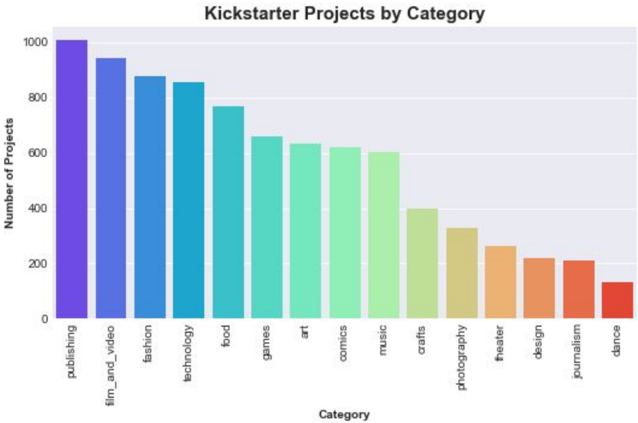
KICKSTARTER

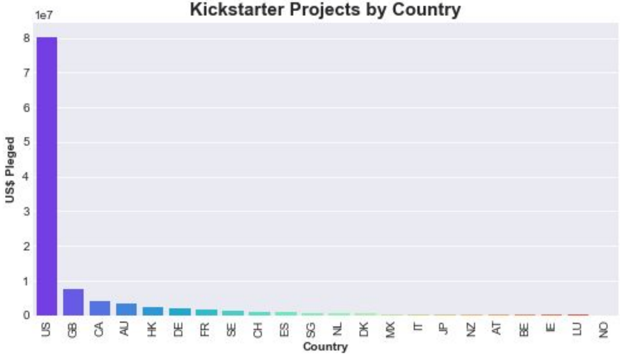
Feature Exploration

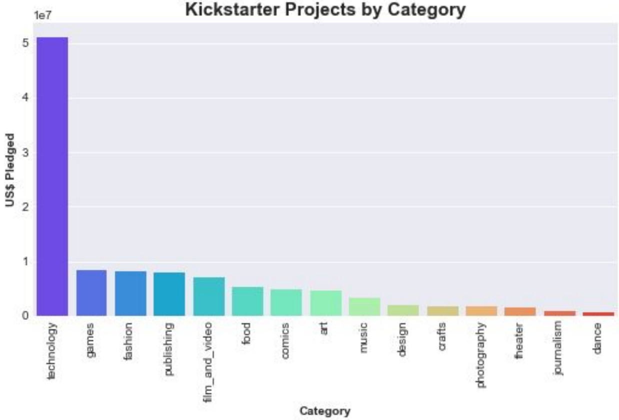


Feature Exploration





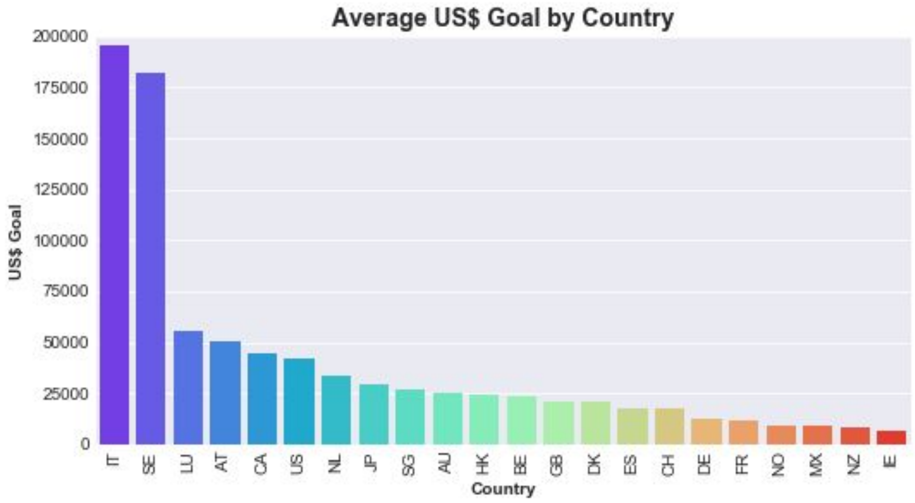


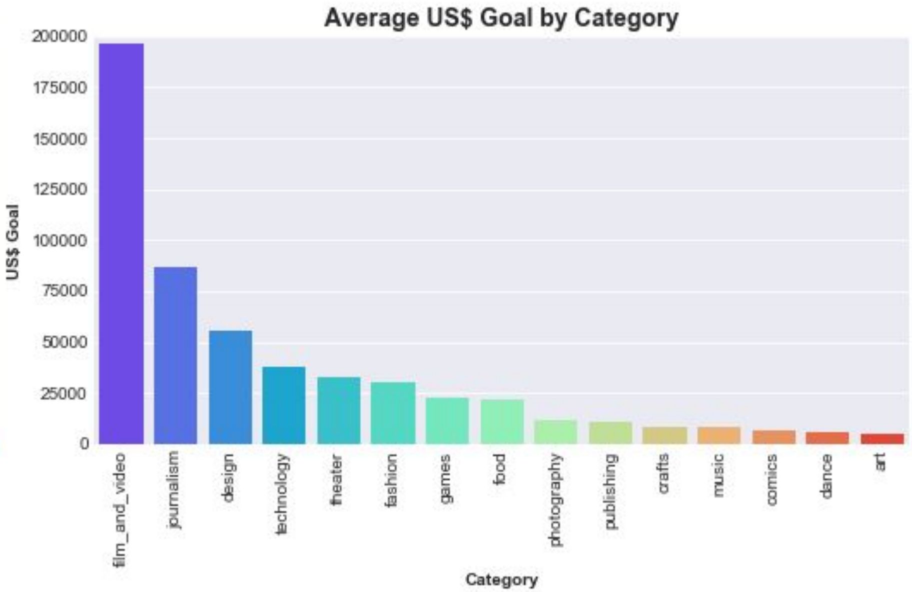


KICKSTARTER

Feature Exploration

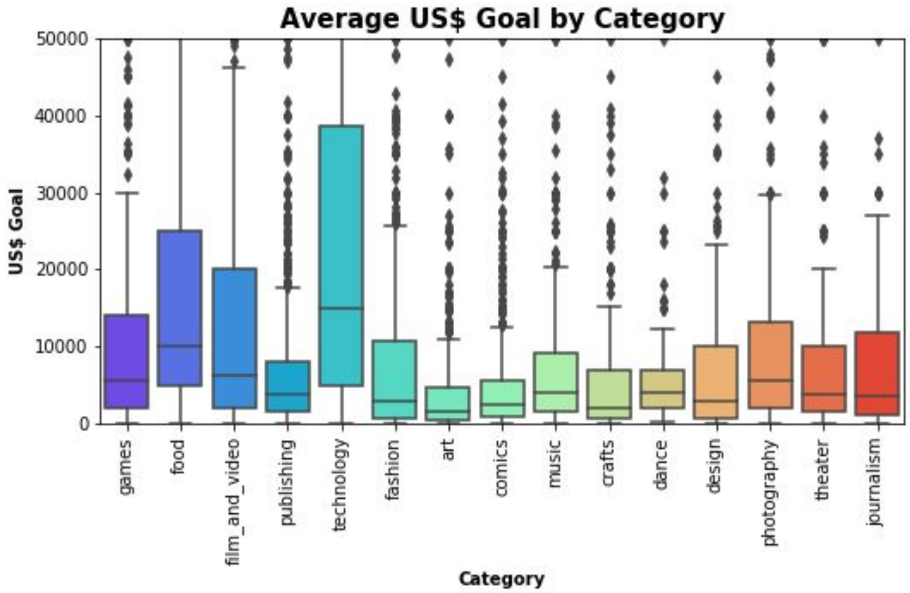
Lower goals typically lead to greater success

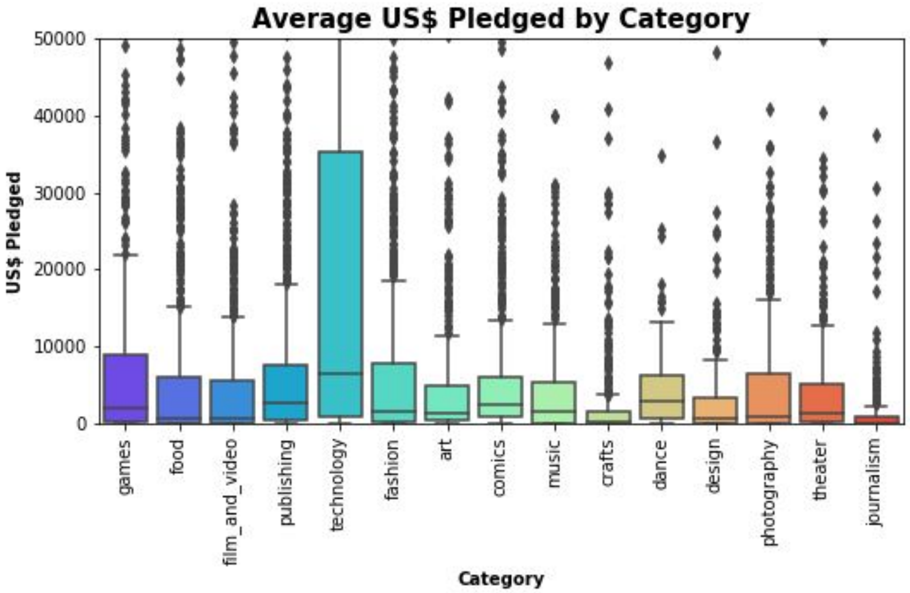




KICKSTARTER

Feature Exploration

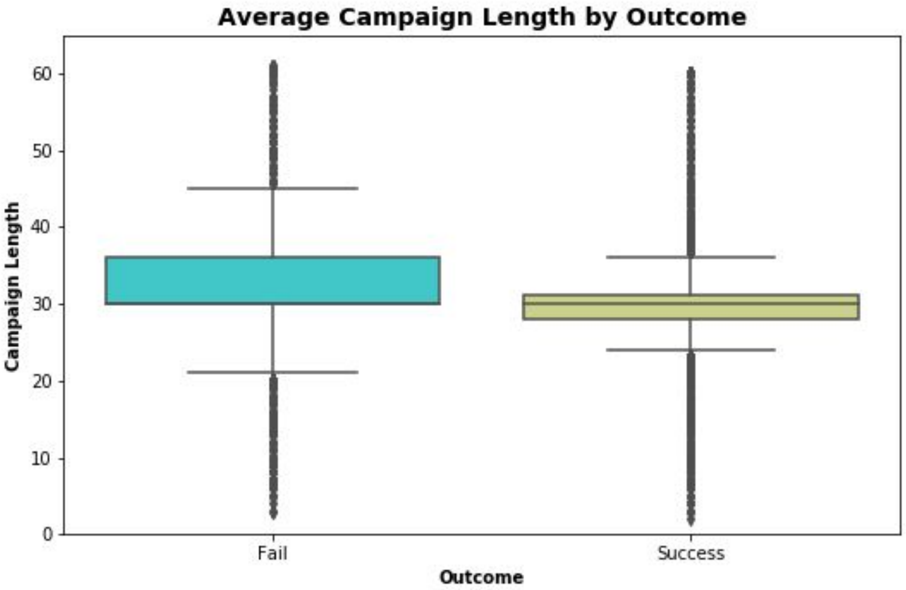


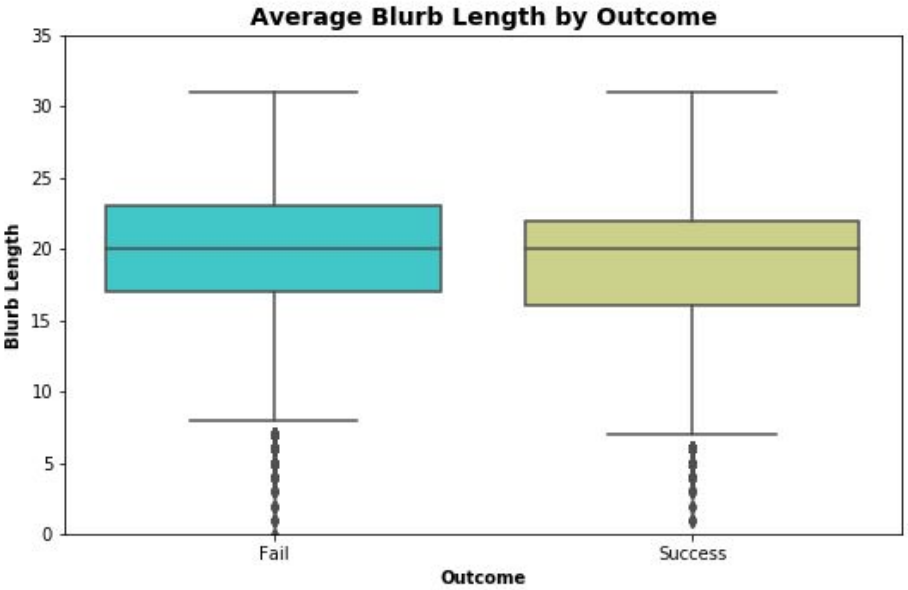


KICKSTARTER

Feature Exploration

Successful campaigns are, on average, shorter in duration with concise descriptions





KICKSTARTER

Process / Challenges

Multiple datasets analyzed - not all datasets come equal!

Had to expand set of features to obtain adequate results

Initial screen with first data set narrowed features to only category, country, currency and goal value

Preprocessing: dummy variables, standardizing, train/test split and cross-validate, removed all but successful and failed (such as live, cancelled)

Initial pass through all models, generating initial ROC curve

Deeper analysis into RF and LR, reasons being [expand]

Next Steps

Integrate twitter analysis of launched projects

Incorporate numerical US$ pledge prediction ie with linear regression