Part III. Data Model Construction and Prediction

April 17, 2019

1 Predicting Crowdfunding Success

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

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

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

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

        from sklearn.naive_bayes import GaussianNB

        %matplotlib inline
```

2 Data Cleaning & Processing

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

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

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

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

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

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

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

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

[1 rows x 41 columns]
```

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

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

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

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

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

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

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

Out[6]:

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

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

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

In [7]:
```
df['state'] = df.state.str.contains('successful').astype(int)
```

In [8]:
```
# add column representing continent
def classifier(row):
    if row.country in ['US', 'CA', 'GT', 'MX', 'PR', 'NI', 'SV', 'PA', 'BO', 'GU']:
        return 'America'
    elif row.country in ['NG', 'GH', 'ZA', 'KE', 'ET', 'CD', 'MA', 'TZ', 'ZM', 'LR', 'I
        return 'Africa'
    elif row.country in ['GB', 'NO', 'DE', 'SE', 'BA', 'IS', 'HU', 'IT', 'NL','FR', 'UI
       'TR','FI', 'CZ','AM', 'PT','DK','CH', 'SJ', 'RU', 'UA', 'BG','ES','PL', 'GE','IE
        return 'Europe'
    elif row.country in ['JM', 'HT','BS','DO','LC', 'DO', 'TT']:
        return 'Carribean'
    elif row.country in ['CN', 'TW', 'HK', 'NP', 'ID', 'SG', 'IN', 'JP', 'LB', 'KZ', 'I
        return 'Asia'
    elif row.country in ['IL','QA', 'AF','KZ','AE','PS','SY','SA', 'IQ','IR','TJ',]:
        return 'Arab'
    else:
        return "Oceania"
df["continent"] = df.apply(classifier, axis=1)
```

In [9]:
```
df.head()
```

Out[9]:

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

```
4             20     Software                        549      US      USD

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 kNN Model

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

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

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

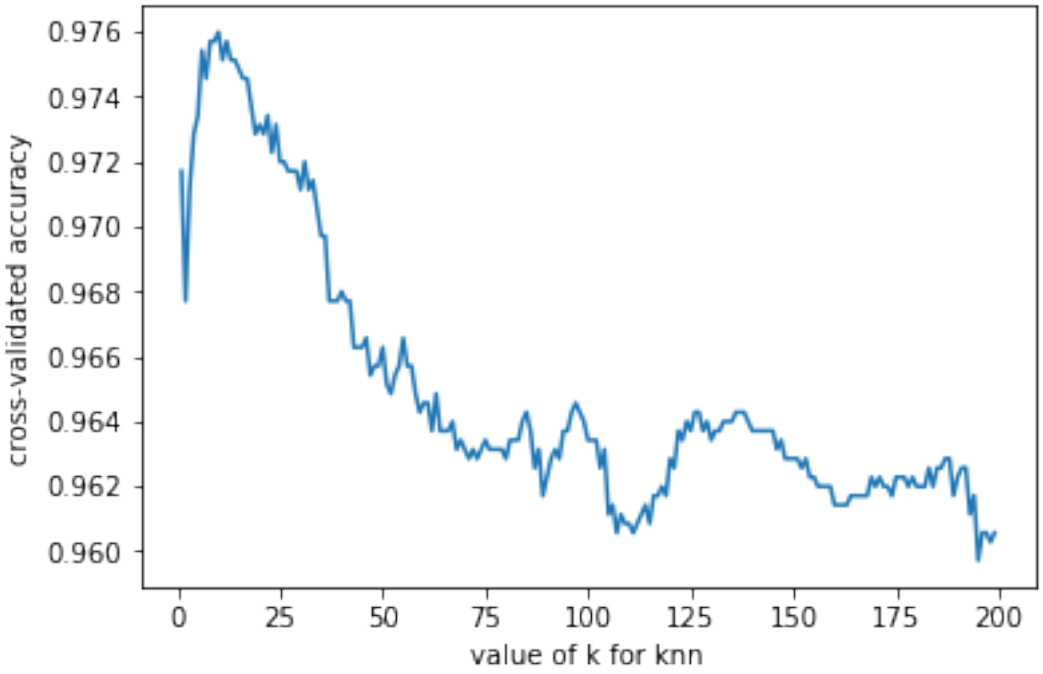
```
Out[15]: 0.9605670786816919
```

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

```
Out[16]: 0.9885386819484241
```

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

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



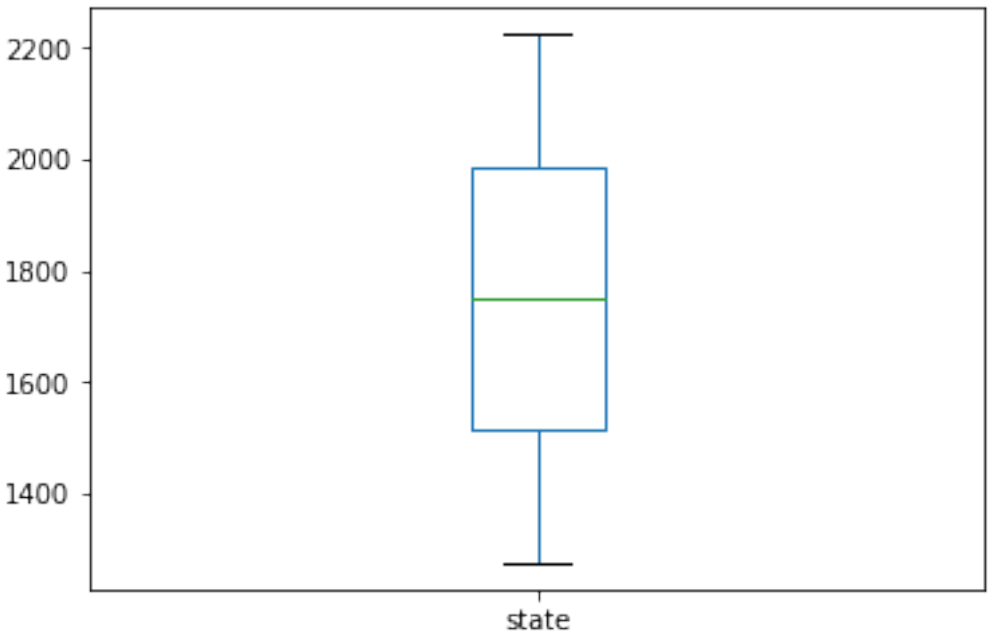

```
In [17]: MSE = [1 - x for x in k_scores]
         optimal_k = k_range[MSE.index(min(MSE))]
         print("The optimal number of neighbors is %d" % optimal_k)
```

```
The optimal number of neighbors is 10
```

4 Logistic Regression Model

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

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




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

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

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

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

Out[21]: 0.838

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

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

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

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

Out[24]: 0.8133333333333334

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

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

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

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

          0       0.77      0.72      0.74       392
          1       0.84      0.87      0.85       658

avg / total       0.81      0.81      0.81      1050
```

5 Support Vector Machine

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

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

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

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

          0       1.00      1.00      1.00       386
          1       1.00      1.00      1.00       664

avg / total       1.00      1.00      1.00      1050
```

6 Naive Bayes

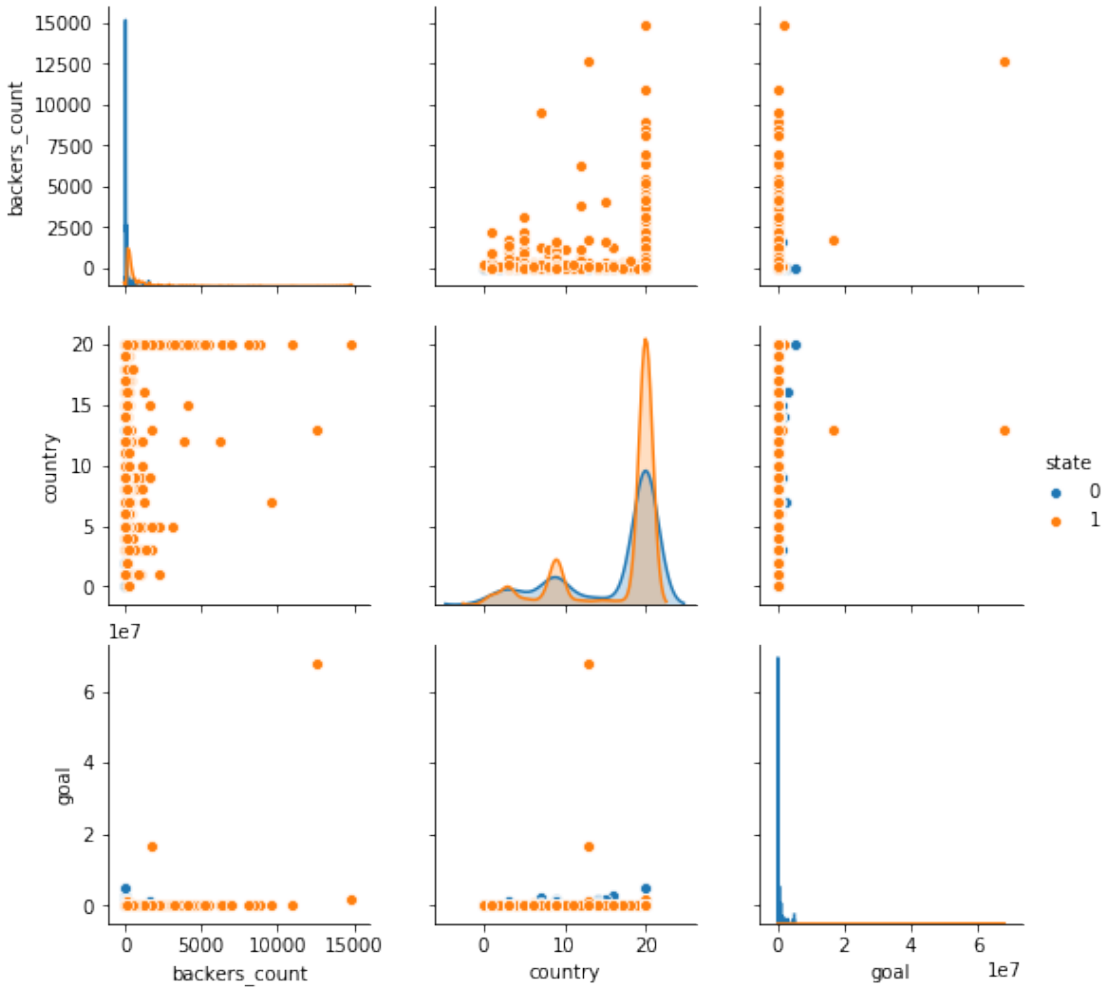
```
In [28]: gnb = GaussianNB()
         y_pred = gnb.fit(X_train, y_train).predict(X_test)

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

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

          0       0.44      0.98      0.61       392
          1       0.95      0.27      0.42       658

avg / total       0.76      0.54      0.49      1050
```

7 Data Visualization

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

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



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

Out[34]:

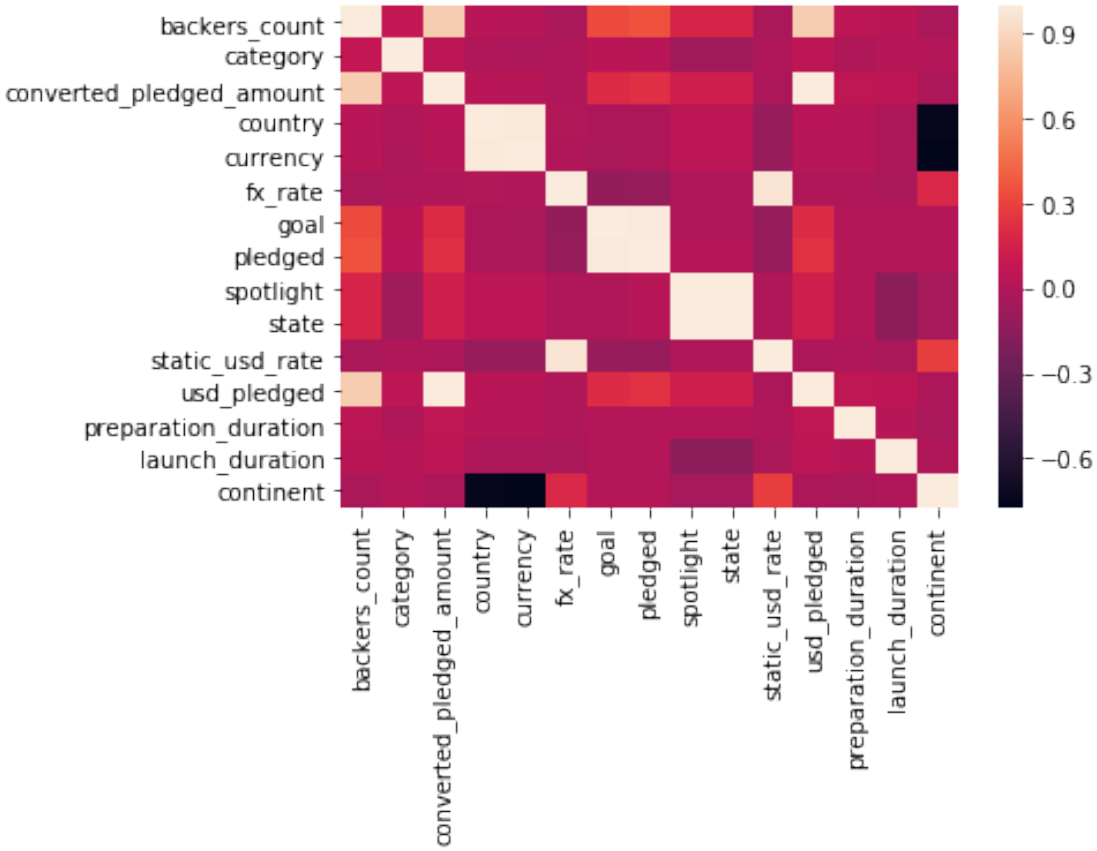
	backers_count	category	converted_pledged_amount	\
backers_count	1.000000	0.074279	0.855609	
category	0.074279	1.000000	0.056564	
converted_pledged_amount	0.855609	0.056564	1.000000	
country	0.031723	-0.007631	0.031882	
currency	0.027896	-0.011621	0.027261	
fx_rate	-0.023226	-0.007235	-0.008231	
goal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

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

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

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

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

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

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

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



8 Citation

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

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