Predicting success of a Startup

When the entrepreneurs begin with their startup, they think they are buiding the next big company! But the reality is 90% of the startups fail and most of them fail because lack of idea about some factors affecting the success in their startup. In this exercise we will predict which startup is going to be successful and which startup is going to fail. Let's see if your startup is the next big thing or not.

The Dataset

Kickstarter is one of the main online crowdfunding platforms in the world. The dataset provided contains more than 300,000 projects launched on the platform in 2018. In the data file there are the following columns:

- **ID**: internal ID, *numeric*
- **name**: name of the project, *string*
- **category**: project's category, *string*
- **main_category**: campaign's category, *string*
- **currency**: project's currency, *string*
- **deadline**: project's deadline date, *timestamp*
- **goal**: fundraising goal, *numeric*
- **launched**: project's start date, *timestamp*
- **pledged**: amount pledged by backers (project's currency), *numeric*
- **state**: project's current state, *string*; **this is what you have to predict**
- **backers**: amount of poeple that backed the project, *numeric*
- **country**: project's country, *string*
- **usd pledged**: amount pledged by backers converted to USD (conversion made by KS), *numeric*
- **usd_pledged_real**: amount pledged by backers converted to USD (conversion made by fixer.io api), *numeric*
- **usd_goal_real**: fundraising goal is USD, *numeric*

Goal

The goal is to predict whether a project will be successful or not.

Importing all the necessary libraries

In [85]:

```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Loading the dataset

In [86]:

```
df = pd.read_csv("/Users/saurabhkarambalkar/Desktop/Kickstarter/data.csv")
df.head(3)
```

Out[86]:

	ID	name	category	main_category	currency	deadline	goal	launched	pledg
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	2015-10-09	1000.0	2015-08-11 12:12:28	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
2	1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26	45000.0	2013-01-12 00:20:50	220

Understanding the dataset

In [87]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 378661 entries, 0 to 378660
Data columns (total 15 columns):
ID                  378661 non-null int64
name                378657 non-null object
category            378661 non-null object
main_category       378661 non-null object
currency            378661 non-null object
deadline            378661 non-null object
goal                378661 non-null float64
launched            378661 non-null object
pledged             378661 non-null float64
state               378661 non-null object
backers             378661 non-null int64
country             378661 non-null object
usd pledged         374864 non-null float64
usd_pledged_real    378661 non-null float64
usd_goal_real       378661 non-null float64
dtypes: float64(5), int64(2), object(8)
memory usage: 43.3+ MB
```

In [88]:

```
df.isnull().sum()
```

Out[88]:

```
ID                    0
name                  4
category              0
main_category         0
currency              0
deadline              0
goal                  0
launched              0
pledged               0
state                 0
backers               0
country               0
usd pledged        3797
usd_pledged_real      0
usd_goal_real         0
dtype: int64
```

In [89]:

```
df.describe()
```

Out[89]:

	ID	goal	pledged	backers	usd pledged	usd_pledged_rea
count	3.786610e+05	3.786610e+05	3.786610e+05	378661.000000	3.748640e+05	3.786610e+0
mean	1.074731e+09	4.908079e+04	9.682979e+03	105.617476	7.036729e+03	9.058924e+0
std	6.190862e+08	1.183391e+06	9.563601e+04	907.185035	7.863975e+04	9.097334e+0
min	5.971000e+03	1.000000e-02	0.000000e+00	0.000000	0.000000e+00	0.000000e+0
25%	5.382635e+08	2.000000e+03	3.000000e+01	2.000000	1.698000e+01	3.100000e+0
50%	1.075276e+09	5.200000e+03	6.200000e+02	12.000000	3.947200e+02	6.243300e+0
75%	1.610149e+09	1.600000e+04	4.076000e+03	56.000000	3.034090e+03	4.050000e+0
max	2.147476e+09	1.000000e+08	2.033899e+07	219382.000000	2.033899e+07	2.033899e+0

Dealing with the missing values

We observe that the column 'name' have 4 missing values and the column 'usd pledged' have 3797 missing values.

Missing values of the column 'name' can be filled using random characters but missing values of the column 'usd pledged' are very important for us to consider during training of the model.

Therefore we are assuming that the missing values of the column 'usd pledged' to be the mean of that same column and we will fill them.

In [90]:

```
df['name'] = df['name'].fillna("abc")
```

In [91]:

```
df['usd pledged'] = df['usd pledged'].fillna((df['usd pledged'].mean()))
```

Now, let's verify whether we have dealt with all the missing values or not.

In [92]:

```
df.isnull().sum()
```

Out[92]:

```
ID                  0
name                0
category            0
main_category       0
currency            0
deadline            0
goal                0
launched            0
pledged             0
state               0
backers             0
country             0
usd pledged         0
usd_pledged_real    0
usd_goal_real       0
dtype: int64
```

The columns 'category', 'currency' and, 'main category' have their data type as object. To fit machine learning models, we first need to extract categorical variables and convert them to numeric variables using "pd.get_dummies".

In [93]:

```
numeric_data_cat = pd.get_dummies(df['category'], prefix='category')
del df['category']
```

In [99]:

```
df1 = df.join(numeric_data_cat)
```

In [100]:

```
numeric_data_cur = pd.get_dummies(df1['currency'], prefix='currency')
del df1['currency']
```

In [101]:

```
df2 = df1.join(numeric_data_cur)
```

In [102]:

```
numeric_data_main_cat = pd.get_dummies(df2['main_category'], prefix='main_cat')
del df2['main_category']
```

In [103]:

```
final_data = df2.join(numeric_data_main_cat)
final_data.head(3)
```

Out[103]:

	ID	name	deadline	goal	launched	pledged	state	backers	country	usd pledged
0	1000002330	The Songs of Adelaide & Abullah	2015-10-09	1000.0	2015-08-11 12:12:28	0.0	failed	0	GB	0.0
1	1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	2017-11-01	30000.0	2017-09-02 04:43:57	2421.0	failed	15	US	100.0
2	1000004038	Where is Hank?	2013-02-26	45000.0	2013-01-12 00:20:50	220.0	failed	3	US	220.0

3 rows × 200 columns

Now we will prepare our dataset which needs to be feed into our machine learning model for prediction.

In [104]:

```
X = final_data.drop(['name','deadline','launched','state','country'],axis=1)
y = final_data['state']
```

Spliting the data into training and test sets

In [105]:

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30, random_st
```

Model Implementation

First we will use Logistic Regression Model

In [54]:

```
# Initiate the classifier and fit the model on the training set

logreg = LogisticRegression(multi_class='multinomial',solver='newton-cg').fit(X_trai
```

```
/anaconda/lib/python3.6/site-packages/sklearn/utils/optimize.py:203: C
onvergenceWarning: newton-cg failed to converge. Increase the number o
f iterations.
  "number of iterations.", ConvergenceWarning)
```

In [106]:

```
logreg
```

Out[106]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='multinomia
l',
          n_jobs=1, penalty='l2', random_state=None, solver='newton-c
g',
          tol=0.0001, verbose=0, warm_start=False)
```

Prediction of the target attribute 'state'

In [107]:

```
y_pred =logreg.predict(X_test)
y_pred
```

Out[107]:

```
array(['failed', 'failed', 'failed', ..., 'failed', 'successful',
       'failed'], dtype=object)
```

Calculating the accuracy score for the model

In [108]:

```
a_score_lgr = accuracy_score(y_test,y_pred)
a_score_lgr = a_score*100
print("Accuracy for training data by Logistic Regression Model is %f" %a_score_lgr)
```

```
Accuracy for training data by Logistic Regression Model is 85.020995
```

Now, we will use Random Forest Classifier

In [109]:

```
# Initiate the classifier and fit the model on the training set

clf = RandomForestClassifier().fit(X_train,np.ravel(y_train))
```

In [110]:

```
clf
```

Out[110]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Prediction of the target attribute 'state'

In [111]:

```
Y_pred_clf = clf.predict(X_test)
Y_pred_clf
```

Out[111]:

```
array(['failed', 'failed', 'failed', ..., 'failed', 'successful',
       'failed'], dtype=object)
```

Calculating the accuracy score for the model

In [113]:

```
a_score_clf = accuracy_score(y_test,Y_pred_clf)
a_score_clf = a_score_clf*100
print("Accuracy for training data by Random Forest Classifier is %f" %a_score_clf)
```

```
Accuracy for training data by Random Forest Classifier is 85.668008
```