DS_Hw04

October 13, 2019

[3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

import plotly.tools as tls
import plotly
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot, plot
import plotly.graph_objs as go
init_notebook_mode(connected=True)
```

[4]:
```
df_kick=pd.read_csv("./ks-projects.csv")
df_kick=df_kick.sample(10000,random_state=42).reset_index().drop('index',axis=1)
```

[5]:
```
def resumetable(df):
    print(f"Dataset Shape: {df.shape}")
    summary = pd.DataFrame(df.dtypes,columns=['dtypes'])
    summary = summary.reset_index()
    summary['Name'] = summary['index']
    summary = summary[['Name','dtypes']]
    summary['Missing'] = df.isnull().sum().values
    summary['Uniques'] = df.nunique().values
    summary['First Value'] = df.loc[0].values
    summary['Second Value'] = df.loc[1].values
    summary['Third Value'] = df.loc[2].values

    for name in summary['Name'].value_counts().index:
        summary.loc[summary['Name'] == name, 'Entropy'] = round(stats.
 ↪entropy(df[name].value_counts(normalize=True), base=2),2)

    return summary

resumetable(df_kick)
```

```
Dataset Shape: (10000, 15)
```

[5]:

	Name	dtypes	Missing	Uniques	First Value	Second Value	Third Value	Entropy
0	ID	int64	0	10000	1576537356	675907016	361890770	13.29
1	name	object	0	9999	Deko	Westside BJ's: The Gluten-Free, Organic Food T…	Crepe Diem Food Truck	13.29
2	category	object	0	158	Hardware	Food Trucks	Food	6.24
3	main_category	object	0	15	Technology	Food	Food	3.57
4	currency	object	0	14	USD	USD	USD	1.27
5	deadline	object	0	2617	2015-10-24	2015-02-01	2014-01-17	11.06
6	goal	float64	0	779	70000	250000	30000	6.50
7	launched	object	0	10000	2015-09-24 03:12:52	2015-01-02 20:55:07	2013-12-18 03:26:04	13.29
8	pledged	float64	0	4601	1888	1466	5723	10.16
9	state	object	0	6	failed	failed	failed	1.52
10	backers	int64	0	718	41	9	90	6.52
11	country	object	0	23	US	US	US	1.47
12	usd pledged	float64	105	5165	1888	1466	5723	
13	usd_pledged_real	float64	0	5660	1888	1466	5723	
14	usd_goal_real	float64	0	2769	70000	250000	30000	

```
12    10.16
13    10.67
14     8.19
```

Firstly, I wrote a function "resumetable" to show the rough information of this dataset. And We can observe this dataset has no missing data except the usd_pledged part

---

Then we check how the actual dataset looks like.

[14]:
```
df_kick.head()
```

[14]:

	ID	name	category	main_category	currency	deadline	goal	launched	pledged	state	backers	country	usd pledged	usd_pledged_real	usd_goal_real
0	1576537356	Deko	Hardware	Technology	USD	2015-10-24	70000.0	2015-09-24 03:12:52	1888.0	failed	41	US	1888.00	1888.00	70000.00
1	675907016	Westside BJ's: The Gluten-Free, Organic Food T…	Food Trucks	Food	USD	2015-02-01	250000.0	2015-01-02 20:55:07	1466.0	failed	9	US	1466.00	1466.00	250000.00
2	361890770	Crepe Diem Food Truck	Food	Food	USD	2014-01-17	30000.0	2013-12-18 03:26:04	5723.0	failed	90	US	5723.00	5723.00	30000.00
3	1225211551	Season's End - A horror novel ready for public…	Fiction	Publishing	GBP	2016-11-23	5500.0	2016-10-24 15:44:36	25.0	failed	2	GB	23.24	31.09	6839.01
4	2122944289	Colorado City Arizona Restaurant (Canceled)	Restaurants	Food	USD	2015-05-13	30000.0	2015-03-14 05:18:34	100.0	canceled	3	US	100.00	100.00	30000.00

[20]:
```
state = round(df_kick["state"].value_counts() / len(df_kick["state"]) * 100,2)

labels = list(state.index)
values = list(state.values)

trace1 = go.Pie(labels=labels, values=values, marker=dict(colors=['red']))

layout = go.Layout(title='Distribuition of States',␣
 ↪legend=dict(orientation="h"));

fig = go.Figure(data=[trace1], layout=layout)
iplot(fig)
```

0.1 The first and foremost of crowdfunding is whether it succeed or not.

`And I choose pie chart is because that it can easily show the percentage of all categories.`

From the results, we can easily see that the success rate is about 36%. Which means though it's not easy,but it still worth a try for those ineed.

0.2 Further, I want to know more about the relation between the goal and pledged. And how it affect the funding will succeed or not

Due to simplicity, I choose to use the natural log of the pledge/goal data. And I Will group some categories and after it, filter by Failed or successful projects. Although suspended and canceled project are caused by different situations, I will replace this categories by 'failed'

[36]:
```
# df_kick.loc[df_kick.state.isin(['suspended', 'canceled']), 'state'] = 'failed'
df_kick = df_kick.loc[df_kick['state'].isin(['failed','successful'])]

# df_kick.loc[df_kick.state.isin(['suspended','canceled']),'state']  --> In␣
 ↪df_kick  .loc[.....,'state'] the ... part means finding all the 'suspended'␣
 ↪and canceled
#  ,'state' part means to specified 'state' feature
# print(df_kick.loc[df_kick.state.isin(['suspended','canceled']),'state'])
```

[40]:
```
print("Min Goal and Pledged values")
print(df_kick[["goal", "pledged"]].min())
print("")
print("Mean Goal and Pledged values")
print(round(df_kick[["goal", "pledged"]].mean(),2))
print("")
print("Median Goal and Pledged values")
print(df_kick[["goal", "pledged"]].median())
print("")
print("Max Goal and Pledged values")
print(df_kick[["goal", "pledged"]].max())
```

```
Min Goal and Pledged values
goal       1.0
pledged    0.0
dtype: float64

Mean Goal and Pledged values
goal       48597.13
pledged    10161.46
dtype: float64

Median Goal and Pledged values
```

```
goal       5500.0
pledged     657.5
dtype: float64

Max Goal and Pledged values
goal       100000000.0
pledged      7072757.0
dtype: float64
```

[41]:
```
df_kick['pledged_log'] = np.log(df_kick['usd_pledged_real'] + 1)
df_kick['goal_log'] = np.log(df_kick['usd_goal_real'] + 1)

df_kick['diff_pledged_goal'] = round((df_kick['usd_pledged_real'] /
 ↪df_kick['usd_goal_real']) * 100, 2)
df_kick['diff_pledged_goal'] = df_kick['diff_pledged_goal'].astype(float)
```

[38]:
```
#First plot
trace0 = go.Box(
    x=df_kick['state'],
    y=df_kick['goal_log'],
    name="Goal Log", showlegend=False
)
#Second plot
trace1 = go.Box(
    x=df_kick['state'],
    y=df_kick['pledged_log'],
    name="Pledged Log", showlegend=False
)
#Third plot
trace2 = go.Scatter(
    x=df_kick['goal_log'], y=df_kick['pledged_log'],
    name="Goal x Pledged Distribuition",
    showlegend=False,
    mode = 'markers'
)

#Creating the grid
fig = tls.make_subplots(rows=5, cols=2, specs=[[{'rowspan': 2}, {'rowspan':␣
 ↪2}],[None,None],[None,None],[{'colspan': 2,'rowspan': 2}, None],[None,None]],
                          subplot_titles=('Goal','Pledged',
                                          "Goal x Pledged (Both)"))

#setting the figs
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 4, 1)
```

```
fig['layout'].update(showlegend=True,
                     title="Goal Log and Pledged Log by State of Projects",
                     xaxis=dict(
                         title='State', ticklen=5, zeroline=False, gridwidth=2
                     ),
                     yaxis=dict(
                         title='Goal(Log)', ticklen=5, gridwidth=2
                     ),
                     xaxis1=dict(title='State', ticklen=5, zeroline=False,␣
 ↪gridwidth=2),
                     yaxis1=dict(title='Goal(Log)', ticklen=5, gridwidth=2),
                     xaxis2=dict(title='State', ticklen=5, zeroline=False,␣
 ↪gridwidth=2),
                     yaxis2=dict(title='Pledged(Log)', ticklen=5, gridwidth=2))
iplot(fig)
```

0.3 By using box plot, and scatter plot we can easily find that the correlation between the correlation between goal/pledged number and success.

In terms of goal number, the difference isn't very notable. * Successful : Q3 = 9.21044 , Q1=7.285, median=8.257, IQR=1.92544 * Failed : Q3=9.998, Q1=7.9554 , median=8.95 , IQR=2.0426 We can probably say that the goal number dosen't directly affect the result.

But in terms of pledged number, we can easily see the difference. * Successful : Q3 = 9.4742 , Q1=7.616, median=8.54, IQR=1.8582 * Failed : Q3=6.566, Q1=0.94 , median=4.615 , IQR=5.626

Which makes perfect sense, the more money you are pledged more likely the funding will succeed.

But from the analysis we can also tell the interquartile range is quite small compared to total range (7.34). Which means the distribution of pledged money is very dispersive.We can also confirm that from there actually exist quite a few outlier.

0.4 Then, I want to analye further the Main_Categorys:

- Sucessful category's frequency
- failed category's frequency
- General Goal Distribuition by Category

[91]:
```
main_cats = df_kick["main_category"].value_counts()
main_cats_failed = df_kick[df_kick["state"] == "failed"]["main_category"].
 ↪value_counts()
main_cats_sucess = df_kick[df_kick["state"] == "successful"]["main_category"].
 ↪value_counts()

# Failed plot
```

```
trace0 = go.Bar(
    x=main_cats_failed.index,
    y=main_cats_failed.values,
    name="Failed Categories",
)
# Success plot
trace1=go.Bar(
    x=main_cats_sucess.index,
    y=main_cats_sucess.values,
    name="Success Categories"
)
#Overall
trace2 = go.Bar(
    x=main_cats.index,
    y=main_cats.values,
    name="Categories Distribuition",
    marker_color='#BF9D7A'
)
#Creating subply
fig = tls.make_subplots(rows=2,cols=2, specs= [[{},{}],[{'colspan':2},None]],␣
 ↪subplot_titles=('Failed','Sucessful', "General Category's"))

fig.append_trace(trace0,1,1)
fig.append_trace(trace1,1,2)
fig.append_trace(trace2,2,1)

fig['layout'].update(showlegend=True,
                     title="Main Category's Distribuition",
                     bargap=0.05,
                     template="seaborn")


iplot(fig)
```

0.5 From the above analysis, we can see that in the main_category part, 'Film & Video' and 'Music' are the most popular projects.

0.6 Following I want to check with the category part.

[92]:
```
categorys_failed = df_kick[df_kick["state"] == "failed"]["category"].
 ↪value_counts()[:25]
categorys_sucessful = df_kick[df_kick["state"] == "successful"]["category"].
 ↪value_counts()[:25]
```

```
categorys_general = df_kick["category"].value_counts()[:25]

#First plot
trace0 = go.Histogram(
    x=df_kick[(df_kick.category.isin(categorys_failed.index.values)) &
              (df_kick["state"] == "failed")]['category'].head(100000),

    histnorm='percent', name="Top 15 Failed",␣
 ↪showlegend=False,marker_color='#36688D',

)
#Second plot
trace1 = go.Histogram(
    x=df_kick[(df_kick.category.isin(categorys_sucessful.index.values)) &
              (df_kick["state"] == "successful")]['category'].head(100000),
    histnorm='percent', name="Top 15 Sucessful",␣
 ↪showlegend=False,marker_color='#A4A4BF'
)

#Third plot
trace2 = go.Histogram(
    x=df_kick[(df_kick.category.isin(categorys_general.index.
 ↪values))]['category'].head(100000),
    histnorm='percent', name="Top 25 All Category's",␣
 ↪showlegend=False,marker_color='#80ADD7'
)

#Creating the grid
fig = tls.make_subplots(rows=5, cols=2, specs=[[{'rowspan':2}, {'rowspan':
 ↪2}],[None,None],[None,None] ,[{'rowspan':2,'colspan': 2}, None],[None,None]],
                          subplot_titles=('Top 15 Failed','Top 15 Sucessful',␣
 ↪"Top 25 All Category's"))

#setting the figs
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 4, 1)

fig['layout'].update(showlegend=True, title="Top Frequency Category's")
iplot(fig)
```

[99]:
```
#First plot
trace0 = go.Box(
    x=df_kick[(df_kick.category.isin(categorys_failed.index.values)) &
              (df_kick["state"] == "failed")]['category'],
    y=df_kick[(df_kick.category.isin(categorys_failed.index.values)) &
```

```
                (df_kick["state"] == "failed")]['pledged_log'].head(100000),
    name="Failed Category's", showlegend=False,marker_color='#36688D'
)

#Second plot
trace1 = go.Box(
    x=df_kick[(df_kick.category.isin(categorys_sucessful.index.values)) &
                (df_kick["state"] == "successful")]['category'],
    y=df_kick[(df_kick.category.isin(categorys_sucessful.index.values)) &
                (df_kick["state"] == "successful")]['pledged_log'].head(100000),
    name="Sucessful Category's", showlegend=False,marker_color='#A4A4BF'
)

#Third plot
trace2 = go.Box(
    x=df_kick[(df_kick.category.isin(categorys_general.index.
 ↪values))]['category'],
    y=df_kick[(df_kick.category.isin(categorys_general.index.
 ↪values))]['pledged_log'].head(100000),
    name="All Category's Distribuition", showlegend=False,marker_color='#80ADD7'
)

#Creating the grid
fig = tls.make_subplots(rows=2, cols=2, specs=[[{}, {}], [{'colspan': 2},␣
 ↪None]],
                          subplot_titles=('Failed','Sucessful', "General␣
 ↪Category's", ))

#setting the figs
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 2, 1)

fig['layout'].update(showlegend=True, title="Main Category's Distribuition")
iplot(fig)
```

0.7 From above information , we can find an interesting fact.

0.8 We can see that almost all categorys in sucessful have the same distribuition of values but some video games projects have the highest values in % difference of Pledged by Goal

0.9 On the other side, the failed ones don't seem like have a pattern. And the distribution is very chaotic. The IQR is quite large, which means the distribution is very dispersive.

0.10 In sum, from the analysis we conducted so far, we can find out that the successful crowdfunding are following more similar pattern. And dose it imply that the subjects aren't the key factors that determine the success or not? It may be a interesting question.