2. Exploring data

February 22, 2024

Introduction

Objectives: The main objectives of the project are to develop a classification system for distinguishing between successful and failed projects on Kickstarter based on data crawled from the platform. This system aims to provide valuable insights to project creators, guiding them in setting up effective campaign strategies and making informed decisions about launching crowdfunding projects.

Data set

We began with an extensive dataset comprising X projects seeking crowdfunding on the Kickstarter platform. This data was organized chronologically by month and year within Kickstarter. Due to the sheer volume of available data, it became apparent that processing it without a structured approach would be impractical. Consequently, we opted to focus on the most recent years, specifically from 2020 onwards, and selected one CSV file per month per year. This resulted in the utilization of 48 CSV files, representing 48 months across 4 years, amounting to _____ entries.

To construct a predictive model, it's essential to establish a structured data lake. This involves organizing the data into a format conducive to analysis and modelin The data lake should include features such as project category, funding goal, campaign duration, project description, creator background, funding success/failure, and any other relevant variables. Each entry should be accurately labeled to facilitate supervised learning.arity.g.

However, several challenges and potential biases may arise during this process:

- Sampling Bias: By focusing solely on recent years, there may be a bias towards contemporary trends and project characteristics. Older projects, which could offer valuable historical insights, may be underrepresented or excluded entirely.
- Selection Bias: The decision to include only one CSV per month per year may inadvertently prioritize certain types of projects or time periods, leading to a biased sample.
- Imbalanced Classes: The dataset may exhibit an imbalance between successful and failed projects, with one class significantly outnumbering the other. This can skew the predictive model's performance and accuracy.
- Missing Data: Some entries may contain missing or incomplete information, which can hinder the effectiveness of the predictive model if not addressed appropriately.
- Feature Engineering: Identifying and extracting relevant features from the raw data requires careful consideration and domain expertise. It's crucial to select features that have predictive power while avoiding those that introduce noise or multicollinearity.

It will necessitate thorough data preprocessing, feature engineering, and model validation techniques to mitigate biases and ensure the model's generalizability and effectiveness.

[21]:
```
import pandas as pd

df = pd.read_csv(r'uncleaned_4years.csv', low_memory=False)
```

[23]:
```
#we need to first exclude any duplicates there might be, so that the data␣
↪exploration doesn't suffer any changes to when modelling
df = df.drop_duplicates(ignore_index=True)

#understanding the size of the df
print("The dataset has",len(df),"rows")
print("They are divided into columns and rows:",df.shape)

#what columns does it include and data types
print(df.info())
```

```
The dataset has 146925 rows
They are divided into columns and rows: (146925, 48)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146925 entries, 0 to 146924
Data columns (total 48 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   friends                  111 non-null     object
 1   state_changed_at         146925 non-null  int64
 2   blurb                    146914 non-null  object
 3   id                       146925 non-null  int64
 4   static_usd_rate          146925 non-null  float64
 5   permissions              111 non-null     object
 6   location                 146785 non-null  object
 7   backers_count            146925 non-null  int64
 8   deadline                 146925 non-null  int64
 9   source_url               146925 non-null  object
 10  usd_type                 146851 non-null  object
 11  photo                    146925 non-null  object
 12  is_starred               111 non-null     object
 13  is_backing               111 non-null     object
 14  category                 146925 non-null  object
 15  goal                     146925 non-null  float64
 16  creator                  146925 non-null  object
 17  is_starrable             146925 non-null  bool
 18  staff_pick               146925 non-null  bool
 19  pledged                  146925 non-null  float64
 20  usd_exchange_rate        89899 non-null   float64
 21  country_displayable_name 146925 non-null  object
 22  currency_symbol          146925 non-null  object
```

```
 23  video                      2271 non-null    object
 24  created_at                 146925 non-null  int64
 25  country                    146925 non-null  object
 26  is_launched                3341 non-null    object
 27  usd_pledged                146014 non-null  float64
 28  unseen_activity_count      0 non-null       float64
 29  is_disliked                3341 non-null    object
 30  name                       146925 non-null  object
 31  urls                       146925 non-null  object
 32  spotlight                  146925 non-null  bool
 33  last_update_published_at   0 non-null       float64
 34  unread_messages_count      0 non-null       float64
 35  currency                   146925 non-null  object
 36  percent_funded             3341 non-null    float64
 37  converted_pledged_amount   146014 non-null  float64
 38  current_currency           146925 non-null  object
 39  is_liked                   3341 non-null    object
 40  state                      146925 non-null  object
 41  slug                       146925 non-null  object
 42  profile                    146925 non-null  object
 43  disable_communication      146925 non-null  bool
 44  currency_trailing_code     146925 non-null  bool
 45  launched_at                146925 non-null  int64
 46  fx_rate                    146925 non-null  float64
 47  prelaunch_activated        3341 non-null    object
dtypes: bool(5), float64(11), int64(6), object(26)
memory usage: 48.9+ MB
None
```

The variables that seem the most interesting at this point are Category, Country, State and Currency, in the object type, and Converted Pledged Amount (as we have several different currencies). It would be interesting to look at goal, but being in different currencies, doesn't really provide a valid insight. The last one could potentially be our target variable. We will then explore those variables further:

[25]:
```
df[['category','country','state','creator','currency']].describe(include=object)
```

[25]:
```
                                                  category country       state  \
count                                               146925  146925      146925
unique                                                 354      25           7
top     {"id":253,"name":"Webcomics","analytics_name":…      US  successful
freq                                                  5621   99284       89622

                                                   creator currency
count                                               146925   146925
unique                                              146715       15
top     {"id":2118747970,"name":"Gladys","slug":"gmutu…      USD
freq                                                     5    99284
```

[27]:
```
df[['converted_pledged_amount']].describe()
```

[27]:

	converted_pledged_amount
count	1.460140e+05
mean	1.626176e+04
std	1.571112e+05
min	0.000000e+00
25%	2.000000e+02
50%	2.109000e+03
75%	8.292250e+03
max	4.175415e+07

The state variable will be our target variable and we noticed that it has 7 different outputs. We also noticed that we have 15 different currencies, with over 67% being USD. Nevertheless, we will we have to use always converted amounts in our analysis, to avoid wrong conclusions caused by different currencies. Regarding the Creator variable, almost all rows have a unique entry, meaning that usually there is only one project per creator. In category we have 354 different ones.

Visual Representation

1. Distribution of Funding Goals vs. Pledged Amounts and Percentage of Goal Pledged:

Create a scatter plot where the x-axis represents the funding goal and the y-axis represents the pledged amount. Use color coding or size of points to indicate successful and failed projects. Analyze the distribution and look for any patterns or outliers.

In order to use the goal variable, we need to convert it to USD, so that it becomes comparable with pledged amount. We will use usd_exchange_rate variable as it was the one used to arrive to converted_pledged_amount from pleged_amount. To analyse the percetange of goal that was fulfilled, we need to create this column in the df dividing the converted pledged by the goal.

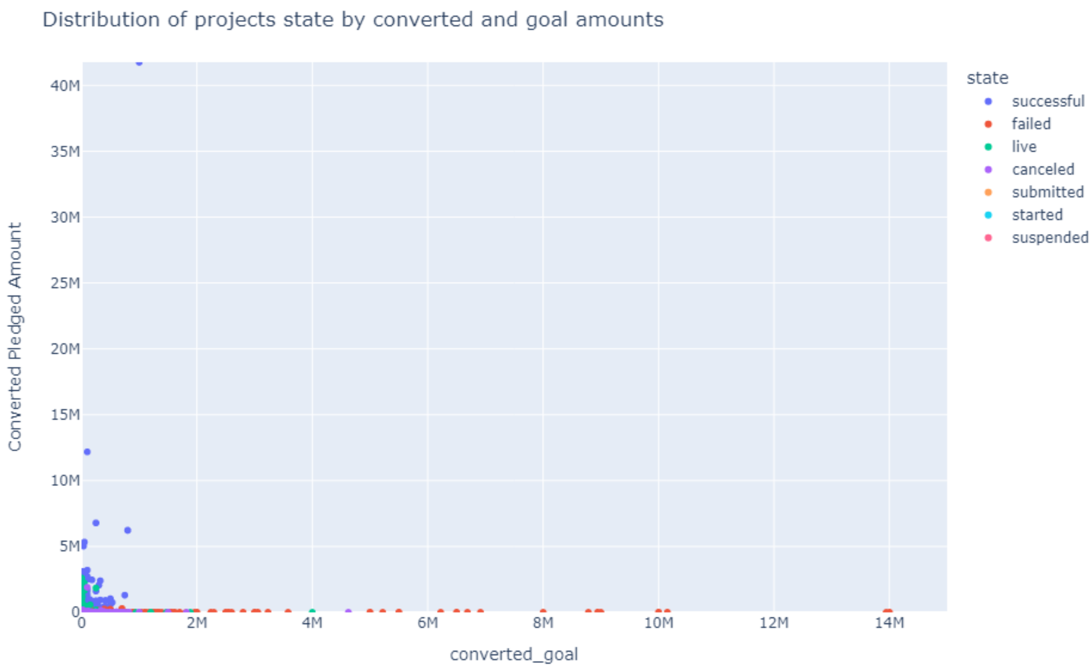
[93]:
```
#we need to create the converted to usd goal column and also the division␣
 ↪between sucessfull and not_successfull projects
df['converted_goal'] = df['usd_exchange_rate'] * df['goal']

import plotly.express as px

df_success = df[(df['state'] == 'successful') | (df['state'] == 'failed')]
fig = px.scatter(df, x='converted_goal', y='converted_pledged_amount',
                 color='state', hover_name="state")

fig.update_layout(title= 'Distribution of projects state by converted and goal␣
 ↪amounts',
                  yaxis_title='Converted Pledged Amount',width=800, height=600)

fig.update_xaxes(range=[0,15000000])
fig.update_yaxes(range=[0, 4.175415e+07])
fig.show()
```



CONCLUSION:

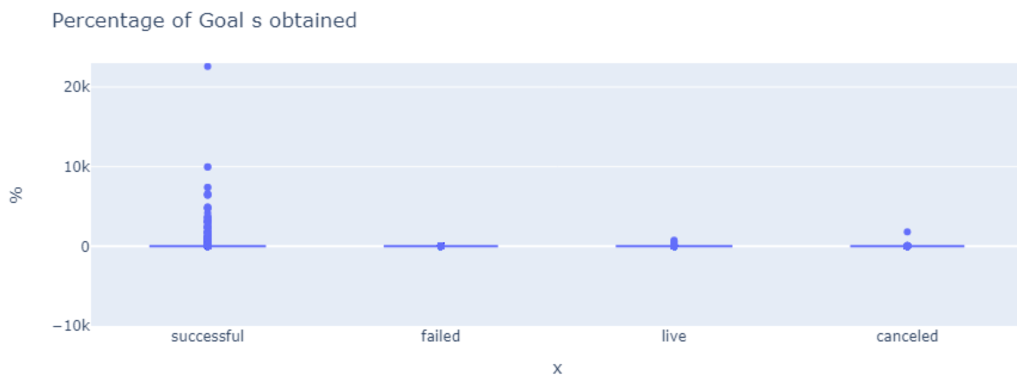
the data is very dispersed in both goal and converted amount which brings the following challenges:

- Difficulty in Identifying Patterns: High dispersion can make it challenging to identify meaningful patterns or relationships between the variables. With a wide range of values and considerable variability, it becomes harder to discern any underlying trends or correlations.
- Overplotting: Overplotting can occur when numerous data points are densely packed in a scatter plot, making it difficult to distinguish individual points and assess their density or distribution accurately. This can obscure patterns and lead to misinterpretations of the dat
- Reduced Predictive Power: In highly dispersed data, predictive models may struggle to generalize well beyond the observed data points. Models trained on such data may have limited ability to make accurate predictions or classifications on unseen data due to the lack of clear trends or patterns.
- Increased Uncertainty: High dispersion typically leads to greater uncertainty in estimates and predictions. Confidence intervals may widen, and predictions may become less precise as the variability in the data increases, making it harder to draw reliable conclusions from the analysis.
- Outlier Influence: In dispersed data, outliers can have a more significant impact on analyses and models. Outliers may skew summary statistics, distort relationships between variables, and disproportionately influence model predictions, leading to biased results if not handled appropriately.
- Model Assumptions Violation: High dispersion can violate assumptions of certain statistical

models, such as normality or homoscedasticity (constant variance). For example, linear regression assumes constant variance along the entire range of predictor variables, which may not hold true in the presence of highly dispersed data.

- Difficulty in Decision Making: When data points are scattered across a wide range, decision-making becomes more challenging. Stakeholders may struggle to derive actionable insights or make informed decisions based on the data, particularly if patterns are unclear or contradictory.ntradictory.

[136]:
```
df['percentage_goal_obtained'] = df['converted_pledged_amount']/
  ↪df['converted_goal']
fig = px.box(x=df['state'], y=df['percentage_goal_obtained'])
fig.update_yaxes(range=[-10000,23000])
fig.update_layout(title='Percentage of Goal s obtained', yaxis_title = '%')
fig.show()
```



2. Success Rate by Project Category:

Create a bar chart showing the count of successful and failed projects for each project category. Calculate the success rate (percentage of successful projects) for each category. Perform statistical analysis (e.g., chi-square test) to determine if there is a significant difference in success rates across categorioject success rates

[156]:
```
# we need to address the column category and break it into different columns to␣
  ↪be able to arrive to the different categories
# we have in the dataset

df_split = df['category'].str.split(',', expand=True)
df_split.columns = [f'coluna_{i}' for i in range(len(df_split.columns))]
df_split = df_split.drop(columns=df_split.columns[2:13])
df_split = df_split.rename({'coluna_0':'category_id','coluna_1':
  ↪'category_name'},axis = 'columns')
```

```
def split_and_rename_column(df, column_name):
    # Splitting the column
    df_split = df[column_name].str.split(':', expand=True)

    # Keeping only the second column after splitting
    df_split = df_split.iloc[:, 1]

    # Renaming the column
    df_split.name = column_name

    return df_split
# Define the column names
columns_to_process = ['category_id','category_name']

for column in columns_to_process:
    df[column] = split_and_rename_column(df_split, column)
```

[192]:
```
state_category = df.groupby('category_name')['state'].value_counts()
print(state_category)
state_category['perc_successful'] = state_category['state'].loc['successful'] /␣
↪state_category.groupby('category_name')['state'].sum()

fig = px.bar(state_category, x='category_name', y='state', color='state',
             title='Percentage of Each State Within Each Category',
             labels={'category': 'Category_Name', 'percentage': 'Percentage␣
↪(%)', 'state': 'State'},
             barmode='group',
             width=800, height=600)
```

```
category_name  state
"3D Printing"  successful    183
               failed        170
               canceled       49
               live            6
               suspended       2
                             …
"Zines"        successful    154
               failed         73
               canceled       14
               live            6
               submitted       1
Name: count, Length: 673, dtype: int64
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/indexes/base.py:3653, in Index.get_loc(self, key)
```

```
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/_libs/index.pyx:147, in pandas._libs.index.IndexEngine.get_loc()

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/_libs/index.pyx:176, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()

KeyError: 'state'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[192], line 3
      1 state_category = df.groupby('category_name')['state'].value_counts()
      2 print(state_category)
----> 3 state_category['perc_successful'] = state_category['state'].
↪loc['successful'] / state_category.groupby('category_name')['state'].sum()
      5 fig = px.bar(state_category, x='category_name', y='state', color='state',
      6              title='Percentage of Each State Within Each Category',
      7              labels={'category': 'Category_Name', 'percentage':␣
↪'Percentage (%)', 'state': 'State'},
      8              barmode='group',
      9              width=800, height=600)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/series.py:1007, in Series.__getitem__(self, key)
   1004     return self._values[key]
   1006 elif key_is_scalar:
-> 1007     return self._get_value(key)
   1009 if is_hashable(key):
   1010     # Otherwise index.get_value will raise InvalidIndexError
   1011     try:
   1012         # For labels that don't resolve as scalars like tuples and␣
↪frozensets

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/series.py:1116, in Series._get_value(self, label, takeable)
   1113     return self._values[label]
   1115 # Similar to Index.get_value, but we do not fall back to positional
```

```
-> 1116 loc = self.index.get_loc(label)
   1118 if is_integer(loc):
   1119     return self._values[loc]

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/indexes/multi.py:2812, in MultiIndex.get_loc(self, key)
   2809     return mask
   2811 if not isinstance(key, tuple):
-> 2812     loc = self._get_level_indexer(key, level=0)
   2813     return _maybe_to_slice(loc)
   2815 keylen = len(key)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/indexes/multi.py:3160, in MultiIndex._get_level_indexer(self, key, ␣
↪level, indexer)
   3157         return slice(i, j, step)
   3159 else:
-> 3160     idx = self._get_loc_single_level_index(level_index, key)
   3162     if level > 0 or self._lexsort_depth == 0:
   3163         # Desired level is not sorted
   3164         if isinstance(idx, slice):
   3165             # test_get_loc_partial_timestamp_multiindex

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/indexes/multi.py:2752, in MultiIndex.
↪_get_loc_single_level_index(self, level_index, key)
   2750     return -1
   2751 else:
-> 2752     return level_index.get_loc(key)

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
↪pandas/core/indexes/base.py:3655, in Index.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'state'
```

[ ]:

```

```

Backers Count Distribution:

Plot a histogram of backers count to visualize the distribution. Calculate summary statistics such as mean, median, and standard deviation. Look for skewness or outliers in the distribution. Time Series Analysis of Project Launches:

Create a line chart showing the number of projects launched over time (e.g., monthly or quarterly). Use a trendline or moving average to identify any long-term trends or seasonal patterns. Analyze any spikes or dips in project launches and correlate them with external factors if possible. Geographical Distribution of Projects:

Create a map visualization showing the geographical distribution of projects. Use different colors or markers to represent successful and failed projects. Analyze if there are any geographical patterns in project success rates

[ ]:
```
from datetime import datetime, timezone

df['created_at'] = df['created_at'].apply(lambda x: datetime.
 ↪utcfromtimestamp(x).replace(tzinfo=timezone.utc))
```