2. Exploring data

February 25, 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.

[ ]:
```
import pandas as pd

df = pd.read_csv(r'uncleaned_4years.csv', low_memory=False)
pd.options.display.float_format = '{:.2f}'.format
```

[4]:
```
#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 State variable is our target variable. We will then explore those variables further:

[6]:
```
df[['category','country','state','creator','currency']].describe(include=object)
```

[6]:

	category	country	state	creator	currency
count	146925	146925	146925	146925	146925
unique	354	25	7	146715	15
top	{"id":253,"name":"Webcomics","analytics_name":…	US	successful	{"id":2118747970,"name":"Gladys","slug":"gmutu…	USD
freq	5621	99284	89622	5	99284

[8]:
```
df[['converted_pledged_amount']].describe()
```

[8]:

	converted_pledged_amount
count	146014.00
mean	16261.76
std	157111.25
min	0.00
25%	200.00
50%	2109.00
75%	8292.25
max	41754153.00

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.

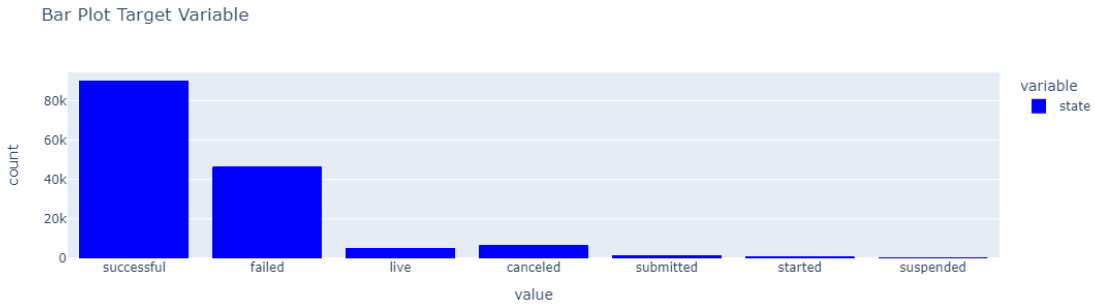
Visual Representation

Target Variable

Understanding the target variable is crucial because it allows me to grasp the core problem I'm trying to solve. By understanding the target variable, I gain insights into what I aim to predict or classify. This comprehension guides my entire modeling process, from selecting appropriate features to choosing the right algorithm.

Checking if I have sufficient data on the target variable is essential because it directly impacts the performance and reliability of my model. Insufficient data may lead to biased or unreliable predictions. It's vital to ensure that I have an adequate number of samples for each class or category within the target variable to train a robust and accurate model. Without enough data, my model may struggle to generalize well to new, unseen instances, resulting in poor performance in real-world scenarios. Therefore, assessing data sufficiency helps me make informed decisions about whether additional data collection or sampling strategies are necessary to improve the quality of my model.

[31]:
```
# we will create a smaller dataframe to increase efficiency in plotting
daf = df['state']

import plotly.express as px

fig = px.bar(daf, title= 'Bar Plot Target Variable', text_auto=True)
fig.update_traces(marker_color = 'blue', marker_line_color = 'blue',
                  marker_line_width = 2, opacity = 1,textfont_size = 8,␣
 ↪textangle = 0, textposition = "outside")
fig.update_yaxes(y = 'Frequency')
fig.update_xaxes(x = 'State')

fig.show()
```



0. Correlation Map

[696]:
```
import matplotlib.pyplot as plt

numeric_columns = df.select_dtypes(include=['int64', 'float']).columns
df_2 = df[numeric_columns].copy()

df_2 = df_2.dropna(axis=1, how='all')

corr = df_2.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[696]: <pandas.io.formats.style.Styler at 0x7f22600bcc50>

Conclusions:

Major Positive Correlations:

Backers Count and USD Pledged: The number of backers is strongly positively correlated with the amount pledged in USD (0.83). This indicates that as the number of backers increases, the total amount pledged tends to increase as well.

Backers Count and Converted Pledged Amount: Similarly, there is a strong positive correlation between the number of backers and the converted pledged amount (0.83). This suggests that as the number of backers increases, the total amount pledged in the project's currency also tends to increas e. Backers Count and Percent Funded: The number of backers is strongly positively correlated with the percentage funded (0.83). This implies that projects with a higher number of backers tend to achieve a higher percentage of their funding goa l. Deadline and Launched At: There is a strong positive correlation between the deadline and the launch time of the project (0.95). Projects launched closer to their deadline are more likely to have a higher number of backe rs. Deadline and State Changed At: Projects with longer durations, indicated by a later deadline, are moderately positively correlated with the date of state change (0.55). This suggests that projects with longer durations are more likely to transition from unfunded to funded status as the deadline approa

ches. Major Negative Correla tions: Static USD Rate and USD Exchange Rate: There is a negative

correlation between the static USD rate and the USD exchange rate (-0.12). This implies that changes in the static USD rate are inversely related to changes in the USD exchange rate. Launched At and FX Rate: The launch time of the project is weakly negatively correlated with the foreign exchange rate (-0.0049). This suggests a slight tendency for projects launched at certain times to coincide with fluctuations in the foreign exchang e rate. None: There don't seem to be any other major negative corre

lations. Neutral Cor relations: ID and Goal: There is a very weak correlation between the project ID and the funding goal (0.0035), indicating no significant relationship between these variables. ID and Pledged: Similarly, the project ID and the pledged amount have a very weak correlation (-0.0045), suggesting no clear relationship be tween them. None: There don't seem to be any other notable neutral c

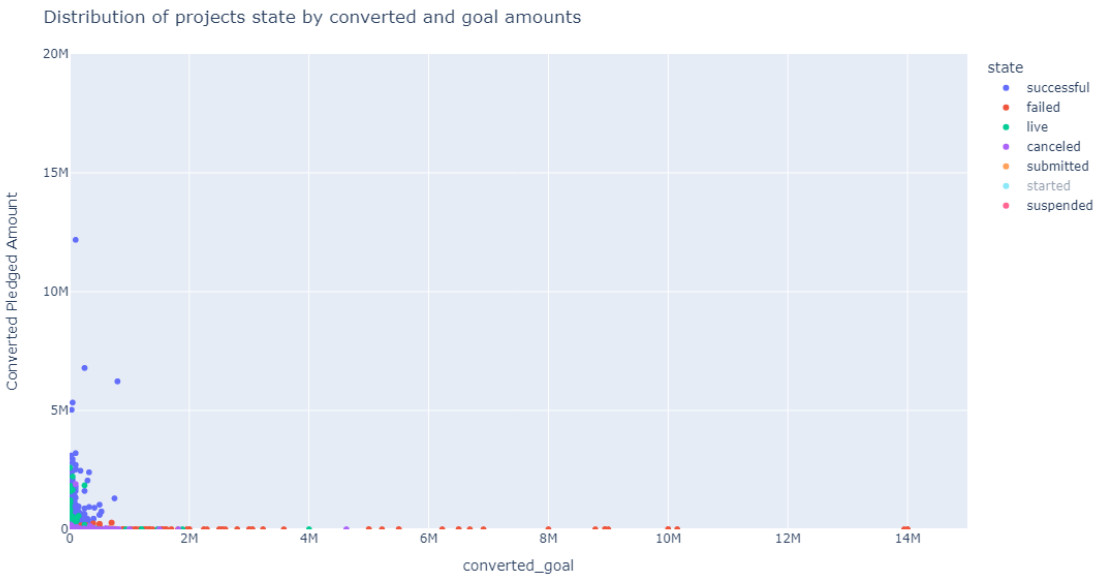
orrelations. These observations provide insights into the relationships between different variables in the dataset. Further analysis can be conducted based on these correlations to understand the underlying dynamics of the data.

1. Distribution of Funding Goals vs. Pledged Amounts:

Scatter plot where the x-axis represents the funding goal and the y-axis represents the pledged amount. The purpose is to 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.

[10]:
```
#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, 20000000])
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 contra-dictory.ntradictory.

2. Success Rate by Project Category:

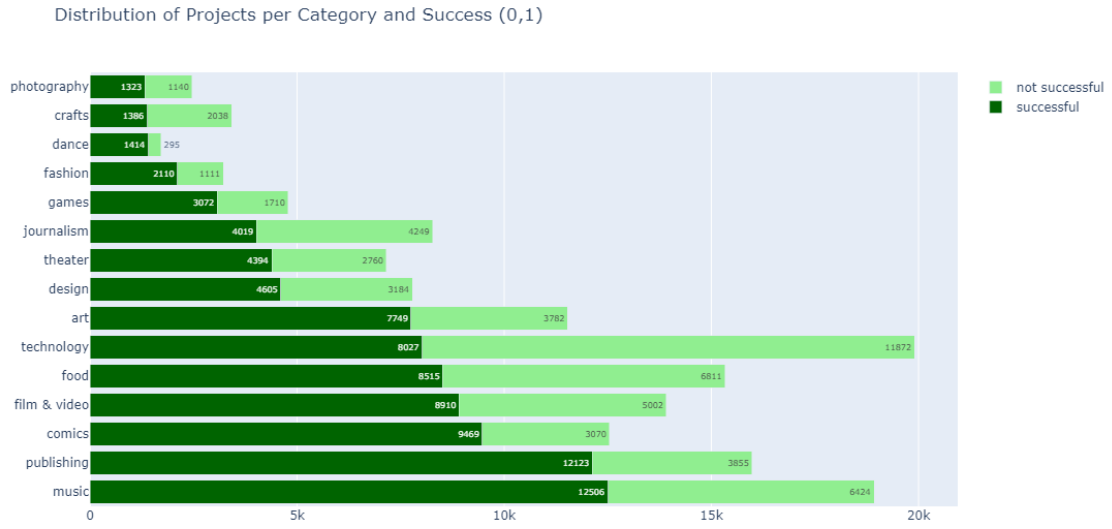
Create a bar chart showing the count of successful and failed projects for each project category. Calculate the success rate (percentage of successful projects) for each category. Perform statistical analysis (e.g., chi-square test) to determine if there is a significant difference in success rates across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

[93]:
```
# 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

import json

def extract_slug(row):
    data_dict = json.loads(row)
    return data_dict.get('slug')

df['category'] = df['category'].apply(extract_slug)
df['category'] = df['category'].str.split('/')
df['category'] = df['category'].str[0]
```

```
count         146925
unique            15
top       technology
freq           19899
Name: category, dtype: object
```

[671]:
```
import plotly.graph_objects as go

s_values = df[df['state'] == 'successful']['category'].value_counts()
ns_values = df[df['state'] != 'successful']['category'].value_counts()


fig = go.Figure()
fig.add_trace(go.Bar(
    y= s_values.index,
    x= s_values.values,
    name='successful',
    orientation='h',
    text=s_values.values,
    textfont=dict(size=8),
    marker=dict(color='darkgreen')
))
```

```
fig.add_trace(go.Bar(
    y= ns_values.index,
    x= ns_values.values,
    name='not successful',
    orientation='h',
    text=ns_values.values,
    textfont=dict(size=8),
    marker=dict(color='lightgreen')
))

fig.update_layout(barmode='stack', title = 'Distribution of Projects per␣
 ↪Category and Success (0,1)', height=600 )
fig.show()
```



CONCLUSIONS:

It appears that the number of observations per category is the same for both successful and unsuccessful projects. This suggests that the distribution of projects across categories is consistent regardless of their success status. Using the "category" feature to predict the success of a project could be a valuable approach. In summary, while there is no data imbalance within each category in the dataset, but the total number of projects varies significantly among different categories, with some categories having many more projects than others. For example, "music" and "technology" have significantly higher numbers of projects compared to "dance" and "photography."

Data imbalance can affect the performance of machine learning models, particularly in cases where the minority class (categories with fewer projects) is of interest. It's essential to address this imbalance during model training and evaluation to ensure fair and accurate predictionsa.

3. 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.

[642]:
```
s_values = df[df['state'] == 'successful']['backers_count'].value_counts().
 ↪reset_index()
ns_values = df[df['state'] != 'successful']['backers_count'].value_counts().
 ↪reset_index()

daf = s_values.merge(ns_values, on='backers_count', how='outer',␣
 ↪suffixes=('_successful', '_not_successful'))
print(daf['backers_count'].describe())
```

```
count     2644.00
mean      2562.57
std       5530.66
min          0.00
25%        660.75
50%       1383.00
75%       2792.25
max     185341.00
Name: backers_count, dtype: float64
```

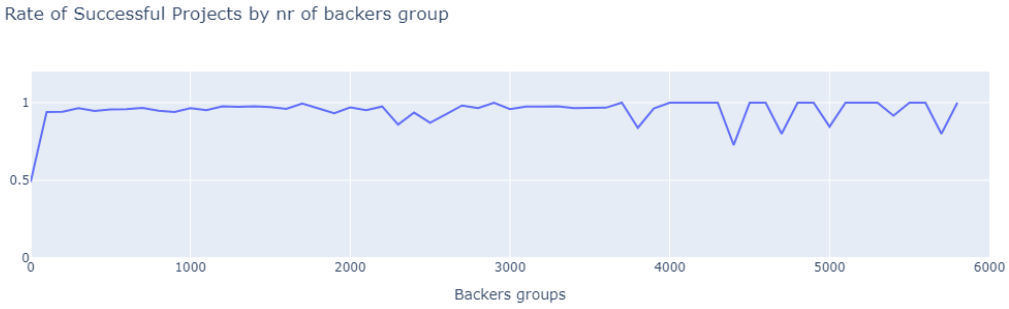
Based on the above information, the stdd deviation for the backers_count variable is very high. Going forward we decided to ignore the outliers and plot the variable without them.

[644]:
```
# Calculate the percentage of successful projects per backers_count

daf['count_successful'] = daf['count_successful'].fillna(0)
daf['count_not_successful'] = daf['count_not_successful'].fillna(0)

daf = daf.sort_values(by='backers_count', ascending=True)
```

[616]:
```
# based on the stats, the Q3 is 2792. We will then group the backers count in␣
 ↪groups of 100, to a max of Q3+1.5*IQR, excluding this way the outliers
# Calculate quartiles
Q1 = daf['backers_count'].quantile(0.25)
Q3 = daf['backers_count'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = round(Q3 + 1.5 * IQR, -2)

upper_bound = int(upper_bound)
bins = range(0, upper_bound, 100)

# Create a new column indicating the group each backers_count falls into
```

```
daf['backers_group'] = pd.cut(daf['backers_count'], bins=bins, right=False,␣
 ↪labels=range(0, 5900, 100))

# Group by the new column and aggregate count_successful and␣
 ↪count_not_successful
daf_g = daf.groupby('backers_group').agg({'count_successful': 'sum',␣
 ↪'count_not_successful': 'sum'}).reset_index()

daf_g['success_rate'] = (daf_g['count_successful'] / (daf_g['count_successful']␣
 ↪+ daf_g['count_not_successful']))
daf_g['not_success_rate'] = (daf_g['count_not_successful'] /␣
 ↪(daf_g['count_successful'] + daf_g['count_not_successful']))
```

[673]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=daf_g['backers_group'], y=daf_g['success_rate'],␣
 ↪name='Successful Projects', mode='lines'))

fig.update_xaxes(title_text='Backers groups')
fig.update_layout(yaxis_range=[0,1.2], xaxis_range=[0,6000], height=300,␣
 ↪title='Rate of Successful Projects by nr of backers group')

fig.show()
```



CONCLUSIONS:

There are some fluctuations in success and failure rates across different groups of backers. For example, there are instances where projects with a moderate number of backers have higher success rates compared to adjacent groups with slightly higher or lower numbers of backers. This indicates that success rates can be influenced by factors other than just the number of backers, such as project quality, marketing efforts, or timing.

T There are some outliers where the success rate is exceptionally high or low compared to adjacent groups. For instance, there are some groups where the success rate is close to 1.00 (100%) or close to 0.00 (0%), indicating unusually high or low success rates compared to the overall trend.

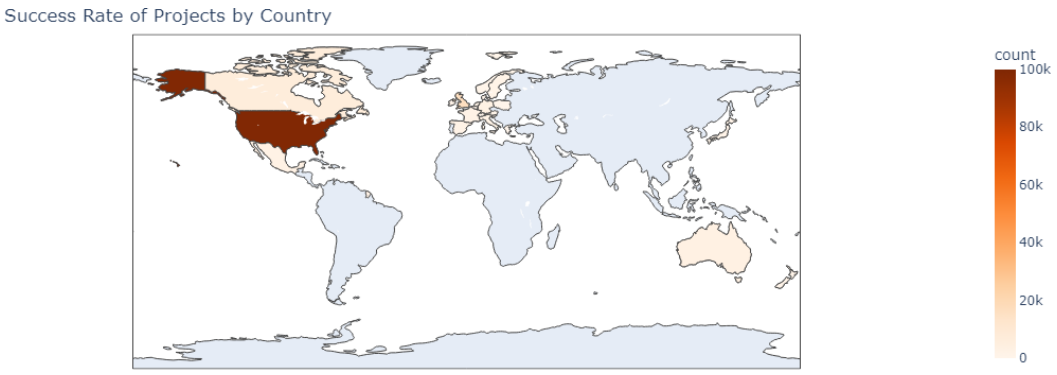
4. Geographical Distribbution of Projects:

Map visualization showing the success rate by geographical distribution of projects.

[397]:
```
# first we need to adjust the country names to ISO-3
import pycountry

def get_iso3(country_name):
    try:
        country = pycountry.countries.get(name=country_name)
        if country:
            return country.alpha_3
        else:
            return country_name
    except LookupError:
        return country_name


df['country_iso3'] = df['country_displayable_name'].apply(get_iso3)

iso3_mapping = {
    'the Netherlands': 'NLD',
    'the United Kingdom': 'GBR',
    'the United States': 'USA'
}

df['country_iso3'] = df['country_iso3'].replace(iso3_mapping)


# then we can build our geographic chart
daf = df[['country_iso3','state']]

country_counts = daf['country_iso3'].value_counts().reset_index()
```

[691]:
```
fig = px.choropleth(country_counts,
                    locations='country_iso3',
                    locationmode="ISO-3",
                    color='count',
                    color_continuous_scale=px.colors.sequential.Oranges,
                    title='Success Rate of Projects by Country',
                    labels={'success_rate': 'Success Rate'},
                    )

fig.update_layout(margin=dict(l=20, r=15, t=25, b=20),
                  autosize=True,
                  coloraxis_colorbar=dict(thickness=20,len=1),
                  coloraxis=dict(cmin=0,cmax=100000)
                 )
```

```
fig.show()
```



CONCLUSIONS:

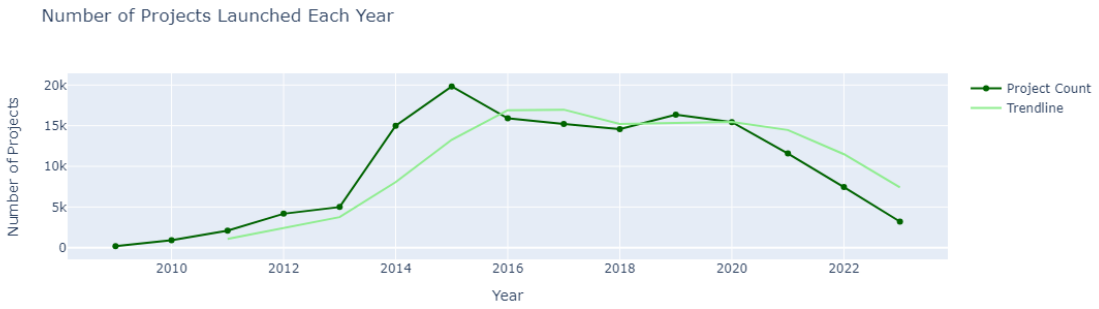
- Skewness: The distribution of counts across countries appears to be highly skewed, with a few countries having significantly higher counts compared to others. For example, USA has a count of 99284, while many other countries have counts less than 1000.
- Outliers: There are clear outliers in the dataset, such as the count for USA, which is much higher than the counts for other countries. These outliers may need to be handled carefully, depending on the analysis goalsal purposes.

5. 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.

[655]:
```
from datetime import datetime, timezone
import plotly.graph_objects as go

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

df['year'] = df['created_at'].dt.year
df_grouped = df.groupby('year').size().reset_index(name='project_count')

fig = go.Figure()
fig.add_trace(go.Scatter(x=df_grouped['year'],
                         y=df_grouped['project_count'],
                         mode='lines+markers',
                         name='Project Count',
                         line=dict(color='darkgreen'),
                         marker=dict(color='darkgreen')
                        ))
```

```
fig1.update_layout(title = 'Distribution of projects created per year',
                   xaxis_title = 'Year of creation',
                   yaxis_title = '% of Projects')


df_grouped['trendline'] = df_grouped['project_count'].rolling(window=3).mean()␣
 ↪# You can adjust the window size
fig.add_trace(go.Scatter(x=df_grouped['year'],
                         y=df_grouped['trendline'],
                         mode='lines',
                         name='Trendline',
                         line=dict(color='lightgreen'),
                         marker=dict(color='lightgreen')
                        ))

fig.update_layout(title='Number of Projects Launched Each Year',
                   xaxis_title='Year',
                   yaxis_title='Number of Projects')
fig.show()
```



CONCLUSIONS:

The trendline represents the moving average of project counts over a rolling window, providing a smoothed representation of the overall trend.

In 2009 and 201ce there is not enough data to calculate the trendliso itionly starts in 2011. nd.

Between 2011 and 2015, there is a noticeable increase in the number of projects launched each year, as indicated by both the project count and the tre line.

From 2016 to 2018, the number of projects launched remains relatively stable, with minor fluctuations around the trF

However, from 2020 onwards, there is a decline in the number of projects launched each year, as indicated by both the project count and the trendline. This decline suggests a potential shift in the trend or external factors affecting pro, for e.g. covid effect and following economic deceleration jec

The quality of the“project_coun by year of creation” variable as a predictor in a prediction model depends on several factors:-

Relevance: The variabit could reflect the level of activity or investment in a certain are- .

Consistency: the trendline shows a relatively consistent trend over the years, indicating that the number of projects launched has some level of stabiy addressed.vity over time.

of the data.