Group Project of **DA623** at *IITG (2023 Jan-May Semester)*

Team Name : **Pulse**

Members: Pallapu Mohan Krishna,Shania H,Avantika Sahu

```
#to upload kaggle token  api json file
from google.colab import files

files.upload()
```

```
Choose Files kaggle.json
• kaggle.json(application/json) - 69 bytes, last modified: 4/27/2023 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json':
b'{"username":"pmohankrishna","key":"d025db3a899ebbdfae8e68b44491849b"}'}
```

```
!ls -lha kaggle.json
```

```
-rw-r--r-- 1 root root 69 Apr 29 18:53 kaggle.json
```

```
!pip install -q kaggle
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
!chmod 600 /root/.kaggle/kaggle.json
```

```
!pwd
```

```
/content
```

```
#change the directory to our working location
%cd /content/drive/MyDrive/BirdClef2023
```

```
/content/drive/MyDrive/BirdClef2023
```

```
#list out the datasets
!kaggle datasets list
```

```
ref                                                            title
-------------------------------------------------------------  --------------------------------------------------
arnabchaki/data-science-salaries-2023                          Data Science Salaries 2023
salvatorerastelli/spotify-and-youtube                          Spotify and Youtube
erdemtaha/cancer-data                                          Cancer Data
evangower/premier-league-2022-2023                             Premier League 2022-2023
lokeshparab/amazon-products-dataset                            Amazon Products Sales Dataset 2023
iammustafatz/diabetes-prediction-dataset                       Diabetes prediction dataset
ulrikthygepedersen/fastfood-nutrition                          Fastfood Nutrition
mikoajfish99/us-recession-and-financial-indicators             Financial Indicators of US Recession
desalegngeb/students-exam-scores                               Students Exam Scores: Extended Dataset
rkiattisak/student-performance-in-mathematics                  Student performance prediction
ppb00x/credit-risk-customers                                   credit_risk_customers
ritwikb3/heart-disease-cleveland                               Heart Disease Cleveland
ppb00x/country-gdp                                             Country_GDP
dansbecker/melbourne-housing-snapshot                          Melbourne Housing Snapshot
r1shabhgupta/google-stock-price-daily-weekly-and-monthly-2023  Google Stock Price: Daily, Weekly & Monthly (2023)
omartorres25/honda-data                                        Honda Cars Data
harshghadiya/kidneystone                                       Kidney Stone Dataset
ashishraut64/internet-users                                    Global Internet users
andrezaza/clapper-massive-rotten-tomatoes-movies-and-reviews   Massive Rotten Tomatoes Movies & Reviews
r1chardson/the-world-university-rankings-2011-2023             THE World University Rankings 2011-2023
```

```
#download the birdclef@2023 dataset in our local machine
! kaggle competitions download -c 'birdclef-2023'


#--below is to search datasets using keyword
#!kaggle datasets list -s birdclef-2023
```

```
Downloading birdclef-2023.zip to /content/drive/MyDrive/BirdClef2023
100% 4.90G/4.91G [00:41<00:00, 71.8MB/s]
100% 4.91G/4.91G [00:42<00:00, 125MB/s]
```

```
#Unzip the dataset
!unzip birdclef-2023.zip
```

```
          inflating: train_audio/yewgre1/XC700615.ogg
          inflating: train_audio/yewgre1/XC703472.ogg
          inflating: train_audio/yewgre1/XC703485.ogg
          inflating: train_audio/yewgre1/XC704433.ogg
```

```
import numpy as np # linear algebra
import pandas as pd #dataframes
```

```
import os
for dirname, _, filenames in os.walk('/content/drive/MyDrive/BirdClef2023'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/content/drive/MyDrive/BirdClef2023/train_audio/yewgre1/XC753190.ogg
```

```
import os
import random
import cv2
import librosa
import folium

import pandas as pd
import numpy as np
from scipy.linalg import norm

from IPython.display import Audio, display
from scipy.stats import zscore

import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from collections import Counter
import matplotlib.pyplot as plt
from IPython.display import Audio
import altair as alt


# defining some helper functions
def normalize(v):
    if norm(v) == 0:
        return v
    return norm(v)


tr_meta_df = pd.read_csv("/content/drive/MyDrive/BirdClef2023/train_metadata.csv")
```

```
tr_meta_df.shape
```

```
(16941, 12)
```

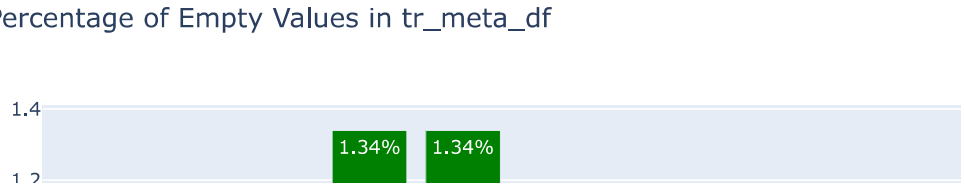
```
tr_meta_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16941 entries, 0 to 16940
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   primary_label     16941 non-null  object 
 1   secondary_labels  16941 non-null  object 
 2   type              16941 non-null  object 
 3   latitude          16714 non-null  float64
 4   longitude         16714 non-null  float64
 5   scientific_name   16941 non-null  object 
 6   common_name       16941 non-null  object 
 7   author            16941 non-null  object 
 8   license           16941 non-null  object 
 9   rating            16941 non-null  float64
 10  url               16941 non-null  object 
 11  filename          16941 non-null  object 
dtypes: float64(3), object(9)
memory usage: 1.6+ MB
```

```
tr_meta_df.head(3)
```

	primary_label	secondary_labels	type	latitude	longitude	scientific_name	commor
0	abethr1	[]	['song']	4.3906	38.2788	Turdus tephronotus	Africar eyed
1	abethr1	[]	['call']	-2.9524	38.2921	Turdus tephronotus	Africar eyed
2	abethr1	[]	['song']	-2.9524	38.2921	Turdus tephronotus	Africar eyed

```
import plotly.graph_objects as go
import pandas as pd

# Calculate percentage of empty values in each column
empty_pct = (tr_meta_df.isnull().sum() / len(tr_meta_df)) * 100

# Create a bar chart using Plotly
fig = go.Figure(data=[go.Bar(
    x=empty_pct.index,  # x-axis values
    y=empty_pct.values,  # y-axis values
    text=empty_pct.round(2).astype(str) + '%',  # text label with percentage
    textposition='auto',  # position of the text label
    marker=dict(color='green')  # set the color of the bar to green
)])

# Update the layout of the chart
fig.update_layout(
    title='Percentage of Empty Values in tr_meta_df',
    xaxis=dict(title='Columns'),
    yaxis=dict(title='Percentage of Empty Values')
)

# Display the chart
fig.show()
```

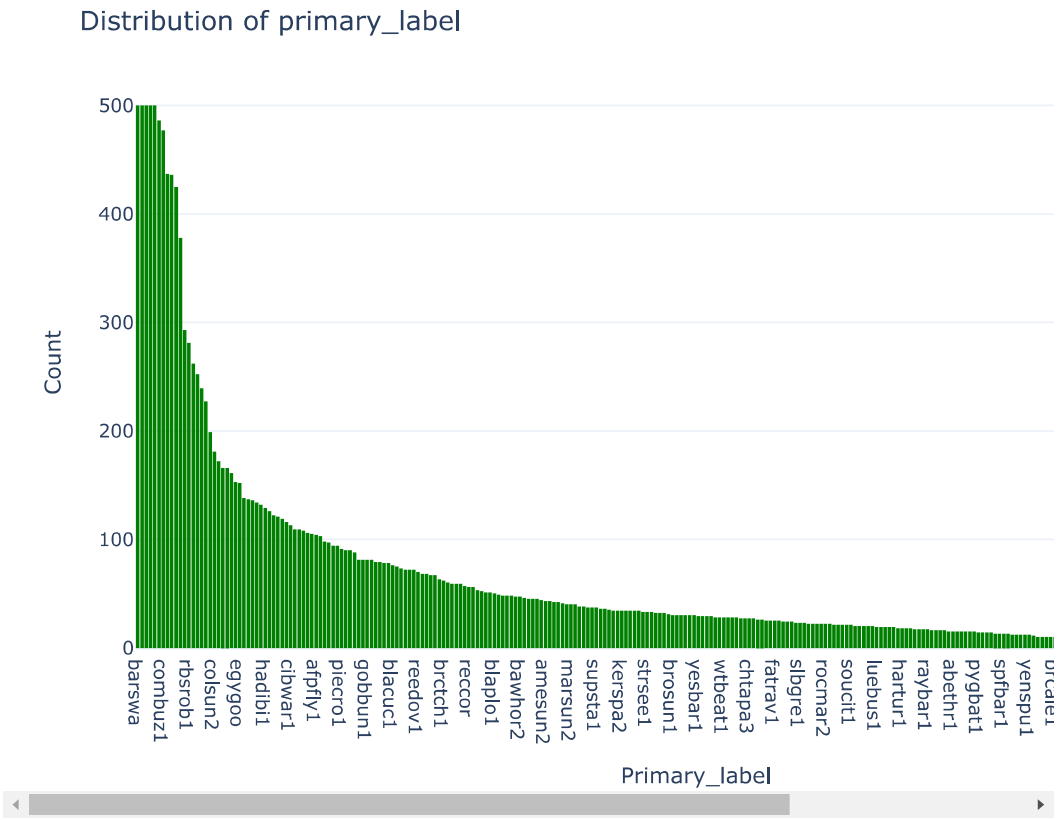



```
print(tr_meta_df['primary_label'].describe())
print(tr_meta_df['primary_label'].unique())
```

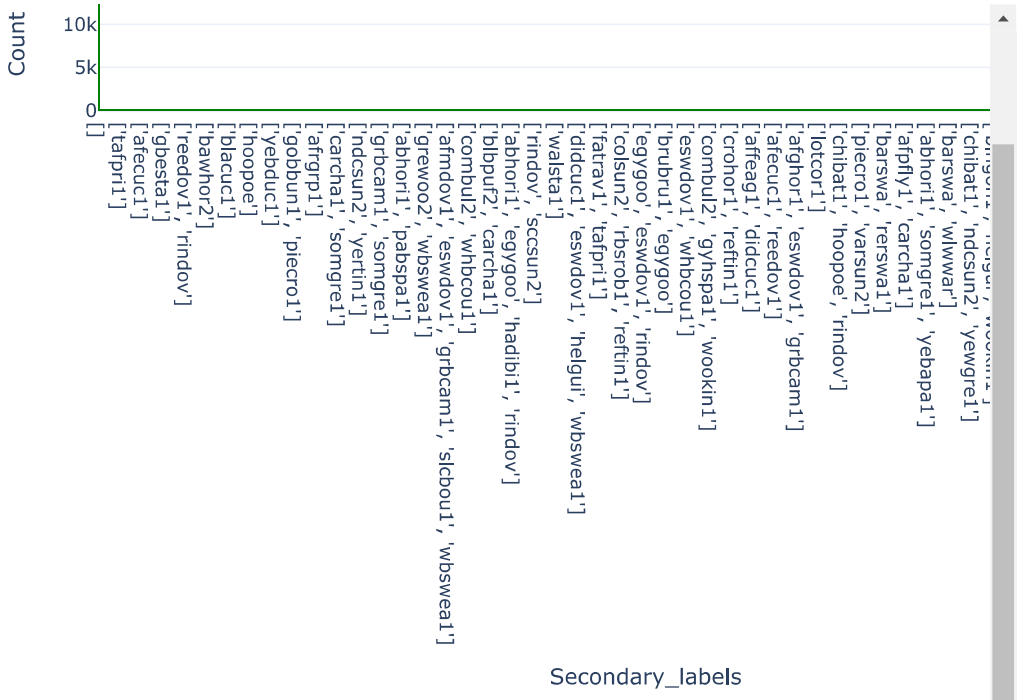
```
count      16941
unique       264
top       barswa
freq         500
Name: primary_label, dtype: object
['abethr1' 'abhori1' 'abythr1' 'afbfly1' 'afdfly1' 'afecuc1' 'affeag1'
 'afgfly1' 'afghor1' 'afmdov1' 'afpfly1' 'afpkin1' 'afpwag1' 'afrgos1'
 'afrgrp1' 'afrjac1' 'afrthr1' 'amesun2' 'augbuz1' 'bagwea1' 'barswa'
 'bawhor2' 'bawman1' 'bcbeat1' 'beasun2' 'bkctch1' 'bkfruw1' 'blacra1'
 'blacuc1' 'blakit1' 'blaplo1' 'blbpuf2' 'blcapa2' 'blfbus1' 'blhgon1'
 'blhher1' 'blksaw1' 'blnmou1' 'blnwea1' 'bltapa1' 'bltbar1' 'bltori1'
 'blwlap1' 'brcale1' 'brcsta1' 'brctch1' 'brcwea1' 'brican1' 'brobab1'
 'broman1' 'brosun1' 'brrwhe3' 'brtcha1' 'brubru1' 'brwwar1' 'bswdov1'
 'btweye2' 'bubwar2' 'butapa1' 'cabgre1' 'carcha1' 'carwoo1' 'categr'
 'ccbeat1' 'chespa1' 'chewea1' 'chibat1' 'chtapa3' 'chucis1' 'cibwar1'
 'cohmar1' 'colsun2' 'combul2' 'combuz1' 'comsan' 'crefra2' 'crheag1'
 'crohor1' 'darbar1' 'darter3' 'didcuc1' 'dotbar1' 'dutdov1' 'easmog1'
 'eaywag1' 'edcsun3' 'egygoo' 'equaka1' 'eswdov1' 'eubeat1' 'fatrav1'
 'fatwid1' 'fislov1' 'fotdro5' 'gabgos2' 'gargan' 'gbesta1' 'gnbcam2'
 'gnhsun1' 'gobbun1' 'gobsta5' 'gobwea1' 'golher1' 'grbcam1' 'grccra1'
 'grecor' 'greegr' 'grewoo2' 'grwpyt1' 'gryapa1' 'grywrw1' 'gybfis1'
 'gycwar3' 'gyhbus1' 'gyhkin1' 'gyhneg1' 'gyhspa1' 'gytbar1' 'hadibi1'
 'hamerk1' 'hartur1' 'helgui' 'hipbab1' 'hoopoe' 'huncis1' 'hunsun2'
 'joygre1' 'kerspa2' 'klacuc1' 'kvbsun1' 'laudov1' 'lawgol' 'lesmaw1'
 'lessts1' 'libeat1' 'litegr' 'litswi1' 'litwea1' 'loceag1' 'lotcor1'
 'lotlap1' 'luebus1' 'mabeat1' 'macshr1' 'malkin1' 'marsto1' 'marsun2'
 'mcptit1' 'meypar1' 'moccha1' 'mouwag1' 'ndcsun2' 'nobfly1' 'norbro1'
 'norcro1' 'norfis1' 'norpuf1' 'nubwoo1' 'pabspa1' 'palfly2' 'palpri1'
 'piecro1' 'piekin1' 'pitwhy' 'purgre2' 'pygbat1' 'quailf1' 'ratcis1'
 'raybar1' 'rbsrob1' 'rebfir2' 'rebhor1' 'reboxp1' 'reccor' 'reccuc1'
 'reedov1' 'refbar2' 'refcro1' 'reftin1' 'refwar2' 'rehblu1' 'rehwea1'
 'reisee2' 'rerswa1' 'rewsta1' 'rindov' 'rocmar2' 'rostur1' 'ruegls1'
 'rufcha2' 'sacibi2' 'sccsun2' 'scrcha1' 'scthon1' 'shesta1' 'sichor1'
 'sincis1' 'slbgre1' 'slcbou1' 'sltnig1' 'sobfly1' 'somgre1' 'somtit4'
 'soucit1' 'soufis1' 'spemou2' 'spepig1' 'spewea1' 'spfbar1' 'spfwea1'
 'spmthr1' 'spwlap1' 'squher1' 'strher' 'strsee1' 'stusta1' 'subbus1'
 'supsta1' 'tacsun1' 'tafpri1' 'tamdov1' 'thrnig1' 'trobou1' 'varsun2'
 'vibsta2' 'vilwea1' 'vimwea1' 'walsta1' 'wbgbir1' 'wbrcha2' 'wbswea1'
 'wfbeat1' 'whbcan1' 'whbcou1' 'whbcro2' 'whbtit5' 'whbwea1' 'whbwhe3'
 'whcpri2' 'whctur2' 'wheslf1' 'whhsaw1' 'whihel1' 'whrshr1' 'witswa1'
 'wlwwar' 'wookin1' 'woosan' 'wtbeat1' 'yebapa1' 'yebbar1' 'yebduc1'
 'yebere1' 'yebgre1' 'yebsto1' 'yeccan1' 'yefcan' 'yelbis1' 'yenspu1'
 'yertin1' 'yesbar1' 'yespet1' 'yetgre1' 'yewgre1']
```

```
def plot_distribution(df, column, nbins=50):
    ordered_values = df[column].value_counts().index.tolist()
    fig = px.histogram(df, x=column, nbins=nbins,
                       color_discrete_sequence=['green'])
    fig.update_layout(template='plotly_white',
                      title=f'Distribution of {column}',
                      xaxis_title=column.capitalize(),
                      yaxis_title='Count')
    fig.update_xaxes(type='category', categoryorder='array', categoryarray=ordered_values)
    fig.show()


def plot_distribution2(df, column, nbins=50):
    fig = px.histogram(df, x=column, nbins=nbins,
```

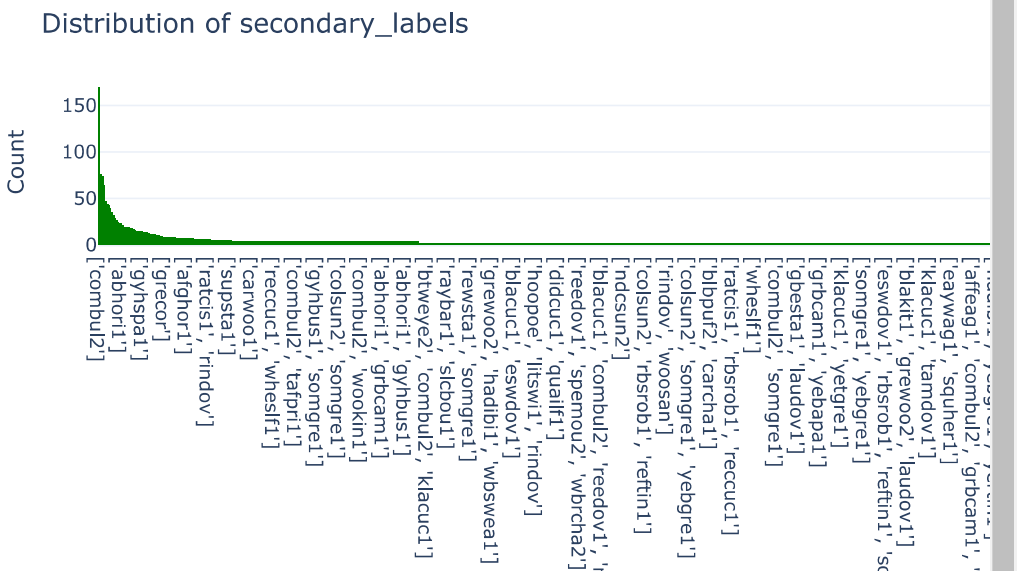
```
                      color_discrete_sequence=['green'])
    fig.update_layout(template='plotly_white',
                      title=f'Distribution of {column}',
                      xaxis_title=column.capitalize(),
                      yaxis_title='Count')
    fig.update_xaxes(type='category', categoryorder='array')
    fig.show()


plot_distribution(tr_meta_df,'primary_label')
```




```
# print(tr_meta_df['secondary_labels'].value_counts())

plot_distribution(tr_meta_df,'secondary_labels')

# removing the first one and then printing again

tr_meta_df_filtered = tr_meta_df.loc[tr_meta_df['secondary_labels'].apply(len) > 2]

# Call the plot_distribution function
print("distribution of secondary_labels modified with removal of []")
plot_distribution(tr_meta_df_filtered, 'secondary_labels')
```




```
distribution of secondary_labels modified with removal of []
```

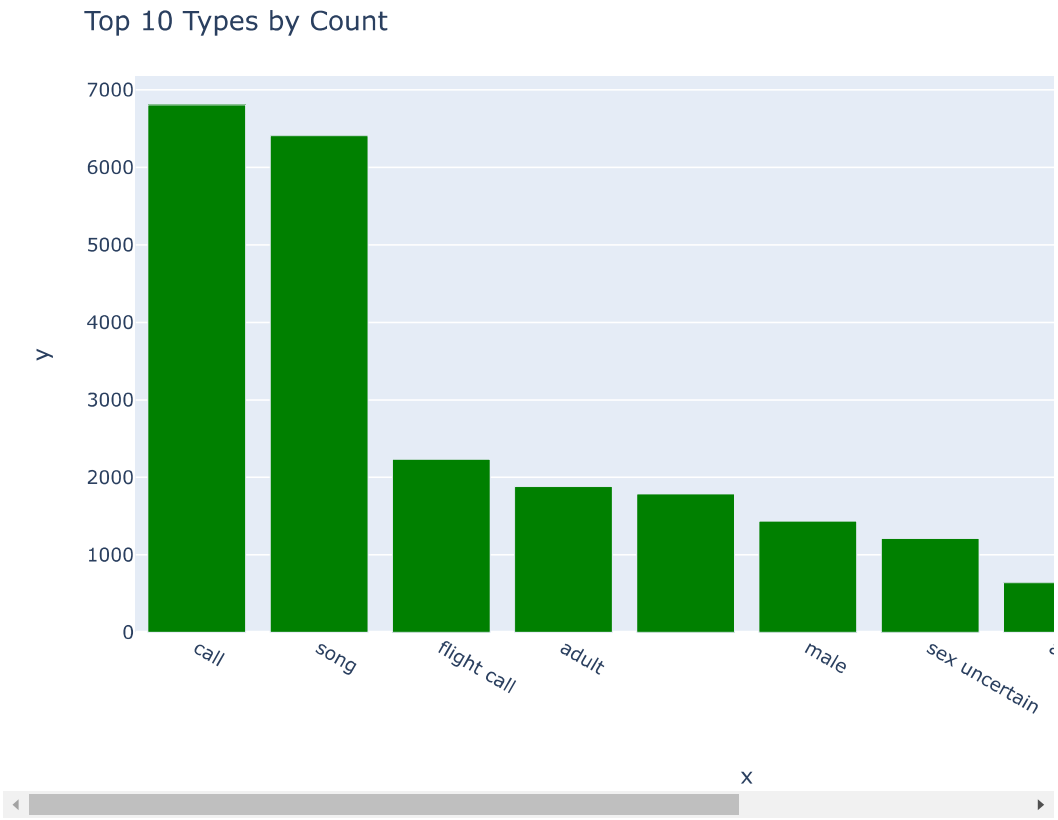



```
tr_meta_df["type"].head()
```

```
0            ['song']
1            ['call']
2            ['song']
3            ['song']
4    ['call', 'song']
Name: type, dtype: object
```

```
import ast
# Flatten the list of labels in the "type" column
labels = [label.strip("[]'") for sublist in tr_meta_df['type'].apply(ast.literal_eval) for label in sublist]

# Count the occurrence of each label
label_counts = Counter(labels)

# Select the top 10 labels
top_labels = dict(sorted(label_counts.items(), key=lambda item: item[1], reverse=True)[:10])

# Create a bar plot of the top 10 label counts
```

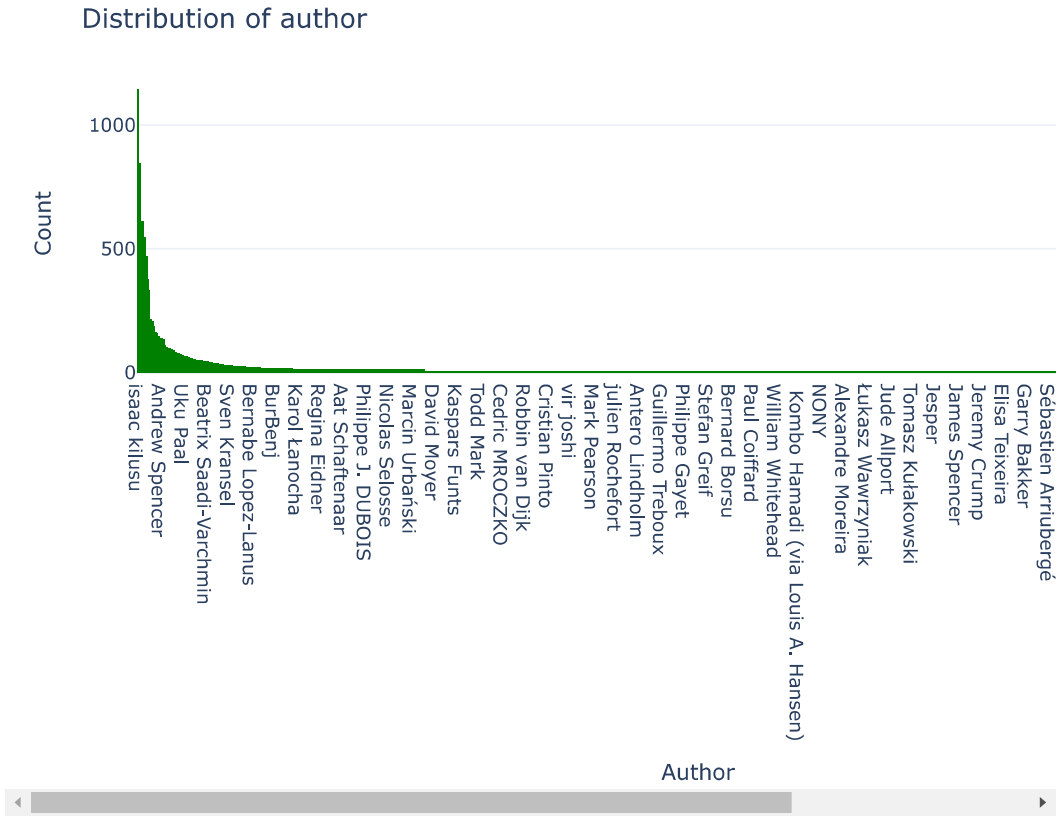
```
fig = px.bar(x=list(top_labels.keys()), y=list(top_labels.values()), color_discrete_sequence=['green'])
fig.update_layout(title_text="Top 10 Types by Count")
fig.show()
```




```
fig = px.density_mapbox(tr_meta_df, lat='latitude', lon='longitude',
                        radius=2, center=dict(lat=0, lon=180),
                        zoom=1, mapbox_style="stamen-terrain", color_continuous_scale='greens')

# set plot title
fig.update_layout(title='Heatmap')

# show the plot
fig.show()
```

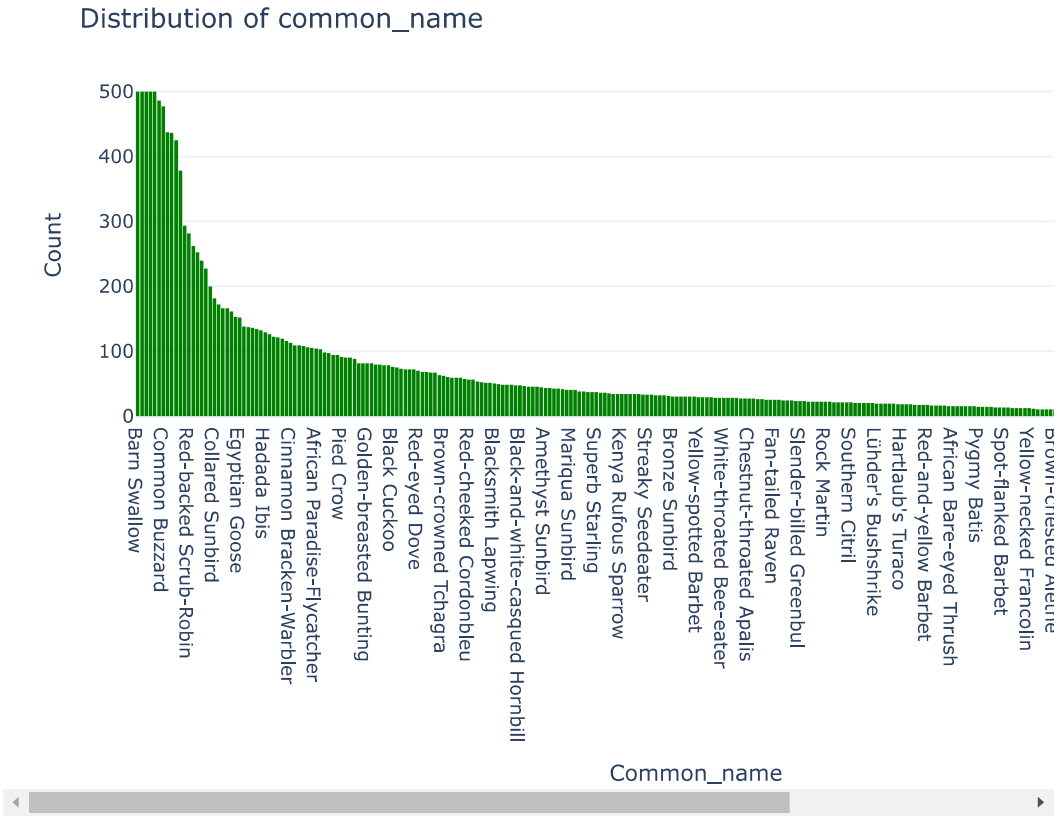
Heatmap

```
plot_distribution(tr_meta_df, 'author')
```

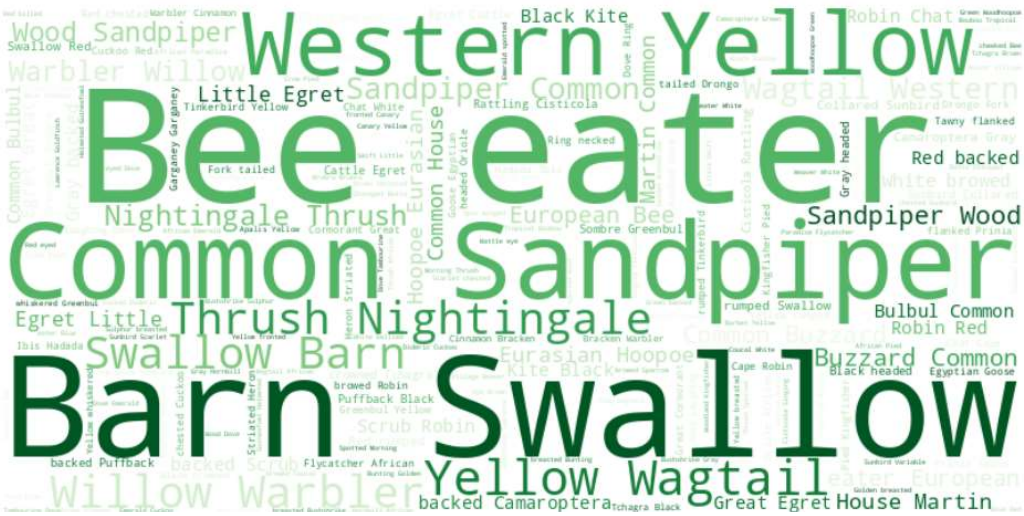



```
# Define your text to visualize
from wordcloud import WordCloud

# Create the WordCloud object with a green color
text = ' '.join(tr_meta_df['author'].astype(str).values.tolist())
wc = WordCloud(background_color='white', width=800, height=400, colormap='Greens').generate(text)

# Display the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```

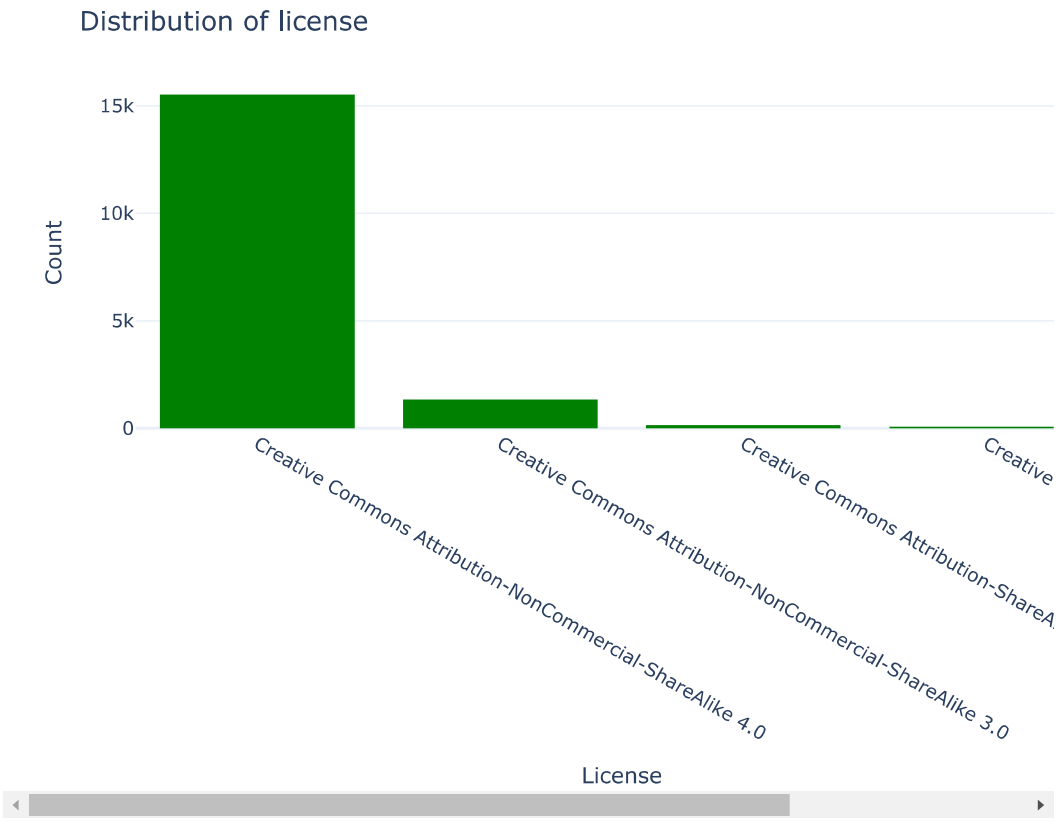

```
plot_distribution(tr_meta_df, 'common_name')
```




```
# Define your text to visualize
from wordcloud import WordCloud

# Create the WordCloud object with a green color
text = ' '.join(tr_meta_df['common_name'].astype(str).values.tolist())
wc = WordCloud(background_color='white', width=800, height=400, colormap='Greens').generate(text)

# Display the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```

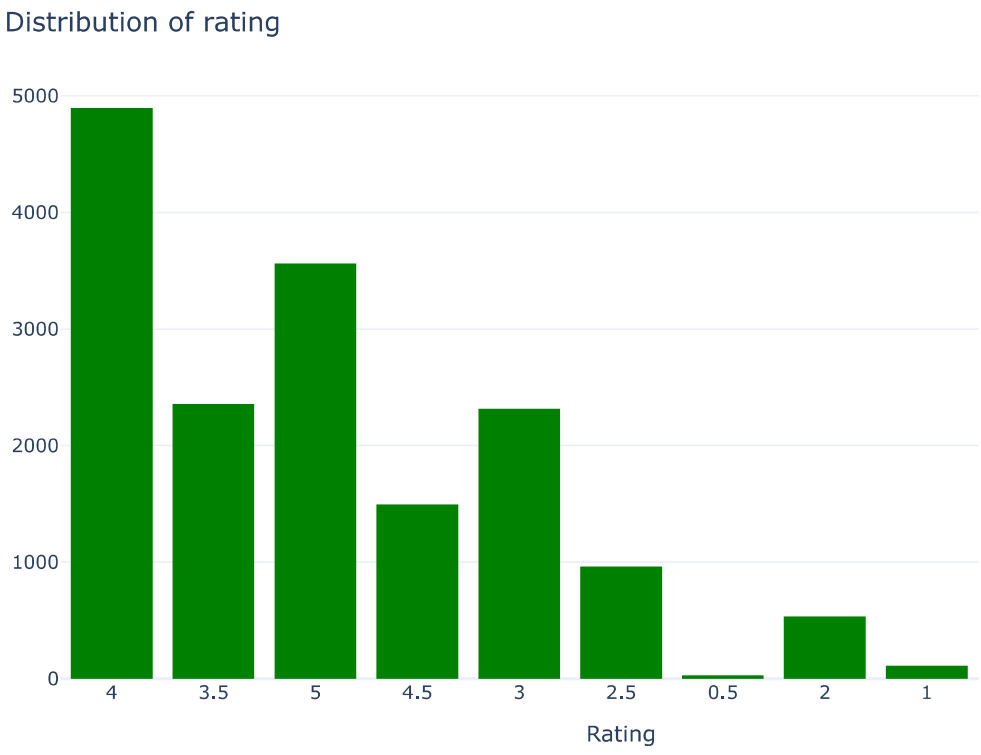



```
plot_distribution(tr_meta_df, 'license')
```



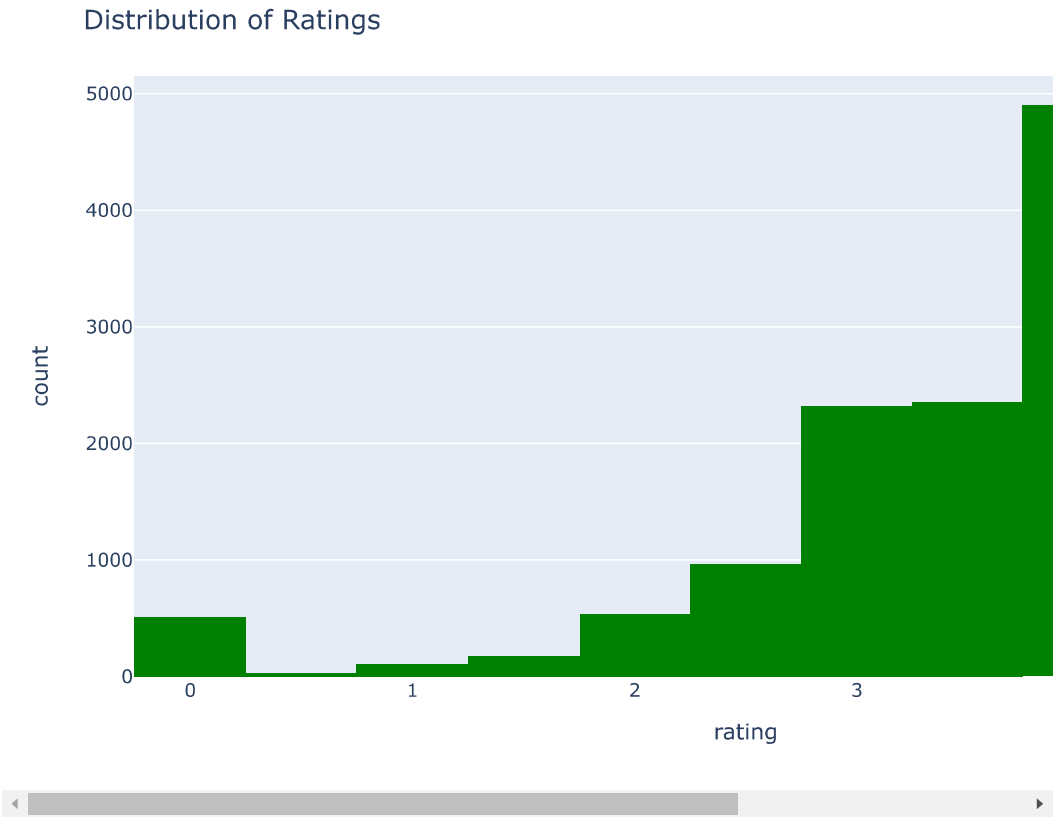

```
# Define your text to visualize
from wordcloud import WordCloud

# Create the WordCloud object with a green color
text = ' '.join(tr_meta_df['license'].astype(str).values.tolist())
wc = WordCloud(background_color='white', width=800, height=400, colormap='Greens').generate(text)

# Display the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```


```
plot_distribution2(tr_meta_df,'rating')
```




```
def create_boxplot(df, y_col, title):
    fig = px.box(df, y=y_col)
    fig.update_traces(marker_color='green')
    fig.update_layout(title=title)
    fig.show()


create_boxplot(tr_meta_df,'rating','Box Plot for Ratings')
```


```
fig = px.histogram(tr_meta_df, x="rating", nbins=len(tr_meta_df["rating"].unique()) , color_discrete_sequence=['green'])
fig.update_layout(title_text="Distribution of Ratings")
fig.show()
```



References:

https://www.kaggle.com/competitions/birdclef-2023/data

✓ 0s completed at 12:48 AM