odiversity-status-prediction-final

May 14, 2024

Project: Biodiversity Distribution Status Prediction of Animals,Plants and Natural Communities

[1]:
```
from IPython.display import Image
Image(filename='/content/
↪image-illustration-nature-durabilite-mode-vie-respectueux-environnement-conservation-art-co
↪jpg')
```

[1]:

Data Source:Data.gov

Data url: https://data.ny.gov/api/views/tk82-7km5/rows.csv?accessType=DOWNLOAD

About the dataset:The NYS Department of Environmental Conservation (DEC) collects and maintains several datasets on the locations, distribution and status of species of plants and animals. Information on distribution by county from the following three databases was extracted and compiled into this dataset. First, the New York Natural Heritage Program biodiversity database: Rare animals, rare plants, and significant natural communities. Significant natural communities are rare or high-quality wetlands, forests, grasslands, ponds, streams, and other types of habitats. Next, the 2nd NYS Breeding Bird Atlas Project database: Birds documented as breeding during the atlas project from 2000-2005. And last, DEC's NYS Reptile and Amphibian Database: Reptiles and amphibians; most records are from the NYS Amphibian & Reptile Atlas Project (Herp Atlas) from 1990-1999.

1 Problem Statement

The "Biodiversity Distribution Status Prediction of Animals, Plants, and Natural Communities" project seeks to develop a robust predictive model using classification algorithms. The model's objective is to accurately determine the distribution status of various species and habitats, including rare animals, rare plants, significant natural communities (such as wetlands, forests, grasslands, ponds, and streams), breeding birds, reptiles, and amphibians. This determination will be based on a comprehensive dataset sourced from the New York Natural Heritage Program, the NYS Breeding Bird Atlas Project (2000-2005), and DEC's NYS Reptile and Amphibian Database (1990-1999).

The primary goal is to leverage machine learning techniques to forecast the distribution status of these biological entities. By accurately predicting distribution patterns, the project aims to provide valuable insights that can support conservation efforts, facilitate environmental planning, and aid in making informed management decisions.

#Importing Libraries and Load dataset Importing essential libraries and load the dataset for insights and creating a data frame.

[2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
df=pd.read_csv('/content/
↪Biodiversity_by_County_-_Distribution_of_Animals__Plants_and_Natural_Communities.
↪csv')
df
```

[2]:
```
       County Category   Taxonomic Group Taxonomic Subgroup  \
0      Albany   Animal        Amphibians    Frogs and Toads
1      Albany   Animal        Amphibians    Frogs and Toads
2      Albany   Animal        Amphibians    Frogs and Toads
3      Albany   Animal        Amphibians    Frogs and Toads
4      Albany   Animal        Amphibians    Frogs and Toads
…        …        …              …                 …
20502   Yates     Plant  Flowering Plants             Sedges
20503   Yates     Plant  Flowering Plants             Sedges
20504   Yates     Plant  Flowering Plants             Sedges
20505   Yates     Plant  Flowering Plants             Sedges
20506   Yates     Plant            Mosses       Other Mosses

                Scientific Name                  Common Name  \
0           Anaxyrus americanus                American Toad
```

```
1              Anaxyrus fowleri                Fowler's Toad
2             Hyla versicolor                Gray Treefrog
3      Lithobates catesbeianus                     Bullfrog
4         Lithobates clamitans                   Green Frog
…                          …                            …
20502             Carex meadii                 Mead's Sedge
20503          Carex retroflexa              Reflexed Sedge
20504          Carex sartwellii             Sartwell's Sedge
20505           Carex straminea                  Straw Sedge
20506        Hyophila involuta  Rolled-leaf wet ground moss

      Year Last Documented      NY Listing Status Federal Listing Status  \
0                1990-1999  Game with open season             not listed
1                1990-1999  Game with open season             not listed
2                1990-1999  Game with open season             not listed
3                1990-1999  Game with open season             not listed
4                1990-1999  Game with open season             not listed
…                      …                      …                    …
20502        not available             Endangered             not listed
20503        not available             not listed             not listed
20504        not available             Endangered             not listed
20505        not available             Endangered             not listed
20506                 2005             not listed             not listed

      State Conservation Rank Global Conservation Rank     Distribution Status
0                          S5                       G5      Recently Confirmed
1                          S4                       G5      Recently Confirmed
2                          S5                       G5      Recently Confirmed
3                          S5                       G5      Recently Confirmed
4                          S5                       G5      Recently Confirmed
…                        …                       …                     …
20502                      SH                     G4G5  Historically Confirmed
20503                      S4                       G5  Historically Confirmed
20504                    S1S2                       G5  Historically Confirmed
20505                      S1                       G5  Historically Confirmed
20506                    S2S3                     G4G5      Recently Confirmed

[20507 rows x 12 columns]
```

2 Overview of the Dataset

Print first and last 5 lines of data using head() and tail() function respectively.

[3]:
```
df.head()
```

[3]:
```
    County Category Taxonomic Group Taxonomic Subgroup  \
0   Albany   Animal      Amphibians    Frogs and Toads
1   Albany   Animal      Amphibians    Frogs and Toads
2   Albany   Animal      Amphibians    Frogs and Toads
3   Albany   Animal      Amphibians    Frogs and Toads
4   Albany   Animal      Amphibians    Frogs and Toads

            Scientific Name    Common Name Year Last Documented  \
0      Anaxyrus americanus  American Toad            1990-1999
1         Anaxyrus fowleri  Fowler's Toad            1990-1999
2          Hyla versicolor  Gray Treefrog            1990-1999
3  Lithobates catesbeianus       Bullfrog            1990-1999
4     Lithobates clamitans     Green Frog            1990-1999

       NY Listing Status Federal Listing Status State Conservation Rank  \
0  Game with open season             not listed                      S5
1  Game with open season             not listed                      S4
2  Game with open season             not listed                      S5
3  Game with open season             not listed                      S5
4  Game with open season             not listed                      S5

  Global Conservation Rank Distribution Status
0                       G5  Recently Confirmed
1                       G5  Recently Confirmed
2                       G5  Recently Confirmed
3                       G5  Recently Confirmed
4                       G5  Recently Confirmed
```

[4]:
```
df.tail()
```

[4]:
```
       County Category   Taxonomic Group Taxonomic Subgroup    Scientific Name  \
20502   Yates    Plant  Flowering Plants             Sedges       Carex meadii
20503   Yates    Plant  Flowering Plants             Sedges   Carex retroflexa
20504   Yates    Plant  Flowering Plants             Sedges   Carex sartwellii
20505   Yates    Plant  Flowering Plants             Sedges    Carex straminea
20506   Yates    Plant            Mosses       Other Mosses  Hyophila involuta

                          Common Name Year Last Documented NY Listing Status  \
20502                    Mead's Sedge        not available        Endangered
20503                  Reflexed Sedge        not available        not listed
20504                Sartwell's Sedge        not available        Endangered
20505                     Straw Sedge        not available        Endangered
20506  Rolled-leaf wet ground moss                   2005        not listed

      Federal Listing Status State Conservation Rank Global Conservation Rank  \
20502             not listed                      SH                     G4G5
20503             not listed                      S4                       G5
```

```
20504          not listed                    S1S2                        G5
20505          not listed                      S1                        G5
20506          not listed                    S2S3                      G4G5

          Distribution Status
20502  Historically Confirmed
20503  Historically Confirmed
20504  Historically Confirmed
20505  Historically Confirmed
20506      Recently Confirmed
```

[5]:
```
df.shape
```

[5]:
```
(20507, 12)
```

Dataset contains 20507 rows and 12 columns respectively.

Dataset Information

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20507 entries, 0 to 20506
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   County                    20507 non-null  object
 1   Category                  20507 non-null  object
 2   Taxonomic Group           20507 non-null  object
 3   Taxonomic Subgroup        20507 non-null  object
 4   Scientific Name           20507 non-null  object
 5   Common Name               20507 non-null  object
 6   Year Last Documented      20507 non-null  object
 7   NY Listing Status         20507 non-null  object
 8   Federal Listing Status    20507 non-null  object
 9   State Conservation Rank   20507 non-null  object
 10  Global Conservation Rank  20507 non-null  object
 11  Distribution Status       20507 non-null  object
dtypes: object(12)
memory usage: 1.9+ MB
```

3 Header details

Print column names of dataframe.

[7]:
```
df.columns
```

```
[7]: Index(['County', 'Category', 'Taxonomic Group', 'Taxonomic Subgroup',
       'Scientific Name', 'Common Name', 'Year Last Documented',
       'NY Listing Status', 'Federal Listing Status',
       'State Conservation Rank', 'Global Conservation Rank',
       'Distribution Status'],
      dtype='object')
```

4 Datatype

Check datatypes for each columns

```
[8]: df.dtypes
```

```
[8]: County                       object
     Category                     object
     Taxonomic Group              object
     Taxonomic Subgroup           object
     Scientific Name              object
     Common Name                  object
     Year Last Documented         object
     NY Listing Status            object
     Federal Listing Status       object
     State Conservation Rank      object
     Global Conservation Rank     object
     Distribution Status          object
     dtype: object
```

- All columns are of category datas.

5 Unique Values

Get unique values of each column

```
[9]: for col in df:
       value=df[col].unique()
       print(col,":",value)
```

```
County : ['Albany' 'Dutchess' 'Fulton' 'Allegany'
 'Atlantic Ocean and Long Island Sound' 'Bronx' 'Erie' 'Broome'
 'Cattaraugus' 'Cayuga' 'Chautauqua' 'Chemung' 'Chenango' 'Clinton'
 'Essex' 'Columbia' 'Cortland' 'Counties Unknown' 'Delaware' 'Franklin'
 'Genesee' 'Greene' 'Hamilton' 'Herkimer' 'Jefferson' 'Kings'
 'Lake Erie Open Waters' 'Lake Ontario Open Waters' 'Lewis' 'Livingston'
 'Madison' 'Monroe' 'Montgomery' 'Nassau' 'New York' 'Niagara' 'Oneida'
 'Onondaga' 'Orange' 'Ontario' 'Orleans' 'Oswego' 'Otsego' 'Putnam'
 'Queens' 'Rensselaer' 'Richmond' 'Rockland' 'Saratoga' 'Schenectady'
 'Schoharie' 'Schuyler' 'Seneca' 'St. Lawrence' 'Steuben' 'Suffolk'
```

```
 'Sullivan' 'Tioga' 'Tompkins' 'Ulster' 'Warren' 'Washington' 'Wayne'
 'Westchester' 'Wyoming' 'Yates']
Category : ['Animal' 'Natural Community' 'Plant']
Taxonomic Group : ['Amphibians' 'Birds' 'Animal Assemblages' 'Bees, Wasps and
Ants'
 'Butterflies and Moths' 'Dragonflies and Damselflies' 'Fish' 'Mammals'
 'Mussels and Clams' 'Other Animals' 'Reptiles'
 'Freshwater Nontidal Wetlands' 'Tidal Wetlands' 'Uplands' 'Conifers'
 'Ferns and Fern Allies' 'Flowering Plants' 'Mosses' 'Marine' 'Beetles'
 'Rivers and Streams' 'Lakes and Ponds' 'Flies' 'Snails' 'Stoneflies'
 'Subterranean' 'Mayflies']
Taxonomic Subgroup : ['Frogs and Toads' 'Herons, Bitterns, Egrets, Pelicans'
'Salamanders'
 'Nuthatches' 'Animal Assemblages' 'Bees' 'Blackbirds and Orioles'
 'Cardinals and Buntings' 'Chickadees and Titmice' 'Cormorants' 'Creepers'
 'Crows and Jays' 'Cuckoos' 'Ducks, Geese, Waterfowl'
 'Finches and Crossbills' 'Flycatchers' 'Gnatcatchers' 'Grebes'
 'Grouse, Pheasants, Turkeys' 'Gulls, Terns, Plovers, Shorebirds'
 'Hawks, Falcons, Eagles, Vultures' 'Hummingbirds and Swifts'
 'Kingfishers' 'Kinglets' 'Mockingbirds and Thrashers' 'Nightbirds'
 'Old World Sparrows' 'Owls' 'Pigeons and Doves' 'Rails, Coots and Cranes'
 'Sparrows and Towhees' 'Starlings' 'Swallows' 'Thrushes and Bluebirds'
 'Vireos' 'Waxwings' 'Woodpeckers' 'Wood-Warblers' 'Wrens'
 'Butterflies and Skippers' 'Moths' 'Dragonflies'
 'Minnows, Shiners, Suckers' 'Sturgeons and Paddlefish' 'Bats' 'Rodents'
 'Freshwater Mussels' 'Other Animals' 'Snakes'
 'Forested Mineral Soil Wetlands' 'Open Mineral Soil Wetlands'
 'Intertidal Wetlands' 'Subtidal Wetlands' 'Barrens and Woodlands'
 'Forested Uplands' 'Open Uplands' 'Conifers' 'Ferns'
 'Asters, Goldenrods and Daisies' 'Grasses' 'Orchids'
 'Other Flowering Plants' 'Rushes' 'Sedges' 'Other Mosses' 'Larks'
 'Catfishes' 'Darters and Sunfishes' 'Lampreys' 'Forested Peatlands'
 'Open Peatlands' 'Whales and Dolphins' 'Marine Subtidal'
 'Carrion Beetles' 'Parrots and Parakeets' 'Needlefishes' 'Lizards'
 'Marine Intertidal' 'Clubmosses' 'Quillworts' 'Damselflies'
 'Rabbits and Hares' 'Natural Rivers and Streams' 'Loons' 'Perches'
 'Salmon and Trout' 'Sculpins' 'Natural Lakes and Ponds' 'Mooneyes'
 'Carnivores' 'Flies' 'Shrikes' 'Snails' 'Stoneflies' 'Horsetails'
 'Herrings and Shad' 'Peat Mosses' 'Lady Beetles' 'Shrews and Moles'
 'Natural Caves' 'Mayflies' 'Silversides' 'Rove Beetles' 'Killifishes'
 'Diving Beetles']
Scientific Name : ['Anaxyrus americanus' 'Anaxyrus fowleri' 'Hyla versicolor'
…
 'Philonotis capillaris' 'Tortula pagorum' 'Calopteryx dimidiata']
Common Name : ['American Toad' "Fowler's Toad" 'Gray Treefrog' … 'Hairy Apple
Moss'
 'Leafy screw moss' 'Sparkling Jewelwing']
Year Last Documented : ['1990-1999' '2006' '2019' 'not available' '2000-2005'
```

```
'1986' '2018'
 '1991' '1987' '2009' '2012' '1983' '2002' '2021' '2016' '1984' '1992'
 '2013' '2017' '2015' '1963' '1970' '2014' '1990' '2008' '1926' '1874'
 '2010' '1890' '1904' '1960' '1998' '1988' '2000' '2003' '2001' '1999'
 '2020' '1997' '1996' '1962' '1959' '1980' '1932' '1937' '1933' '1907'
 '1910' '1928' '1936' '1951' '1995' '1939' '1923' '1920' '2004' '1948'
 '1955' '1974' '1835' '1865' '1942' '1919' '1957' '1950' '2005' '2007'
 '1981' '1989' '1994' '1913' '1897' '1946' '1906' '1953' '1918' '1895'
 '1893' '1894' '1954' '1899' '1947' '1898' '1843' '1840' '1892' '1896'
 '1940' '1915' '1879' '1880' '1900' '1901' '1966' '1891' '1882' '1938'
 '2011' '1982' '1889' '1885' '1929' '1930' '1975' '1977' '1927' '1943'
 '1958' '1873' '1931' '1924' '1921' '1934' '1985' '1922' '1941' '1917'
 '1916' '1993' '1877' '1887' '1979' '1878' '1845' '1956' '1949' '1935'
 '1869' '1965' '1902' '1976' '1969' '1851' '1875' '1971' '1867' '1868'
 '1861' '1863' '1888' '1834' '1908' '1862' '1871' '1967' '1856' '1905'
 '1914' '1912' '1964' '1864' '1952' '1909' '1846' '1870' '1842' '1841'
 '1883' '1881' '1800' '1911' '1903' '1831' '1830' '1945' '1886' '1925'
 '1961' '1811' '1817' '1872' '1838' '1823' '1944' '1837' '1884' '1854'
 '1978' '1876' '1853' '1836' '1857' '1968' '1832' '1815' '1860' '1973']
NY Listing Status : ['Game with open season' 'Special Concern' 'Threatened'
 'Game with no open season' 'Endangered' 'Protected Bird' 'not applicable'
 'not listed' 'Protected Bird - Game with open season'
 'Protected - no open season' 'Rare']
Federal Listing Status : ['not listed' 'not applicable' 'Endangered'
'Threatened'
 'Proposed Threatened' 'Proposed Endangered']
State Conservation Rank : ['S5' 'S4' 'S2S3' 'S3B,S1N' 'SNA' 'S3' 'S1S2' 'S3S4'
'SNRN' 'SH' 'S1'
 'S5B' 'S4B' 'S2S3B,SNRN' 'S3B,SNRN' 'S3B' 'S3S4B,S3N' 'S3B,S3N'
 'S2S3B,S2N' 'S3S4B' 'S1B' 'S2?B' 'S2B' 'S3?B' 'SU' 'S1S3' 'S2S4' 'S2?'
 'S2S3B' 'S2' 'S4S5' 'SX' 'S1N' 'S1?' 'S3?' 'S3S4N' 'SNAB,S3N' 'SNAB'
 'SNR' 'SHB,S1N' 'SNRB' 'S2S3M' 'S1B,S3?N']
Global Conservation Rank : ['G5' 'G4G5' 'GU' 'GNR' 'G2' 'G3G4' 'GNA' 'G4' 'G2G3'
'G3' 'G5T1T3' 'G5T1'
 'G1G2' 'G4T2' 'G5T5' 'G4?' 'G5?' 'G5?T3' 'G5TNR' 'G5T4T5' 'G5T3T5'
 'G4?T4?' 'G4T4' 'G5T4?' 'G5T2' 'G5T4' 'G5T5?' 'G1' 'GUT1Q' 'G5?T4T5'
 'G5T3' 'G3G5' 'G3?' 'G4G5T4' 'G5?TNR' 'G5?T4?' 'G5?T3T5' 'G4T1T3'
 'G4G5T4T5' 'G3T1' 'GH' 'G5TNRQ' 'G5T3T4' 'G5T3?' 'G4G5Q' 'G2?' 'G3G4T2'
 'G4T3' 'G5T2T4' 'G1Q' 'G5T5?Q' 'G4Q' 'G4G5T3?Q' 'G4G5T3?' 'G3T3' 'G3Q'
 'G2G3T1T2' 'G3T1T3' 'GNRT4?' 'GNRTNR' 'GXQ' 'G5T1T2' 'G5T4Q']
Distribution Status : ['Recently Confirmed' 'Possible but not Confirmed'
 'Historically Confirmed' 'Extirpated']
```

6 Description

Displaying Description of the object data.

[10]:
```python
df.describe(include='O')
```

[10]:
```
         County Category Taxonomic Group      Taxonomic Subgroup  \
count     20507    20507           20507                   20507
unique       66        3              27                     105
top     Suffolk   Animal           Birds  Other Flowering Plants
freq        733    13611            9678                    3185

                  Scientific Name       Common Name Year Last Documented  \
count                       20507             20507                20507
unique                       1582              1578                  187
top     Lasionycteris noctivagans  Great Blue Heron            2000-2005
freq                           62                62                 8939

       NY Listing Status Federal Listing Status State Conservation Rank  \
count              20507                  20507                   20507
unique                11                      6                      43
top       Protected Bird             not listed                     S5B
freq                7011                  19220                    4189

       Global Conservation Rank Distribution Status
count                     20507               20507
unique                       63                   4
top                          G5  Recently Confirmed
freq                      15062               16127
```

7 Missing Values

Check the missing values in the dataset using isna().sum() and get a final summation results.

[11]:
```python
df.isna().sum()
```

[11]:
```
County                      0
Category                    0
Taxonomic Group             0
Taxonomic Subgroup          0
Scientific Name             0
Common Name                 0
Year Last Documented        0
NY Listing Status           0
Federal Listing Status      0
State Conservation Rank     0
Global Conservation Rank    0
Distribution Status         0
dtype: int64
```

Missing Values:- * Zero missing values

8 Find value counts of each columns

Find value counts for column ‘Distribution Status’

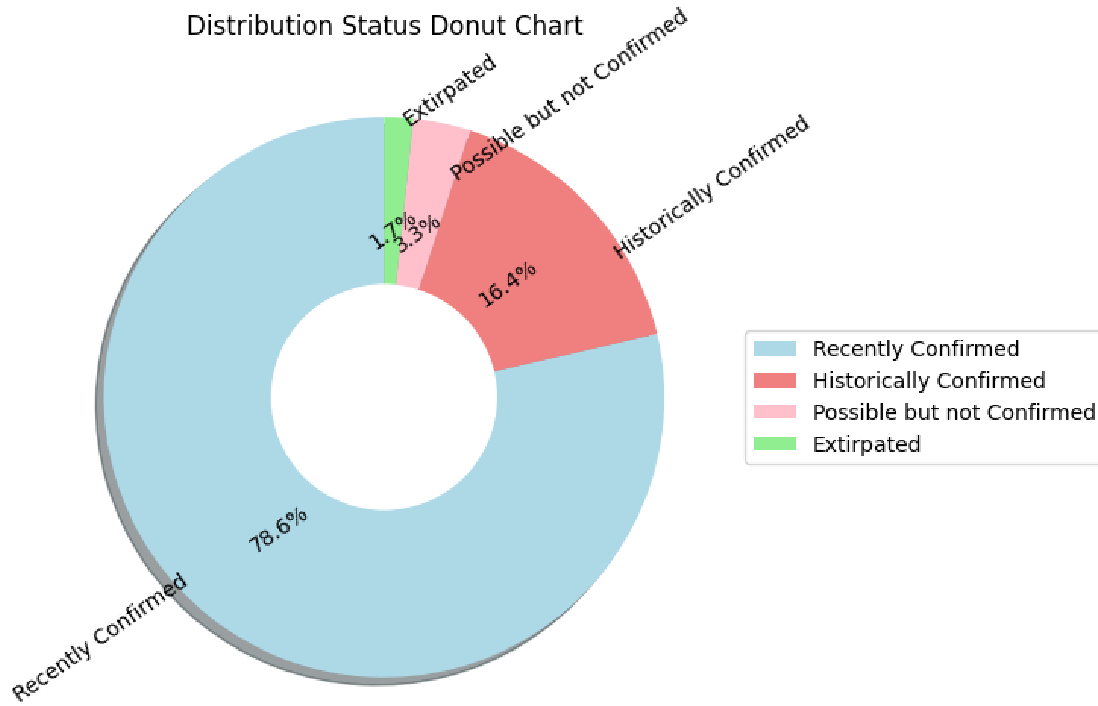
[12]:
```
df['Distribution Status'].value_counts()
```

[12]:
```
Distribution Status
Recently Confirmed            16127
Historically Confirmed         3363
Possible but not Confirmed      675
Extirpated                      342
Name: count, dtype: int64
```

9 Visualization

Visualize the column using Donut Chart.

[13]:
```
# Count the occurrences of each Distribution Status
counts = df['Distribution Status'].value_counts()

# Plotting
plt.figure(figsize=(6, 6))  # Increase the figure size to avoid label␣
 ↪overlapping
plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90,
        colors=['lightblue','lightcoral','pink','lightgreen'], shadow=True,
        textprops={'rotation': 35})

# Draw a circle at the center to create a donut chart
circle = plt.Circle((0, 0), 0.4, color='white')
plt.gca().add_artist(circle)

# Add a title
plt.title('Distribution Status Donut Chart')
# Position the percentage labels outside the chart
plt.legend(bbox_to_anchor=(1, 0.5), loc='center left')
# Show the plot
plt.show()
```



Observation:- The target column is the 'Distribution status' and it consists of 4 classes of data as Recently Confirmed, Historically Confirmed,Possible but not Confirmed and Extirpated.Clearly we can see its set of imbalanced data and can be treated before train test split step.

[14]:
```
df['County'].value_counts()
```

[14]:
```
County
Suffolk                                733
Essex                                  489
Orange                                 468
Nassau                                 466
Ulster                                 459
                                      …
New York                               161
Atlantic Ocean and Long Island Sound    14
Lake Ontario Open Waters                 4
Counties Unknown                         3
Lake Erie Open Waters                    1
Name: count, Length: 66, dtype: int64
```

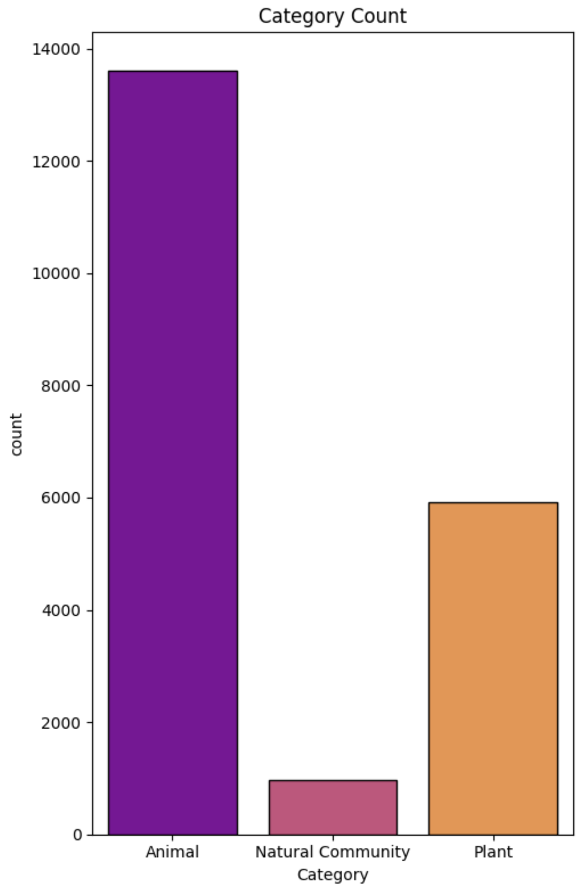
[15]:
```
category=df['Category'].value_counts()
category
```

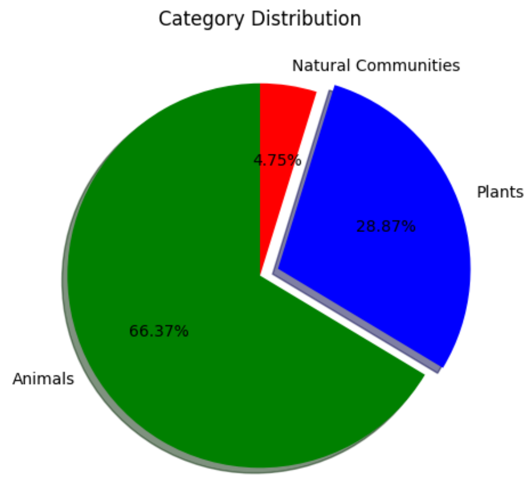
[15]:

```
Category
Animal               13611
Plant                 5921
Natural Community      975
Name: count, dtype: int64
```

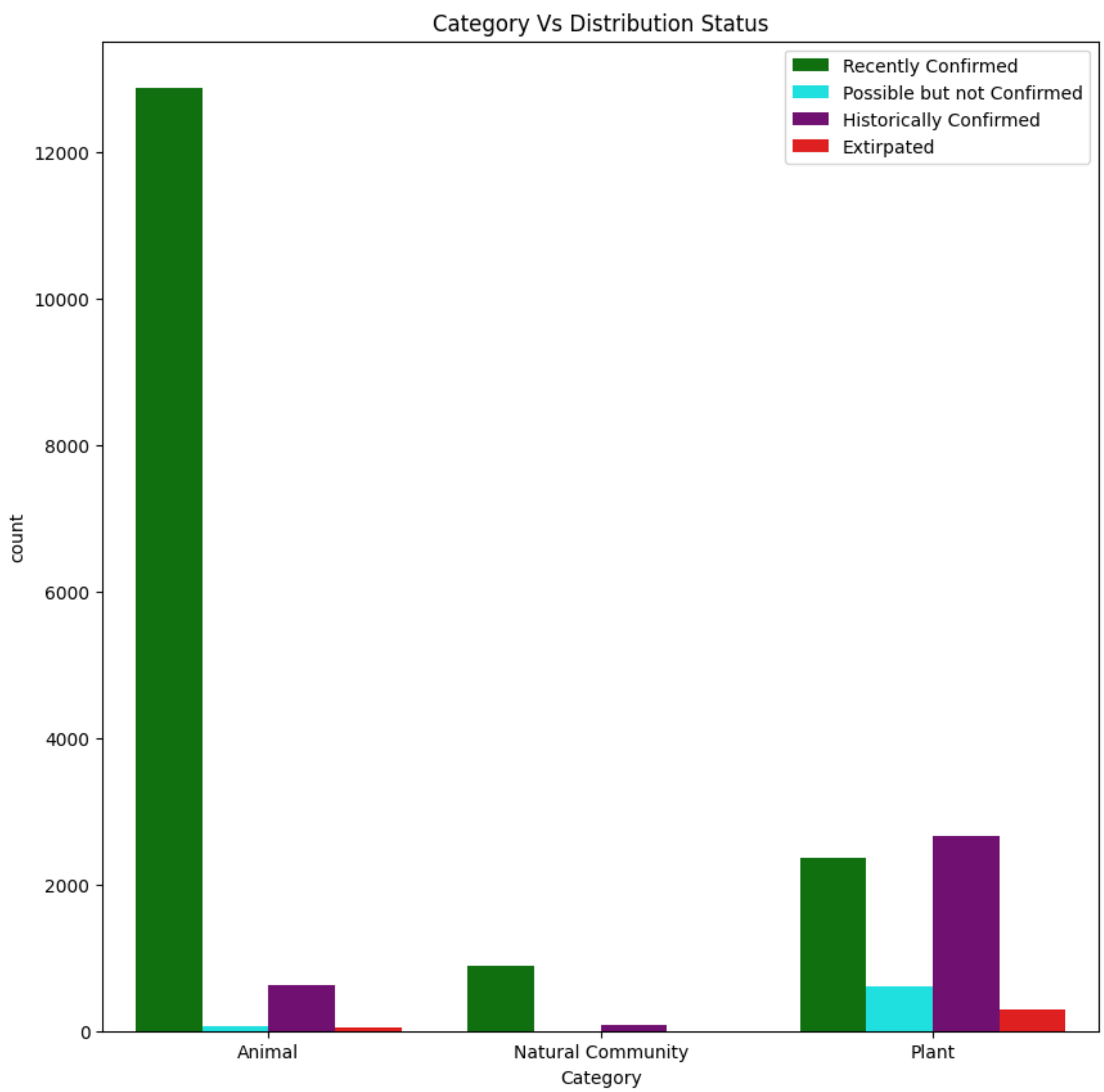
[16]:

```python
# Create figure and axes for subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 8))

# Plot the countplot with 'Category' as hue and specified colors
sns.countplot(x='Category', hue='Category', data=df, palette='plasma',
              edgecolor='black', ax=ax1, legend=False)
ax1.set_title("Category Count")
# Plot 2 - Seaborn Pie Chart
labels = ["Animals", "Plants", "Natural Communities"]
colors = ['green', 'blue', 'red']
explode = [0, 0.1, 0]  # Explode the "Plants" category
ax2.pie(df['Category'].value_counts(), colors=colors, labels=labels,
        autopct='%1.2f%%', explode=explode, startangle=90, shadow=True)
ax2.set_title("Category Distribution")

plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```





[17]:
```
# Category Vs Distribution Status
plt.figure(figsize=(10, 10))
sns.countplot(x='Category', hue='Distribution Status', data=df,
    palette=["green","cyan","purple","red"])
plt.legend(loc='upper right')  # Show legend
plt.title("Category Vs Distribution Status")
plt.show()
```



Observations:- * Animals ('Animal') have the highest count among the categories, followed by Plants ('Plant') and Natural Communities ('Natural Community'). * The majority of records fall under the 'Recently Confirmed' status in the Distribution Status column, indicating recent confirmations of observations.

[18]:
```
df['Common Name'].value_counts()
```

[18]:
```
Common Name
Great Blue Heron        62
Song Sparrow            62
Rock Pigeon             62
House Sparrow           62
Northern Mockingbird    62
                        ..
```

```
River Redhorse          1
Northern Riffleshell    1
Northern Holly Fern     1
Midwestern Hops         1
Sparkling Jewelwing     1
Name: count, Length: 1578, dtype: int64
```

[19]:
```
df['Common Name'].unique()
```

[19]:
```
array(['American Toad', "Fowler's Toad", 'Gray Treefrog', …,
       'Hairy Apple Moss', 'Leafy screw moss', 'Sparkling Jewelwing'],
      dtype=object)
```

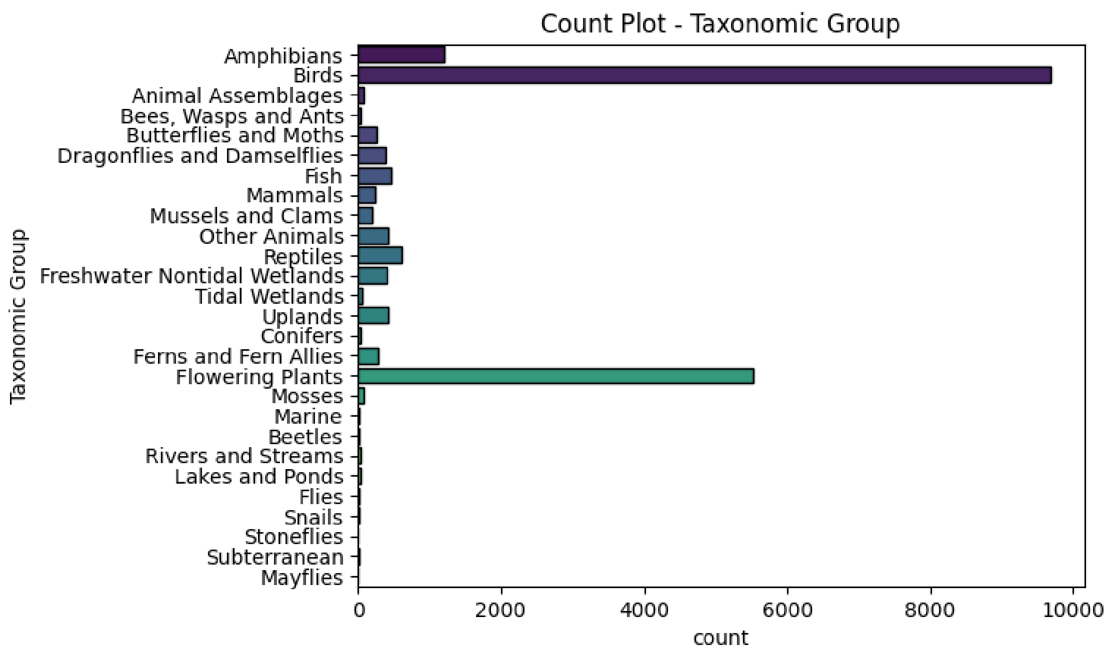
'Scientific Name' can be dropped as 'Common Name' is only required.

[20]:
```
total=df['Taxonomic Group'].value_counts()
total
```

[20]:
```
Taxonomic Group
Birds                           9678
Flowering Plants                5535
Amphibians                      1193
Reptiles                         600
Fish                             459
Other Animals                    418
Uplands                          417
Freshwater Nontidal Wetlands     396
Dragonflies and Damselflies      382
Ferns and Fern Allies            276
Butterflies and Moths            264
Mammals                          249
Mussels and Clams                206
Animal Assemblages                83
Mosses                            77
Tidal Wetlands                    61
Lakes and Ponds                   41
Rivers and Streams                37
Bees, Wasps and Ants              34
Conifers                          33
Beetles                           20
Marine                            16
Snails                            11
Flies                             10
Subterranean                       7
Stoneflies                         2
Mayflies                           2
Name: count, dtype: int64
```

[21]:
```
sns.countplot(y=df['Taxonomic Group'],data=df,edgecolor='k',palette='viridis')
plt.title('Count Plot - Taxonomic Group')
```

[21]:
```
Text(0.5, 1.0, 'Count Plot - Taxonomic Group')
```



[22]:
```
df['Taxonomic Subgroup'].value_counts()
```

[22]:
```
Taxonomic Subgroup
Other Flowering Plants              3185
Wood-Warblers                       1402
Sedges                              1097
Hawks, Falcons, Eagles, Vultures     703
Salamanders                          644
                                    ...
Stoneflies                             2
Mayflies                               2
Rove Beetles                           1
Killifishes                            1
Diving Beetles                         1
Name: count, Length: 105, dtype: int64
```

[23]:
```
df['Taxonomic Subgroup'].unique()
```

[23]:
```
array(['Frogs and Toads', 'Herons, Bitterns, Egrets, Pelicans',
       'Salamanders', 'Nuthatches', 'Animal Assemblages', 'Bees',
       'Blackbirds and Orioles', 'Cardinals and Buntings',
```

```
'Chickadees and Titmice', 'Cormorants', 'Creepers',
'Crows and Jays', 'Cuckoos', 'Ducks, Geese, Waterfowl',
'Finches and Crossbills', 'Flycatchers', 'Gnatcatchers', 'Grebes',
'Grouse, Pheasants, Turkeys', 'Gulls, Terns, Plovers, Shorebirds',
'Hawks, Falcons, Eagles, Vultures', 'Hummingbirds and Swifts',
'Kingfishers', 'Kinglets', 'Mockingbirds and Thrashers',
'Nightbirds', 'Old World Sparrows', 'Owls', 'Pigeons and Doves',
'Rails, Coots and Cranes', 'Sparrows and Towhees', 'Starlings',
'Swallows', 'Thrushes and Bluebirds', 'Vireos', 'Waxwings',
'Woodpeckers', 'Wood-Warblers', 'Wrens',
'Butterflies and Skippers', 'Moths', 'Dragonflies',
'Minnows, Shiners, Suckers', 'Sturgeons and Paddlefish', 'Bats',
'Rodents', 'Freshwater Mussels', 'Other Animals', 'Snakes',
'Forested Mineral Soil Wetlands', 'Open Mineral Soil Wetlands',
'Intertidal Wetlands', 'Subtidal Wetlands',
'Barrens and Woodlands', 'Forested Uplands', 'Open Uplands',
'Conifers', 'Ferns', 'Asters, Goldenrods and Daisies', 'Grasses',
'Orchids', 'Other Flowering Plants', 'Rushes', 'Sedges',
'Other Mosses', 'Larks', 'Catfishes', 'Darters and Sunfishes',
'Lampreys', 'Forested Peatlands', 'Open Peatlands',
'Whales and Dolphins', 'Marine Subtidal', 'Carrion Beetles',
'Parrots and Parakeets', 'Needlefishes', 'Lizards',
'Marine Intertidal', 'Clubmosses', 'Quillworts', 'Damselflies',
'Rabbits and Hares', 'Natural Rivers and Streams', 'Loons',
'Perches', 'Salmon and Trout', 'Sculpins',
'Natural Lakes and Ponds', 'Mooneyes', 'Carnivores', 'Flies',
'Shrikes', 'Snails', 'Stoneflies', 'Horsetails',
'Herrings and Shad', 'Peat Mosses', 'Lady Beetles',
'Shrews and Moles', 'Natural Caves', 'Mayflies', 'Silversides',
'Rove Beetles', 'Killifishes', 'Diving Beetles'], dtype=object)
```

The ‘Taxonomic Group’ and ‘Taxonomic Subgroup’ columns have similar labels and one of them can be dropped.

[24]:
```
df['Federal Listing Status'].value_counts()
```

[24]:
```
Federal Listing Status
not listed             19220
not applicable          1058
Threatened               142
Endangered                84
Proposed Threatened        2
Proposed Endangered        1
Name: count, dtype: int64
```

Observation: Federal listing status contains samples with not listed and not applicable with values 19220 and 1058 respectively. so can be dropped.

[25]:
```python
df['NY Listing Status'].unique()
```

[25]:
```
array(['Game with open season', 'Special Concern', 'Threatened',
       'Game with no open season', 'Endangered', 'Protected Bird',
       'not applicable', 'not listed',
       'Protected Bird - Game with open season',
       'Protected - no open season', 'Rare'], dtype=object)
```

[26]:
```python
df['NY Listing Status'].value_counts()
```

[26]:
```
NY Listing Status
Protected Bird                            7011
Endangered                                2740
Threatened                                2531
not listed                                2001
Game with no open season                  1166
Special Concern                           1164
Protected Bird - Game with open season    1164
not applicable                            1058
Rare                                      1054
Game with open season                      606
Protected - no open season                  12
Name: count, dtype: int64
```

'NY Listing Status' contains 'not applicable' and 'not listed' and can be replaced with the new category label 'unknown

[27]:
```python
# Define the new category label
new_category = 'Unknown'
# Replace 'not applicable' and 'not listed' with the new category label
df['NY Listing Status'] = df['NY Listing Status'].replace(['not applicable',␣
↪'not listed'], new_category)
df['NY Listing Status'].unique()
```

[27]:
```
array(['Game with open season', 'Special Concern', 'Threatened',
       'Game with no open season', 'Endangered', 'Protected Bird',
       'Unknown', 'Protected Bird - Game with open season',
       'Protected - no open season', 'Rare'], dtype=object)
```

[28]:
```python
df['NY Listing Status'].value_counts()
```

[28]:
```
NY Listing Status
Protected Bird                            7011
Unknown                                   3059
Endangered                                2740
Threatened                                2531
Game with no open season                  1166
```

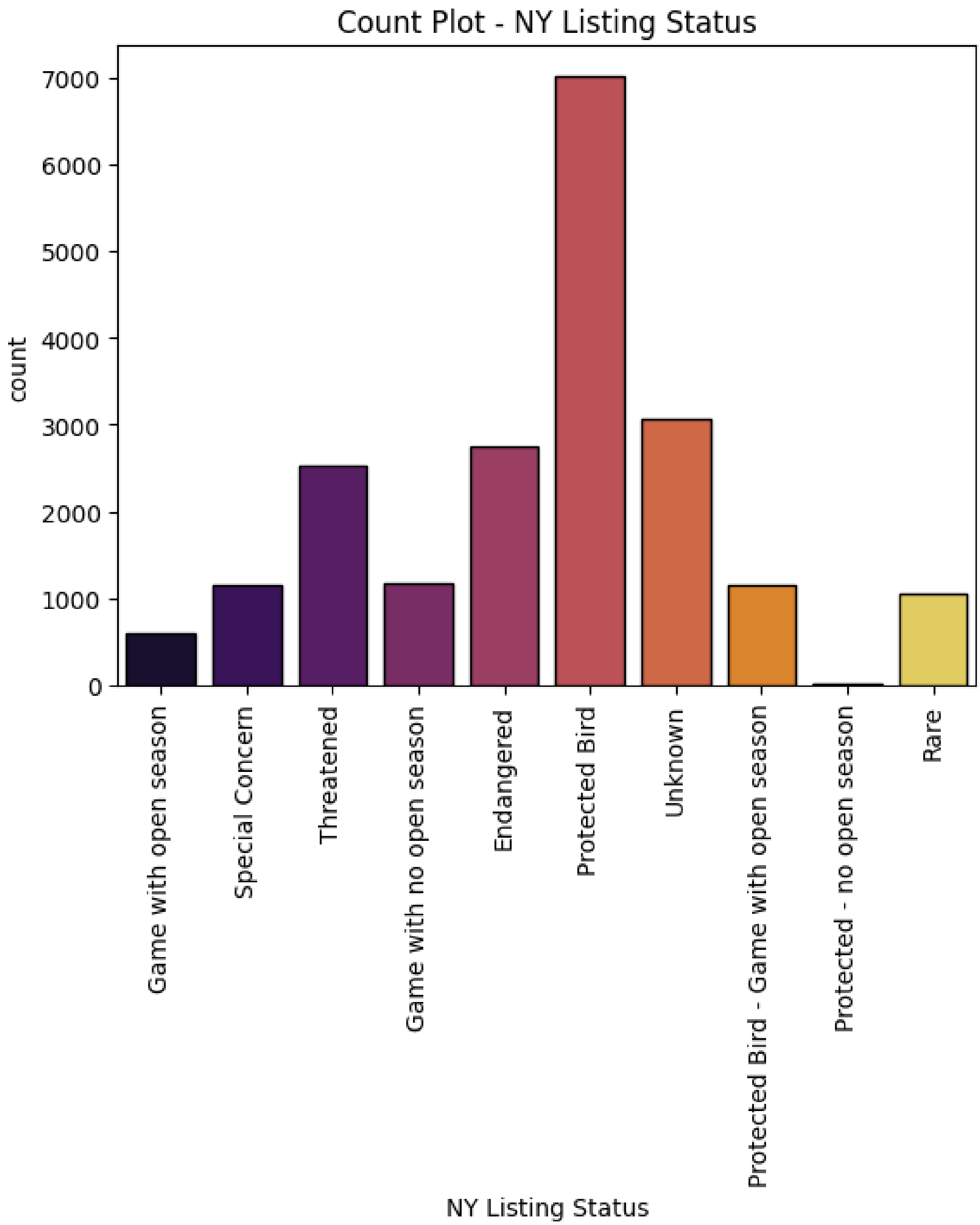
```
Special Concern                             1164
Protected Bird - Game with open season      1164
Rare                                        1054
Game with open season                        606
Protected - no open season                    12
Name: count, dtype: int64
```

[29]:
```
df['NY Listing Status'].unique()
```

[29]:
```
array(['Game with open season', 'Special Concern', 'Threatened',
       'Game with no open season', 'Endangered', 'Protected Bird',
       'Unknown', 'Protected Bird - Game with open season',
       'Protected - no open season', 'Rare'], dtype=object)
```

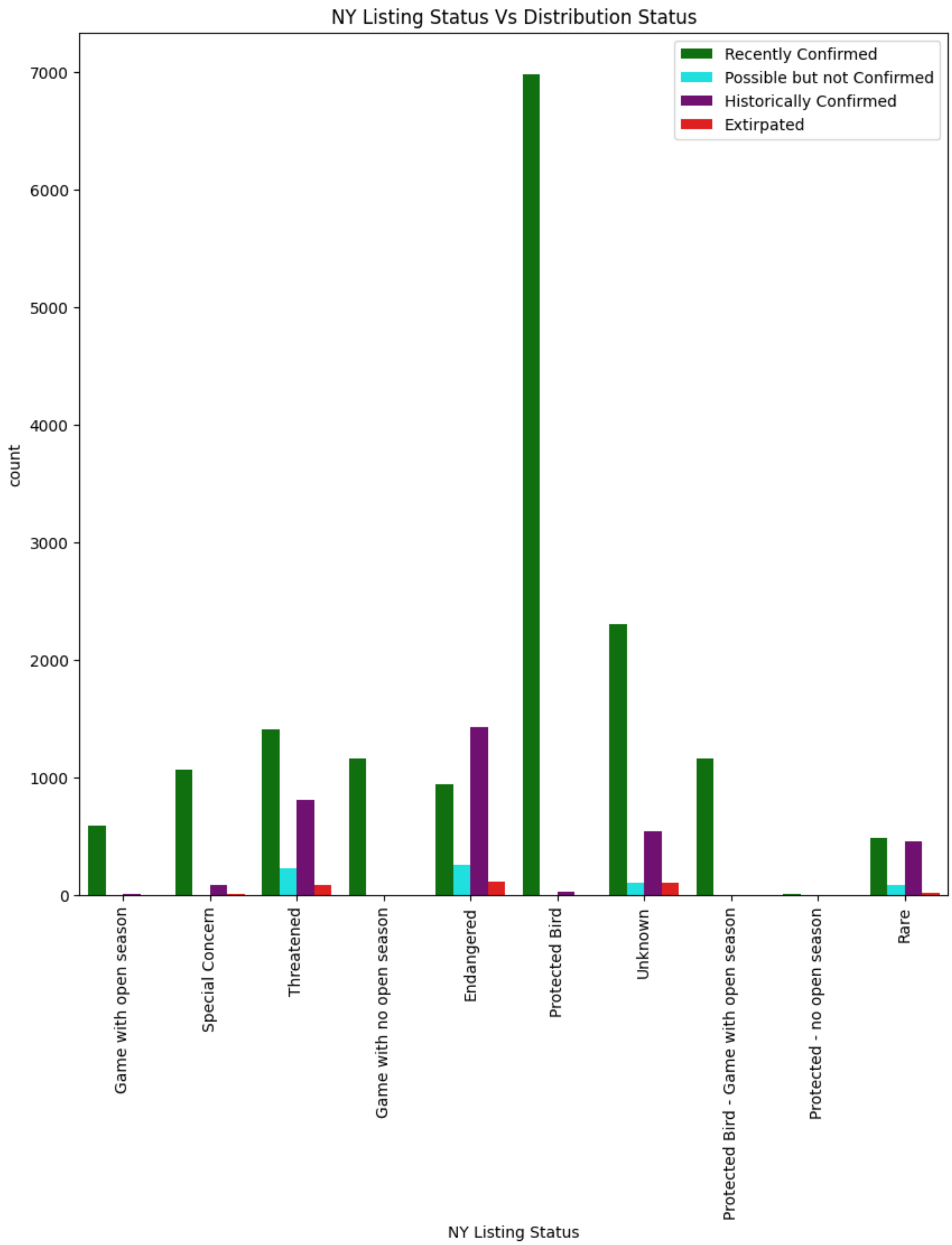
[30]:
```
sns.countplot(x=df['NY Listing Status'],data=df,palette='inferno',edgecolor='k')
plt.title('Count Plot - NY Listing Status')
plt.xticks(rotation=90)
```

[30]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'Game with open season'),
  Text(1, 0, 'Special Concern'),
  Text(2, 0, 'Threatened'),
  Text(3, 0, 'Game with no open season'),
  Text(4, 0, 'Endangered'),
  Text(5, 0, 'Protected Bird'),
  Text(6, 0, 'Unknown'),
  Text(7, 0, 'Protected Bird - Game with open season'),
  Text(8, 0, 'Protected - no open season'),
  Text(9, 0, 'Rare')])
```



[31]:
```python
# 'NY Listing Status' Vs Distribution Status
plt.figure(figsize=(10, 10))
sns.countplot(x='NY Listing Status', hue='Distribution Status', data=df,␣
 ↪palette=["green","cyan","purple","red"])
plt.legend(loc='upper right')  # Show legend
plt.title("NY Listing Status Vs Distribution Status")
```

```
plt.xticks(rotation=90)
plt.show()
```



```
[32]: df['Global Conservation Rank'].unique()
```

```
[32]: array(['G5', 'G4G5', 'GU', 'GNR', 'G2', 'G3G4', 'GNA', 'G4', 'G2G3', 'G3',
       'G5T1T3', 'G5T1', 'G1G2', 'G4T2', 'G5T5', 'G4?', 'G5?', 'G5?T3',
       'G5TNR', 'G5T4T5', 'G5T3T5', 'G4?T4?', 'G4T4', 'G5T4?', 'G5T2',
       'G5T4', 'G5T5?', 'G1', 'GUT1Q', 'G5?T4T5', 'G5T3', 'G3G5', 'G3?',
       'G4G5T4', 'G5?TNR', 'G5?T4?', 'G5?T3T5', 'G4T1T3', 'G4G5T4T5',
       'G3T1', 'GH', 'G5TNRQ', 'G5T3T4', 'G5T3?', 'G4G5Q', 'G2?',
       'G3G4T2', 'G4T3', 'G5T2T4', 'G1Q', 'G5T5?Q', 'G4Q', 'G4G5T3?Q',
       'G4G5T3?', 'G3T3', 'G3Q', 'G2G3T1T2', 'G3T1T3', 'GNRT4?', 'GNRTNR',
       'GXQ', 'G5T1T2', 'G5T4Q'], dtype=object)
```

```
[33]: df['Global Conservation Rank']=df['Global Conservation Rank'].replace(['G4?',␣
↪'G5?','G5?TNR','G5?T4?' ,'G5?T3','G4?T4?', 'G5T4?','G5T5?','G5?T4T5', 'G3?',␣
↪'G5?TNR','G5?T4?', 'G5?T3T5','G5T3?','G2?', 'G5T5?Q','G4G5T3?Q','G4G5T3?
↪','GNRT4?'],'Unknown_Rank')
```

```
[34]: df['Global Conservation Rank'].unique()
```

```
[34]: array(['G5', 'G4G5', 'GU', 'GNR', 'G2', 'G3G4', 'GNA', 'G4', 'G2G3', 'G3',
       'G5T1T3', 'G5T1', 'G1G2', 'G4T2', 'G5T5', 'Unknown_Rank', 'G5TNR',
       'G5T4T5', 'G5T3T5', 'G4T4', 'G5T2', 'G5T4', 'G1', 'GUT1Q', 'G5T3',
       'G3G5', 'G4G5T4', 'G4T1T3', 'G4G5T4T5', 'G3T1', 'GH', 'G5TNRQ',
       'G5T3T4', 'G4G5Q', 'G3G4T2', 'G4T3', 'G5T2T4', 'G1Q', 'G4Q',
       'G3T3', 'G3Q', 'G2G3T1T2', 'G3T1T3', 'GNRTNR', 'GXQ', 'G5T1T2',
       'G5T4Q'], dtype=object)
```

‘G4?’, ‘G5?’, ‘G5?T3’,‘G4?T4?’, ‘G5T4?’,‘G5T5?’,‘G5?T4T5’, ‘G3?’, ‘G5?TNR’‘G5?T4?’, ‘G5?T3T5’,‘G5T3?,’G2?’, ‘G5T5?Q’,‘G4G5T3?Q’,‘G4G5T3?’,‘GNRT4?’ are Inexact Numeric Rank so can be considered as a separate class Unknown Rank.

```
[35]: df['Global Conservation Rank'].value_counts()
```

```
[35]: Global Conservation Rank
G5              15062
G4               1556
G4G5              767
G3G4              692
G5T5              615
G3                437
Unknown_Rank      353
G2G3              137
G5T4T5            137
GNR               129
GNA               103
G2                 77
G5TNR              72
```

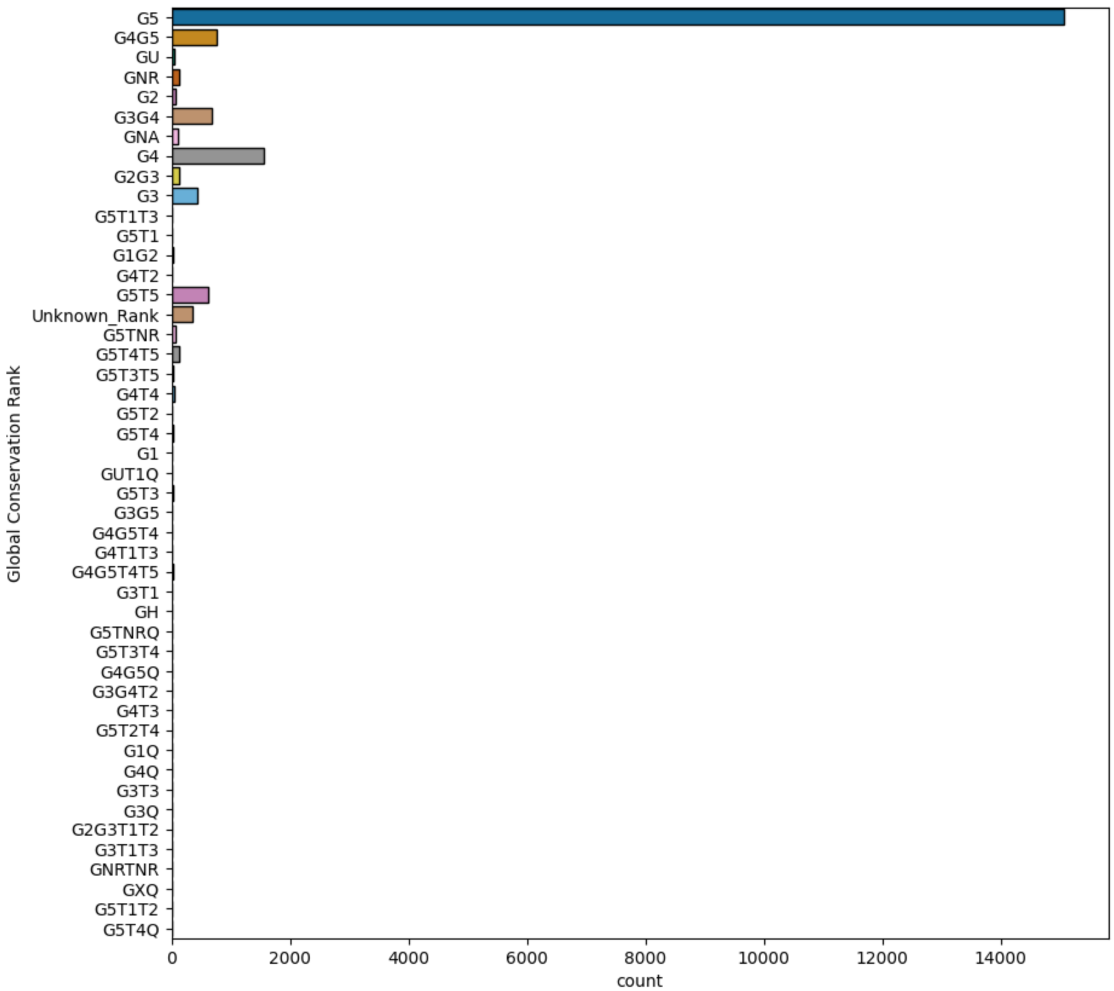
```
G4T4           60
GU             44
G1G2           36
G5T4           35
G5T3           35
G5T3T5         28
G4G5T4T5       20
G4G5T4         17
G1             14
G5T2           11
G5T1T3          7
G5T1            6
GH              6
G4T2            6
G5TNRQ          5
G3G5            4
G3G4T2          4
G3Q             3
G3T3            3
G3T1            3
G4G5Q           3
G4T3            3
G4Q             3
G5T1T2          2
G5T2T4          2
GNRTNR          2
G3T1T3          1
GXQ             1
GUT1Q           1
G2G3T1T2        1
G1Q             1
G5T3T4          1
G4T1T3          1
G5T4Q           1
Name: count, dtype: int64
```

[36]:
```
df['Global Conservation Rank'].unique()
```

[36]:
```
array(['G5', 'G4G5', 'GU', 'GNR', 'G2', 'G3G4', 'GNA', 'G4', 'G2G3', 'G3',
       'G5T1T3', 'G5T1', 'G1G2', 'G4T2', 'G5T5', 'Unknown_Rank', 'G5TNR',
       'G5T4T5', 'G5T3T5', 'G4T4', 'G5T2', 'G5T4', 'G1', 'GUT1Q', 'G5T3',
       'G3G5', 'G4G5T4', 'G4T1T3', 'G4G5T4T5', 'G3T1', 'GH', 'G5TNRQ',
       'G5T3T4', 'G4G5Q', 'G3G4T2', 'G4T3', 'G5T2T4', 'G1Q', 'G4Q',
       'G3T3', 'G3Q', 'G2G3T1T2', 'G3T1T3', 'GNRTNR', 'GXQ', 'G5T1T2',
       'G5T4Q'], dtype=object)
```

[37]:
```
plt.figure(figsize=(10,10))
sns.countplot(df['Global Conservation Rank'],palette='colorblind',edgecolor='k')
```

[37]: <Axes: xlabel='count', ylabel='Global Conservation Rank'>



[38]:
```
df['State Conservation Rank'].unique()
```

[38]:
```
array(['S5', 'S4', 'S2S3', 'S3B,S1N', 'SNA', 'S3', 'S1S2', 'S3S4', 'SNRN',
       'SH', 'S1', 'S5B', 'S4B', 'S2S3B,SNRN', 'S3B,SNRN', 'S3B',
       'S3S4B,S3N', 'S3B,S3N', 'S2S3B,S2N', 'S3S4B', 'S1B', 'S2?B', 'S2B',
       'S3?B', 'SU', 'S1S3', 'S2S4', 'S2?', 'S2S3B', 'S2', 'S4S5', 'SX',
       'S1N', 'S1?', 'S3?', 'S3S4N', 'SNAB,S3N', 'SNAB', 'SNR', 'SHB,S1N',
       'SNRB', 'S2S3M', 'S1B,S3?N'], dtype=object)
```

'S3?B','S2?','S1?', 'S3?','S1?', 'S3?','S1B,S3?N' are considered as inexact ranks can be treated as separate label.

[39]:
```
df['State Conservation Rank']=df['State Conservation Rank'].replace(['S3?B','S2?
↪','S1?', '2?','S2?B', 'S3?','S1?', 'S3?','S1B,S3?N'],'Unknown')
```

[40]:
```
df['State Conservation Rank'].unique()
```

[40]:
```
array(['S5', 'S4', 'S2S3', 'S3B,S1N', 'SNA', 'S3', 'S1S2', 'S3S4', 'SNRN',
       'SH', 'S1', 'S5B', 'S4B', 'S2S3B,SNRN', 'S3B,SNRN', 'S3B',
       'S3S4B,S3N', 'S3B,S3N', 'S2S3B,S2N', 'S3S4B', 'S1B', 'Unknown',
       'S2B', 'SU', 'S1S3', 'S2S4', 'S2S3B', 'S2', 'S4S5', 'SX', 'S1N',
       'S3S4N', 'SNAB,S3N', 'SNAB', 'SNR', 'SHB,S1N', 'SNRB', 'S2S3M'],
      dtype=object)
```

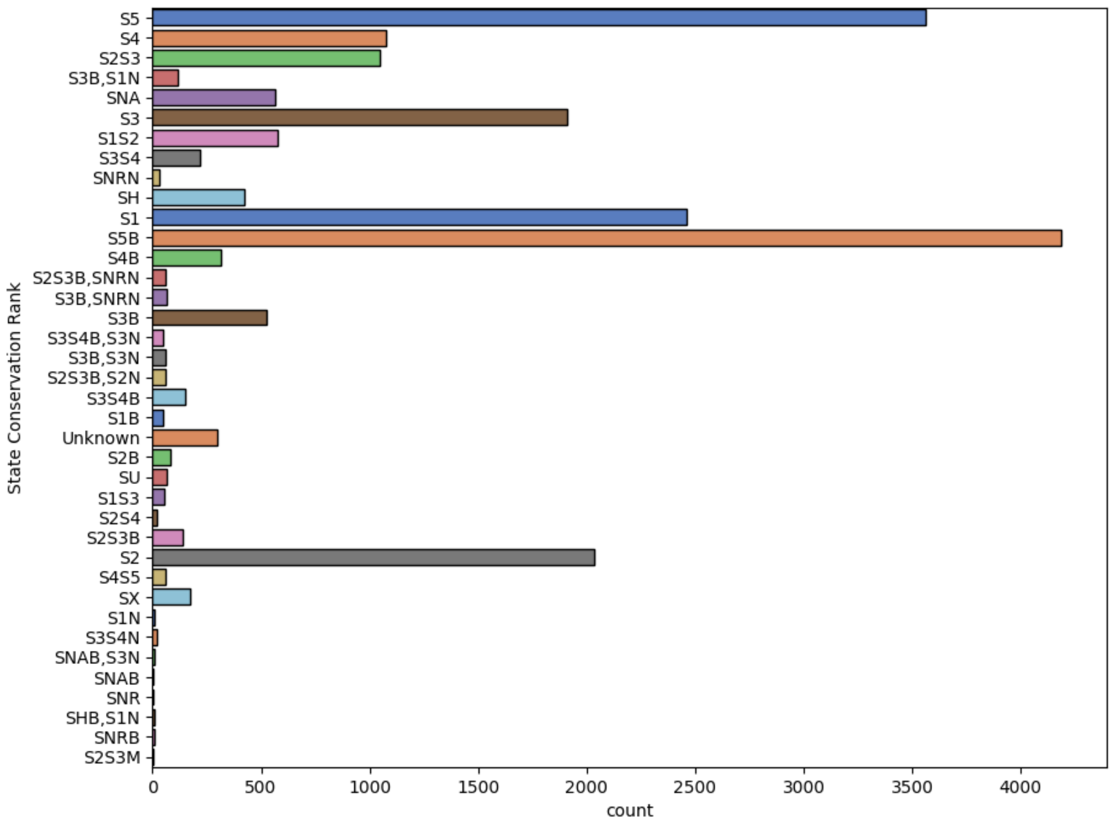
[41]:
```
df['State Conservation Rank'].value_counts()
```

[41]:
```
State Conservation Rank
S5B             4189
S5              3563
S1              2459
S2              2037
S3              1911
S4              1075
S2S3            1047
S1S2             575
SNA              568
S3B              524
SH               426
S4B              316
Unknown          300
S3S4             218
SX               176
S3S4B            149
S2S3B            137
S3B,S1N          115
S2B               82
SU                68
S3B,SNRN          67
S4S5              62
S2S3B,SNRN        61
S2S3B,S2N         58
S3B,S3N           58
S1S3              53
S3S4B,S3N         47
S1B               47
SNRN              33
S3S4N             23
S2S4              19
SNRB              11
```

```
S1N              8
SNAB,S3N         8
SHB,S1N          7
SNAB             4
S2S3M            4
SNR              2
Name: count, dtype: int64
```

[42]:
```
plt.figure(figsize=(10,8))
sns.countplot(df['State Conservation Rank'],palette='muted',edgecolor='k')
```

[42]: <Axes: xlabel='count', ylabel='State Conservation Rank'>



Find unique values for column 'Year Last Documented'

[43]:
```
df['Year Last Documented'].unique()
```

[43]:
```
array(['1990-1999', '2006', '2019', 'not available', '2000-2005', '1986',
       '2018', '1991', '1987', '2009', '2012', '1983', '2002', '2021',
       '2016', '1984', '1992', '2013', '2017', '2015', '1963', '1970',
       '2014', '1990', '2008', '1926', '1874', '2010', '1890', '1904',
```

```
        '1960', '1998', '1988', '2000', '2003', '2001', '1999', '2020',
        '1997', '1996', '1962', '1959', '1980', '1932', '1937', '1933',
        '1907', '1910', '1928', '1936', '1951', '1995', '1939', '1923',
        '1920', '2004', '1948', '1955', '1974', '1835', '1865', '1942',
        '1919', '1957', '1950', '2005', '2007', '1981', '1989', '1994',
        '1913', '1897', '1946', '1906', '1953', '1918', '1895', '1893',
        '1894', '1954', '1899', '1947', '1898', '1843', '1840', '1892',
        '1896', '1940', '1915', '1879', '1880', '1900', '1901', '1966',
        '1891', '1882', '1938', '2011', '1982', '1889', '1885', '1929',
        '1930', '1975', '1977', '1927', '1943', '1958', '1873', '1931',
        '1924', '1921', '1934', '1985', '1922', '1941', '1917', '1916',
        '1993', '1877', '1887', '1979', '1878', '1845', '1956', '1949',
        '1935', '1869', '1965', '1902', '1976', '1969', '1851', '1875',
        '1971', '1867', '1868', '1861', '1863', '1888', '1834', '1908',
        '1862', '1871', '1967', '1856', '1905', '1914', '1912', '1964',
        '1864', '1952', '1909', '1846', '1870', '1842', '1841', '1883',
        '1881', '1800', '1911', '1903', '1831', '1830', '1945', '1886',
        '1925', '1961', '1811', '1817', '1872', '1838', '1823', '1944',
        '1837', '1884', '1854', '1978', '1876', '1853', '1836', '1857',
        '1968', '1832', '1815', '1860', '1973'], dtype=object)
```

- Some samples exhibit a range of years in their data.
- These ranges can be substituted with the last year of documentation, mirroring the treatment of other samples.
- Entries marked as 'Not available' can be considered as NaN in the following steps.

[44]:
```
df['Year Last Documented']=df['Year Last Documented'].str.
 ↪replace('2000-2005','2005')
df['Year Last Documented']=df['Year Last Documented'].str.
 ↪replace('1990-1999','1999')
```

[45]:
```
df['Year Last Documented'].unique()
```

[45]:
```
array(['1999', '2006', '2019', 'not available', '2005', '1986', '2018',
        '1991', '1987', '2009', '2012', '1983', '2002', '2021', '2016',
        '1984', '1992', '2013', '2017', '2015', '1963', '1970', '2014',
        '1990', '2008', '1926', '1874', '2010', '1890', '1904', '1960',
        '1998', '1988', '2000', '2003', '2001', '2020', '1997', '1996',
        '1962', '1959', '1980', '1932', '1937', '1933', '1907', '1910',
        '1928', '1936', '1951', '1995', '1939', '1923', '1920', '2004',
        '1948', '1955', '1974', '1835', '1865', '1942', '1919', '1957',
        '1950', '2007', '1981', '1989', '1994', '1913', '1897', '1946',
        '1906', '1953', '1918', '1895', '1893', '1894', '1954', '1899',
        '1947', '1898', '1843', '1840', '1892', '1896', '1940', '1915',
        '1879', '1880', '1900', '1901', '1966', '1891', '1882', '1938',
        '2011', '1982', '1889', '1885', '1929', '1930', '1975', '1977',
        '1927', '1943', '1958', '1873', '1931', '1924', '1921', '1934',
```

```
'1985', '1922', '1941', '1917', '1916', '1993', '1877', '1887',
'1979', '1878', '1845', '1956', '1949', '1935', '1869', '1965',
'1902', '1976', '1969', '1851', '1875', '1971', '1867', '1868',
'1861', '1863', '1888', '1834', '1908', '1862', '1871', '1967',
'1856', '1905', '1914', '1912', '1964', '1864', '1952', '1909',
'1846', '1870', '1842', '1841', '1883', '1881', '1800', '1911',
'1903', '1831', '1830', '1945', '1886', '1925', '1961', '1811',
'1817', '1872', '1838', '1823', '1944', '1837', '1884', '1854',
'1978', '1876', '1853', '1836', '1857', '1968', '1832', '1815',
'1860', '1973'], dtype=object)
```

Entries marked as 'Not available' can be replaced with mode value.

[46]:
```
mode_value = df['Year Last Documented'].mode()[0]
df['Year Last Documented'] = df['Year Last Documented'].replace('not␣
↪available', mode_value)
```

[47]:
```
df.isna().sum()
```

[47]:
```
County                    0
Category                  0
Taxonomic Group           0
Taxonomic Subgroup        0
Scientific Name           0
Common Name               0
Year Last Documented      0
NY Listing Status         0
Federal Listing Status    0
State Conservation Rank   0
Global Conservation Rank  0
Distribution Status       0
dtype: int64
```

Convert the data type of the column 'Year Last Documented' from object to integer using the astype function.

[48]:
```
df['Year Last Documented']=df['Year Last Documented'].astype('int')
```

[49]:
```
df['Year Last Documented'].unique()
```

[49]:
```
array([1999, 2006, 2019, 2005, 1986, 2018, 1991, 1987, 2009, 2012, 1983,
       2002, 2021, 2016, 1984, 1992, 2013, 2017, 2015, 1963, 1970, 2014,
       1990, 2008, 1926, 1874, 2010, 1890, 1904, 1960, 1998, 1988, 2000,
       2003, 2001, 2020, 1997, 1996, 1962, 1959, 1980, 1932, 1937, 1933,
       1907, 1910, 1928, 1936, 1951, 1995, 1939, 1923, 1920, 2004, 1948,
       1955, 1974, 1835, 1865, 1942, 1919, 1957, 1950, 2007, 1981, 1989,
       1994, 1913, 1897, 1946, 1906, 1953, 1918, 1895, 1893, 1894, 1954,
       1899, 1947, 1898, 1843, 1840, 1892, 1896, 1940, 1915, 1879, 1880,
```

```
       1900, 1901, 1966, 1891, 1882, 1938, 2011, 1982, 1889, 1885, 1929,
       1930, 1975, 1977, 1927, 1943, 1958, 1873, 1931, 1924, 1921, 1934,
       1985, 1922, 1941, 1917, 1916, 1993, 1877, 1887, 1979, 1878, 1845,
       1956, 1949, 1935, 1869, 1965, 1902, 1976, 1969, 1851, 1875, 1971,
       1867, 1868, 1861, 1863, 1888, 1834, 1908, 1862, 1871, 1967, 1856,
       1905, 1914, 1912, 1964, 1864, 1952, 1909, 1846, 1870, 1842, 1841,
       1883, 1881, 1800, 1911, 1903, 1831, 1830, 1945, 1886, 1925, 1961,
       1811, 1817, 1872, 1838, 1823, 1944, 1837, 1884, 1854, 1978, 1876,
       1853, 1836, 1857, 1968, 1832, 1815, 1860, 1973])
```

10 Drop function

Dropping less impact columns.

[50]:
```python
df.drop(['Common Name','Scientific Name','Federal Listing Status','Taxonomic␣
 ↪Group'],axis=1,inplace=True)
```

[51]:
```python
df.dtypes
```

[51]:
```
County                      object
Category                    object
Taxonomic Subgroup          object
Year Last Documented         int64
NY Listing Status           object
State Conservation Rank     object
Global Conservation Rank    object
Distribution Status         object
dtype: object
```

[52]:
```python
df.columns
```

[52]:
```
Index(['County', 'Category', 'Taxonomic Subgroup', 'Year Last Documented',
       'NY Listing Status', 'State Conservation Rank',
       'Global Conservation Rank', 'Distribution Status'],
      dtype='object')
```

11 Encoding the columns 'Common Name', 'NY Listing Status', 'State Conservation Rank' and 'Global Conservation Rank'

Since 'State Conservation Rank' and 'Global Conservation Rank' are ordinal data so encoding can be done by label encoder but its order is different in the dataset. So i created separate list with actual order.

[53]:
```python
from sklearn.preprocessing import LabelEncoder
encode = LabelEncoder()
# Define the custom mapping based on State conservation rank order
```

```
state_rank_order = {
    'S1': 0, 'S1B': 1, 'S1N': 2, 'S1S2': 3, 'S2': 4, 'S2B': 5, 'S2S3': 6,
    'S2S3B': 7, 'S2S3B,S2N': 8, 'S2S3B,SNRN': 9, 'S2S3M': 10, 'S2S4': 11, 'S3': 12,
    'S3B': 13, 'S3B,S1N': 14, 'S3B,S3N': 15, 'S3B,SNRN': 16, 'S3S4': 17, 'S3S4B': 18,
    'S3S4B,S3N': 19, 'S3S4N': 20, 'S4': 21, 'S4B': 22, 'S4S5': 23, 'S5': 24, 'S5B': 25,
    'SH': 26, 'SHB,S1N': 27, 'SNAB': 28, 'SNAB,S3N': 29, 'SNAB,S3N': 30, 'SNAB,S3N': 31,
    'SNAB,S3N': 32, 'SNR': 33, 'SNRB': 34, 'SU': 35, 'SX': 36, 'Unknown': 37
}

# Create a LabelEncoder object and fit_transform your column
encode = LabelEncoder()
df['State Conservation Rank'] = encode.fit_transform(df['State Conservation Rank'].map(state_rank_order))
```

[54]:

```
# Define the custom mapping based on corrected order
global_rank_order = {
    'G1': 0,
    'G1G2': 1,
    'G1Q': 2,
    'G2': 3,
    'G2G3': 4,
    'G2G3T1T2': 5,
    'G3': 6,
    'G3G4': 7,
    'G3G4T2': 8,
    'G3Q': 9,
    'G3T1': 10,
    'G3T1T3': 11,
    'G3T3': 12,
    'G3G5': 13,
    'G3T1T3': 14,
    'G3G4T2': 15,
    'G3Q': 16,
    'G4': 17,
    'G4Q': 18,
    'G4T1T3': 19,
    'G4T2': 20,
    'G4T3': 21,
    'G4T4': 22,
    'G4G5': 23,
    'G4G5Q': 24,
    'G4G5T4': 25,
    'G4G5T4T5': 26,
```

```
    'G5': 27,
    'G5T1': 28,
    'G5T1T2': 29,
    'G5T1T3': 30,
    'G5T2': 31,
    'G5T2T4': 32,
    'G5T3': 33,
    'G5T3T4': 34,
    'G5T3T5': 35,
    'G5T4': 36,
    'G5T4Q': 37,
    'G5T4T5': 38,
    'G5T5': 39,
    'G5TNR': 40,
    'G5TNRQ': 41,
    'GH': 42,
    'GNR': 43,
    'GNA': 44,
    'GNRTNR': 45,
    'GU': 46,
    'GUT1Q': 47,
    'GXQ': 48,
    'Unknown_Rank': 49,
   }

# Create a LabelEncoder object and fit_transform your column
encode = LabelEncoder()
df['Global Conservation Rank'] = encode.fit_transform(df['Global Conservation␣
↪Rank'].map(global_rank_order))
```

Label encoding is suitable when dealing with a large number of unique values (like 1578 unique common names) because it encodes each unique value with a unique integer label.

[55]:
```
#encode.fit(df['Common Name'])
#df['Common Name']=encode.transform(df['Common Name'])
#df['Common Name']
```

[56]:
```
encode.fit(df['County'])
df['County']=encode.transform(df['County'])
df['County']
```

[56]:
```
0          0
1          0
2          0
3          0
4          0
          ..
```

```
20502    65
20503    65
20504    65
20505    65
20506    65
Name: County, Length: 20507, dtype: int64
```

[57]:
```
encode.fit(df['Category'])
df['Category']=encode.transform(df['Category'])
df['Category']
```

[57]:
```
0        0
1        0
2        0
3        0
4        0
        ..
20502    2
20503    2
20504    2
20505    2
20506    2
Name: Category, Length: 20507, dtype: int64
```

12 Label Encoding for Target Column

Apply label encoding to the target column 'Distribution Status'.

[58]:
```
encode.fit(df['Distribution Status'])
df['Distribution Status']=encode.transform(df['Distribution Status'])
df['Distribution Status']
# Get the actual class names in the correct order
class_names = encode.classes_
print("Class Names:", class_names)
```

```
Class Names: ['Extirpated' 'Historically Confirmed' 'Possible but not Confirmed'
 'Recently Confirmed']
```

13 Get dummies Encoding

Transform the 'County', 'Category', and 'Taxonomic Group' columns into numerical format using with get_dummies encoding.

[59]:
```
df1=pd.get_dummies(df[['NY Listing Status','Taxonomic␣
↪Subgroup']],drop_first=False,dtype=int)
```

[60]:
```
df1.columns
```

[60]:
```
Index(['NY Listing Status_Endangered',
       'NY Listing Status_Game with no open season',
       'NY Listing Status_Game with open season',
       'NY Listing Status_Protected - no open season',
       'NY Listing Status_Protected Bird',
       'NY Listing Status_Protected Bird - Game with open season',
       'NY Listing Status_Rare', 'NY Listing Status_Special Concern',
       'NY Listing Status_Threatened', 'NY Listing Status_Unknown',
       …
       'Taxonomic Subgroup_Sturgeons and Paddlefish',
       'Taxonomic Subgroup_Subtidal Wetlands', 'Taxonomic Subgroup_Swallows',
       'Taxonomic Subgroup_Thrushes and Bluebirds',
       'Taxonomic Subgroup_Vireos', 'Taxonomic Subgroup_Waxwings',
       'Taxonomic Subgroup_Whales and Dolphins',
       'Taxonomic Subgroup_Wood-Warblers', 'Taxonomic Subgroup_Woodpeckers',
       'Taxonomic Subgroup_Wrens'],
      dtype='object', length=115)
```

14 Combine dataframes

Combine the encoded columns and the original dataframe into a single dataframe.

[61]:
```
df2=pd.concat([df,df1],axis=1)
df2.shape
```

[61]:
```
(20507, 123)
```

[62]:
```
df2.columns
```

[62]:
```
Index(['County', 'Category', 'Taxonomic Subgroup', 'Year Last Documented',
       'NY Listing Status', 'State Conservation Rank',
       'Global Conservation Rank', 'Distribution Status',
       'NY Listing Status_Endangered',
       'NY Listing Status_Game with no open season',
       …
       'Taxonomic Subgroup_Sturgeons and Paddlefish',
       'Taxonomic Subgroup_Subtidal Wetlands', 'Taxonomic Subgroup_Swallows',
       'Taxonomic Subgroup_Thrushes and Bluebirds',
       'Taxonomic Subgroup_Vireos', 'Taxonomic Subgroup_Waxwings',
       'Taxonomic Subgroup_Whales and Dolphins',
       'Taxonomic Subgroup_Wood-Warblers', 'Taxonomic Subgroup_Woodpeckers',
       'Taxonomic Subgroup_Wrens'],
      dtype='object', length=123)
```

Remove the ‘County’, ‘Category’, and ‘Taxonomic Group’ columns from the dataframe since they have already been encoded.

[63]:
```
df2.drop(['NY Listing Status','Taxonomic Subgroup'],axis=1,inplace=True)
```

[64]:
```
df2.dtypes
```

[64]:
```
County                                    int64
Category                                  int64
Year Last Documented                      int64
State Conservation Rank                   int64
Global Conservation Rank                  int64
                                          …
Taxonomic Subgroup_Waxwings               int64
Taxonomic Subgroup_Whales and Dolphins    int64
Taxonomic Subgroup_Wood-Warblers          int64
Taxonomic Subgroup_Woodpeckers            int64
Taxonomic Subgroup_Wrens                  int64
Length: 121, dtype: object
```

[65]:
```
df2.shape
```

[65]:
```
(20507, 121)
```

15 x and y separation

Separate the features (x) and the target variable (y) from the dataframe.

[66]:
```
x=df2.drop(['Distribution Status'],axis=1)
x
```

[66]:
```
       County  Category  Year Last Documented  State Conservation Rank  \
0           0         0                  1999                       24
1           0         0                  1999                       21
2           0         0                  1999                       24
3           0         0                  1999                       24
4           0         0                  1999                       24
…        …       …                   …                        …
20502      65         2                  2005                       26
20503      65         2                  2005                       21
20504      65         2                  2005                        3
20505      65         2                  2005                        0
20506      65         2                  2005                        6

       Global Conservation Rank  NY Listing Status_Endangered  \
0                            24                             0
1                            24                             0
2                            24                             0
```

```
3                         24                          0
4                         24                          0
…                       …                         …
20502                     20                          1
20503                     24                          0
20504                     24                          1
20505                     24                          1
20506                     20                          0

       NY Listing Status_Game with no open season  \
0                                             0
1                                             0
2                                             0
3                                             0
4                                             0
…                                           …
20502                                         0
20503                                         0
20504                                         0
20505                                         0
20506                                         0

       NY Listing Status_Game with open season  \
0                                          1
1                                          1
2                                          1
3                                          1
4                                          1
…                                        …
20502                                      0
20503                                      0
20504                                      0
20505                                      0
20506                                      0

       NY Listing Status_Protected - no open season  \
0                                               0
1                                               0
2                                               0
3                                               0
4                                               0
…                                             …
20502                                           0
20503                                           0
20504                                           0
20505                                           0
20506                                           0
```

```
       NY Listing Status_Protected Bird  …  \
0                                      0  …   
1                                      0  …   
2                                      0  …   
3                                      0  …   
4                                      0  …   
…                                    …  …   
20502                                  0  …   
20503                                  0  …   
20504                                  0  …   
20505                                  0  …   
20506                                  0  …   

       Taxonomic Subgroup_Sturgeons and Paddlefish  \
0                                                0   
1                                                0   
2                                                0   
3                                                0   
4                                                0   
…                                              …   
20502                                            0   
20503                                            0   
20504                                            0   
20505                                            0   
20506                                            0   

       Taxonomic Subgroup_Subtidal Wetlands  Taxonomic Subgroup_Swallows  \
0                                         0                            0   
1                                         0                            0   
2                                         0                            0   
3                                         0                            0   
4                                         0                            0   
…                                       …                          …   
20502                                     0                            0   
20503                                     0                            0   
20504                                     0                            0   
20505                                     0                            0   
20506                                     0                            0   

       Taxonomic Subgroup_Thrushes and Bluebirds  Taxonomic Subgroup_Vireos  \
0                                              0                          0   
1                                              0                          0   
2                                              0                          0   
3                                              0                          0   
4                                              0                          0   
…                                            …                        …   
```

```
20502                                          0                          0
20503                                          0                          0
20504                                          0                          0
20505                                          0                          0
20506                                          0                          0

       Taxonomic Subgroup_Waxwings  Taxonomic Subgroup_Whales and Dolphins  \
0                                0                                       0
1                                0                                       0
2                                0                                       0
3                                0                                       0
4                                0                                       0
…                            …                                     …
20502                            0                                       0
20503                            0                                       0
20504                            0                                       0
20505                            0                                       0
20506                            0                                       0

       Taxonomic Subgroup_Wood-Warblers  Taxonomic Subgroup_Woodpeckers  \
0                                     0                               0
1                                     0                               0
2                                     0                               0
3                                     0                               0
4                                     0                               0
…                                 …                             …
20502                                 0                               0
20503                                 0                               0
20504                                 0                               0
20505                                 0                               0
20506                                 0                               0

       Taxonomic Subgroup_Wrens
0                             0
1                             0
2                             0
3                             0
4                             0
…                         …
20502                         0
20503                         0
20504                         0
20505                         0
20506                         0

[20507 rows x 120 columns]
```

[67]:
```
y=df2['Distribution Status']
y
```

[67]:
```
0        3
1        3
2        3
3        3
4        3
        ..
20502    1
20503    1
20504    1
20505    1
20506    3
Name: Distribution Status, Length: 20507, dtype: int64
```

[68]:
```
df2['Distribution Status']=y
```

Check again the datatypes after encoding the target column.

[69]:
```
df2.dtypes
```

[69]:
```
County                                    int64
Category                                  int64
Year Last Documented                      int64
State Conservation Rank                   int64
Global Conservation Rank                  int64
                                          …
Taxonomic Subgroup_Waxwings               int64
Taxonomic Subgroup_Whales and Dolphins    int64
Taxonomic Subgroup_Wood-Warblers          int64
Taxonomic Subgroup_Woodpeckers            int64
Taxonomic Subgroup_Wrens                  int64
Length: 121, dtype: object
```

16 Feature Selection using Pearson Correlation

Regarding the correlation matrix calculation, the df2.corr() function computes pairwise correlation of columns, generating a correlation matrix. This matrix shows how each column in the DataFrame is correlated with every other column, which is helpful for identifying relationships and dependencies between variables.

[70]:
```
corre=df2.corr()
corre
```

[70]:
```
                                          County  Category  \
County                                  1.000000  0.046736
```

```
Category                                   0.046736  1.000000
Year Last Documented                      -0.001353 -0.294774
State Conservation Rank                   -0.023515 -0.576090
Global Conservation Rank                   0.013963  0.096424
…                                               …         …
Taxonomic Subgroup_Waxwings               -0.001483 -0.038199
Taxonomic Subgroup_Whales and Dolphins -0.031087 -0.012818
Taxonomic Subgroup_Wood-Warblers          -0.016140 -0.187910
Taxonomic Subgroup_Woodpeckers            -0.006499 -0.100912
Taxonomic Subgroup_Wrens                  -0.002774 -0.080123

                                          Year Last Documented  \
County                                               -0.001353
Category                                             -0.294774
Year Last Documented                                  1.000000
State Conservation Rank                               0.133962
Global Conservation Rank                             -0.043267
…                                                          …
Taxonomic Subgroup_Waxwings                           0.013123
Taxonomic Subgroup_Whales and Dolphins                0.008727
Taxonomic Subgroup_Wood-Warblers                      0.063691
Taxonomic Subgroup_Woodpeckers                        0.036380
Taxonomic Subgroup_Wrens                              0.027961

                                          State Conservation Rank  \
County                                                  -0.023515
Category                                                -0.576090
Year Last Documented                                     0.133962
State Conservation Rank                                  1.000000
Global Conservation Rank                                 0.143852
…                                                             …
Taxonomic Subgroup_Waxwings                              0.045671
Taxonomic Subgroup_Whales and Dolphins                   0.018343
Taxonomic Subgroup_Wood-Warblers                         0.209278
Taxonomic Subgroup_Woodpeckers                           0.126456
Taxonomic Subgroup_Wrens                                 0.067755

                                          Global Conservation Rank  \
County                                                    0.013963
Category                                                  0.096424
Year Last Documented                                     -0.043267
State Conservation Rank                                   0.143852
Global Conservation Rank                                  1.000000
…                                                              …
Taxonomic Subgroup_Waxwings                               0.008204
Taxonomic Subgroup_Whales and Dolphins                   -0.036442
Taxonomic Subgroup_Wood-Warblers                          0.046038
```

```
Taxonomic Subgroup_Woodpeckers                             0.021672
Taxonomic Subgroup_Wrens                                   0.017207

                                        Distribution Status  \
County                                            -0.018673
Category                                          -0.564018
Year Last Documented                               0.454529
State Conservation Rank                            0.353223
Global Conservation Rank                          -0.043861
…                                                     …
Taxonomic Subgroup_Waxwings                        0.027644
Taxonomic Subgroup_Whales and Dolphins             0.009276
Taxonomic Subgroup_Wood-Warblers                   0.125129
Taxonomic Subgroup_Woodpeckers                     0.058811
Taxonomic Subgroup_Wrens                           0.049622

                                        NY Listing Status_Endangered  \
County                                                     0.023385
Category                                                   0.506073
Year Last Documented                                      -0.291429
State Conservation Rank                                   -0.439645
Global Conservation Rank                                   0.048379
…                                                              …
Taxonomic Subgroup_Waxwings                               -0.021626
Taxonomic Subgroup_Whales and Dolphins                     0.039296
Taxonomic Subgroup_Wood-Warblers                          -0.106382
Taxonomic Subgroup_Woodpeckers                            -0.057129
Taxonomic Subgroup_Wrens                                  -0.045360

                                        NY Listing Status_Game with no open
season  \
County
-0.000703
Category
-0.170318
Year Last Documented
-0.006080
State Conservation Rank
0.154441
Global Conservation Rank
0.049782
…
…
Taxonomic Subgroup_Waxwings
-0.013521
Taxonomic Subgroup_Whales and Dolphins
-0.004537
```

```
Taxonomic Subgroup_Wood-Warblers
-0.066514
Taxonomic Subgroup_Woodpeckers
-0.035719
Taxonomic Subgroup_Wrens
-0.028361

                                        NY Listing Status_Game with open season
\
County                                                                -0.004836
Category                                                              -0.121046
Year Last Documented                                                  -0.003350
State Conservation Rank                                                0.101679
Global Conservation Rank                                               0.022675
…                                                                         …
Taxonomic Subgroup_Waxwings                                           -0.009610
Taxonomic Subgroup_Whales and Dolphins                                -0.003225
Taxonomic Subgroup_Wood-Warblers                                      -0.047271
Taxonomic Subgroup_Woodpeckers                                        -0.025386
Taxonomic Subgroup_Wrens                                              -0.020156

                                        NY Listing Status_Protected - no open
season  \
County
-0.002983
Category
-0.016785
Year Last Documented
0.012907
State Conservation Rank
-0.037565
Global Conservation Rank
-0.059180
…
…
Taxonomic Subgroup_Waxwings
-0.001333
Taxonomic Subgroup_Whales and Dolphins
-0.000447
Taxonomic Subgroup_Wood-Warblers
-0.006555
Taxonomic Subgroup_Woodpeckers
-0.003520
Taxonomic Subgroup_Wrens
-0.002795

                                        …  \
```

```
County                                  …
Category                                …
Year Last Documented                    …
State Conservation Rank                 …
Global Conservation Rank                …
…                                     …
Taxonomic Subgroup_Waxwings             …
Taxonomic Subgroup_Whales and Dolphins  …
Taxonomic Subgroup_Wood-Warblers        …
Taxonomic Subgroup_Woodpeckers          …
Taxonomic Subgroup_Wrens                …

                                        Taxonomic Subgroup_Sturgeons and
Paddlefish  \
County
-0.004680
Category
-0.031797
Year Last Documented
0.021872
State Conservation Rank
-0.059586
Global Conservation Rank
-0.109191
…
…
Taxonomic Subgroup_Waxwings
-0.002524
Taxonomic Subgroup_Whales and Dolphins
-0.000847
Taxonomic Subgroup_Wood-Warblers
-0.012418
Taxonomic Subgroup_Woodpeckers
-0.006669
Taxonomic Subgroup_Wrens
-0.005295

                                        Taxonomic Subgroup_Subtidal Wetlands  \
County                                                              -0.000681
Category                                                             0.011260
Year Last Documented                                                 0.001178
State Conservation Rank                                             -0.011586
Global Conservation Rank                                            -0.036790
…                                                                         …
Taxonomic Subgroup_Waxwings                                         -0.001490
Taxonomic Subgroup_Whales and Dolphins                              -0.000500
Taxonomic Subgroup_Wood-Warblers                                    -0.007329
```

```
Taxonomic Subgroup_Woodpeckers                                        -0.003936
Taxonomic Subgroup_Wrens                                              -0.003125

                                        Taxonomic Subgroup_Swallows  \
County                                                    -0.002923
Category                                                  -0.090471
Year Last Documented                                       0.031081
State Conservation Rank                                    0.103169
Global Conservation Rank                                   0.019430
…                                                               …
Taxonomic Subgroup_Waxwings                               -0.007182
Taxonomic Subgroup_Whales and Dolphins                    -0.002410
Taxonomic Subgroup_Wood-Warblers                          -0.035331
Taxonomic Subgroup_Woodpeckers                            -0.018974
Taxonomic Subgroup_Wrens                                  -0.015065

                                        Taxonomic Subgroup_Thrushes and
Bluebirds  \
County
-0.004108
Category
-0.089392
Year Last Documented
0.030931
State Conservation Rank
0.102260
Global Conservation Rank
-0.019064
…
…
Taxonomic Subgroup_Waxwings
-0.007097
Taxonomic Subgroup_Whales and Dolphins
-0.002381
Taxonomic Subgroup_Wood-Warblers
-0.034910
Taxonomic Subgroup_Woodpeckers
-0.018747
Taxonomic Subgroup_Wrens
-0.014885

                                        Taxonomic Subgroup_Vireos  \
County                                                  -0.004784
Category                                                -0.078759
Year Last Documented                                     0.027057
State Conservation Rank                                  0.087949
Global Conservation Rank                                 0.016914
```

```
…                                                  …
Taxonomic Subgroup_Waxwings                -0.006252
Taxonomic Subgroup_Whales and Dolphins     -0.002098
Taxonomic Subgroup_Wood-Warblers           -0.030758
Taxonomic Subgroup_Woodpeckers             -0.016517
Taxonomic Subgroup_Wrens                   -0.013115

                                        Taxonomic Subgroup_Waxwings  \
County                                                    -0.001483
Category                                                  -0.038199
Year Last Documented                                       0.013123
State Conservation Rank                                    0.045671
Global Conservation Rank                                   0.008204
…                                                                 …
Taxonomic Subgroup_Waxwings                                1.000000
Taxonomic Subgroup_Whales and Dolphins                    -0.001018
Taxonomic Subgroup_Wood-Warblers                          -0.014918
Taxonomic Subgroup_Woodpeckers                            -0.008011
Taxonomic Subgroup_Wrens                                  -0.006361

                                        Taxonomic Subgroup_Whales and Dolphins
\
County                                                               -0.031087
Category                                                             -0.012818
Year Last Documented                                                  0.008727
State Conservation Rank                                               0.018343
Global Conservation Rank                                             -0.036442
…                                                                            …
Taxonomic Subgroup_Waxwings                                          -0.001018
Taxonomic Subgroup_Whales and Dolphins                                1.000000
Taxonomic Subgroup_Wood-Warblers                                     -0.005006
Taxonomic Subgroup_Woodpeckers                                       -0.002688
Taxonomic Subgroup_Wrens                                             -0.002134

                                        Taxonomic Subgroup_Wood-Warblers  \
County                                                         -0.016140
Category                                                       -0.187910
Year Last Documented                                            0.063691
State Conservation Rank                                         0.209278
Global Conservation Rank                                        0.046038
…                                                                      …
Taxonomic Subgroup_Waxwings                                    -0.014918
Taxonomic Subgroup_Whales and Dolphins                         -0.005006
Taxonomic Subgroup_Wood-Warblers                                1.000000
Taxonomic Subgroup_Woodpeckers                                 -0.039409
Taxonomic Subgroup_Wrens                                       -0.031290
```

```
                                          Taxonomic Subgroup_Woodpeckers  \
County                                                         -0.006499
Category                                                       -0.100912
Year Last Documented                                            0.036380
State Conservation Rank                                         0.126456
Global Conservation Rank                                        0.021672
…                                                                    …
Taxonomic Subgroup_Waxwings                                    -0.008011
Taxonomic Subgroup_Whales and Dolphins                         -0.002688
Taxonomic Subgroup_Wood-Warblers                               -0.039409
Taxonomic Subgroup_Woodpeckers                                  1.000000
Taxonomic Subgroup_Wrens                                       -0.016804

                                          Taxonomic Subgroup_Wrens
County                                                   -0.002774
Category                                                 -0.080123
Year Last Documented                                      0.027961
State Conservation Rank                                   0.067755
Global Conservation Rank                                  0.017207
…                                                              …
Taxonomic Subgroup_Waxwings                              -0.006361
Taxonomic Subgroup_Whales and Dolphins                   -0.002134
Taxonomic Subgroup_Wood-Warblers                         -0.031290
Taxonomic Subgroup_Woodpeckers                           -0.016804
Taxonomic Subgroup_Wrens                                  1.000000

[121 rows x 121 columns]
```

[71]:
```
#plt.figure(figsize=(30,20))
#sns.heatmap(corre.round(2),annot=True)
```

17 Identify the Correlation Pairs from the Correlation Matrix

corr_pairs, list is generated using nested loops to iterate through the columns of the correlation matrix and identify correlated pairs based on the specified threshold (0.80).

[72]:
```
#Feature Selection using Pearson Correlation
corr_pairs=[]

for i in range(len(corre.columns)):
  for j in range(i):
    if corre.iloc[i,j]>0.90:
      corr_pairs.append((corre.columns[i],corre.columns[j],corre.iloc[i,j]))
corr_pairs
```

[72]:
```
[('Taxonomic Subgroup_Frogs and Toads',
  'NY Listing Status_Game with open season',
  0.9058501085438643)]
```

18 Correlation-Based Feature Dropping

Implements a feature dropping strategy based on correlation analysis, ensuring effective feature selection by keeping only one feature from each correlated pair.

[73]:
```
# Initialize a list to keep track of features to drop
features_to_drop = []

# Iterate through correlated pairs
for pair in corr_pairs:
    feature1, feature2, correlation = pair
    if feature1 not in features_to_drop:
        features_to_drop.append(feature2)  # Add feature2 to drop list if␣
 ↪feature1 is not already marked for dropping
    else:
        features_to_drop.append(feature1)  # Otherwise, add feature1 to drop␣
 ↪list

# Now you have a list of unique features to drop
print("Features to drop:", features_to_drop)
```

```
Features to drop: ['NY Listing Status_Game with open season']
```

Drop the highest correlated features from the list.

[74]:
```
df2 = df2.drop(features_to_drop, axis=1)
```

[75]:
```
#df2.drop(['Taxonomic Group_Flowering Plants', 'State Conservation␣
 ↪Rank_SNR'],axis=1,inplace=True)
```

[76]:
```
df2.shape
```

[76]:
```
(20507, 120)
```

[77]:
```
x1= df2.drop(['Distribution Status'],axis=1)
x1
```

[77]:
```
       County  Category  Year Last Documented  State Conservation Rank  \
0           0         0                  1999                       24
1           0         0                  1999                       21
2           0         0                  1999                       24
3           0         0                  1999                       24
4           0         0                  1999                       24
```

```
…        …       …                  …                      …
20502      65         2                    2005                      26
20503      65         2                    2005                      21
20504      65         2                    2005                       3
20505      65         2                    2005                       0
20506      65         2                    2005                       6

       Global Conservation Rank  NY Listing Status_Endangered  \
0                            24                             0
1                            24                             0
2                            24                             0
3                            24                             0
4                            24                             0
…                           …                            …
20502                        20                             1
20503                        24                             0
20504                        24                             1
20505                        24                             1
20506                        20                             0

       NY Listing Status_Game with no open season  \
0                                               0
1                                               0
2                                               0
3                                               0
4                                               0
…                                             …
20502                                           0
20503                                           0
20504                                           0
20505                                           0
20506                                           0

       NY Listing Status_Protected - no open season  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0
…                                               …
20502                                             0
20503                                             0
20504                                             0
20505                                             0
20506                                             0

       NY Listing Status_Protected Bird  \
```

```
0                                      0
1                                      0
2                                      0
3                                      0
4                                      0
…                                    …
20502                                  0
20503                                  0
20504                                  0
20505                                  0
20506                                  0

       NY Listing Status_Protected Bird - Game with open season  …  \
0                                                      0         …
1                                                      0         …
2                                                      0         …
3                                                      0         …
4                                                      0         …
…                                                   …         …
20502                                                  0         …
20503                                                  0         …
20504                                                  0         …
20505                                                  0         …
20506                                                  0         …

       Taxonomic Subgroup_Sturgeons and Paddlefish  \
0                                                0
1                                                0
2                                                0
3                                                0
4                                                0
…                                             …
20502                                            0
20503                                            0
20504                                            0
20505                                            0
20506                                            0

       Taxonomic Subgroup_Subtidal Wetlands  Taxonomic Subgroup_Swallows  \
0                                         0                            0
1                                         0                            0
2                                         0                            0
3                                         0                            0
4                                         0                            0
…                                      …                         …
20502                                     0                            0
20503                                     0                            0
```

```
20504                                    0                            0
20505                                    0                            0
20506                                    0                            0

       Taxonomic Subgroup_Thrushes and Bluebirds  Taxonomic Subgroup_Vireos  \
0                                              0                          0
1                                              0                          0
2                                              0                          0
3                                              0                          0
4                                              0                          0
…                                            …                          …
20502                                          0                          0
20503                                          0                          0
20504                                          0                          0
20505                                          0                          0
20506                                          0                          0

       Taxonomic Subgroup_Waxwings  Taxonomic Subgroup_Whales and Dolphins  \
0                                0                                       0
1                                0                                       0
2                                0                                       0
3                                0                                       0
4                                0                                       0
…                              …                                     …
20502                            0                                       0
20503                            0                                       0
20504                            0                                       0
20505                            0                                       0
20506                            0                                       0

       Taxonomic Subgroup_Wood-Warblers  Taxonomic Subgroup_Woodpeckers  \
0                                     0                               0
1                                     0                               0
2                                     0                               0
3                                     0                               0
4                                     0                               0
…                                   …                             …
20502                                 0                               0
20503                                 0                               0
20504                                 0                               0
20505                                 0                               0
20506                                 0                               0

       Taxonomic Subgroup_Wrens
0                             0
1                             0
2                             0
```

```
3                              0
4                              0
…                         …
20502                          0
20503                          0
20504                          0
20505                          0
20506                          0

[20507 rows x 119 columns]
```

[78]:
```
y1=df2['Distribution Status']
y1
```

[78]:
```
0        3
1        3
2        3
3        3
4        3
        ..
20502    1
20503    1
20504    1
20505    1
20506    3
Name: Distribution Status, Length: 20507, dtype: int64
```

19 Feature Selection using SelectkBest and chi2

Performing feature selection using SelectKBest with the chi2 scoring function, where it selects the top 60 features based on their chi-squared scores.

[79]:
```
# Feature Selection using SelectKBest and chi2
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(score_func=chi2, k=100)
x_new = selector.fit_transform(x1, y1)
```

[80]:
```
columns_to_check=df2.columns[df2.columns != 'Distribution Status']
```

20 Display Feature Scores

Creates a DataFrame called feature_scores_desc containing the feature names and their corresponding scores from SelectKBest. It then sorts this DataFrame by score in descending order and prints the feature scores in descending order.

[81]:

```python
# Display feature scores from SelectKBest in descending order
# Create a DataFrame to store the feature scores
feature_scores_desc = pd.DataFrame({'Feature': columns_to_check, 'Score':␣
 ↪selector.scores_})
feature_scores_desc = feature_scores_desc.sort_values(by='Score',␣
 ↪ascending=False)
# Display the entire DataFrame without truncation
pd.set_option('display.max_rows', None)  # Set option to display all rows
print("Feature Scores from SelectKBest (Descending Order):")
print(feature_scores_desc)
```

```
Feature Scores from SelectKBest (Descending Order):
                                                 Feature         Score
3                                State Conservation Rank  20827.181391
1                                               Category   9297.483508
5                           NY Listing Status_Endangered   3215.864893
84             Taxonomic Subgroup_Other Flowering Plants   3154.321431
8                       NY Listing Status_Protected Bird   1843.716007
2                                    Year Last Documented   1084.054021
82                             Taxonomic Subgroup_Orchids    836.513755
12                          NY Listing Status_Threatened    823.292534
100                             Taxonomic Subgroup_Sedges    737.347854
10                                NY Listing Status_Rare    696.668091
15    Taxonomic Subgroup_Asters, Goldenrods and Daisies    578.804113
17                                Taxonomic Subgroup_Bats    407.226703
116                      Taxonomic Subgroup_Wood-Warblers    325.062381
9     NY Listing Status_Protected Bird - Game with o…   308.562995
6             NY Listing Status_Game with no open season    306.607232
103                        Taxonomic Subgroup_Silversides    269.007502
37                               Taxonomic Subgroup_Ferns    243.295561
68           Taxonomic Subgroup_Minnows, Shiners, Suckers    235.781581
75                         Taxonomic Subgroup_Needlefishes    231.240787
0                                                  County    229.589078
92                    Taxonomic Subgroup_Rabbits and Hares    201.240243
106                Taxonomic Subgroup_Sparrows and Towhees    172.190736
51    Taxonomic Subgroup_Hawks, Falcons, Eagles, Vul…   166.425545
97                         Taxonomic Subgroup_Salamanders    160.002523
36            Taxonomic Subgroup_Ducks, Geese, Waterfowl    156.472473
47                             Taxonomic Subgroup_Grasses    149.455399
40                         Taxonomic Subgroup_Flycatchers    133.352762
4                                 Global Conservation Rank    131.622488
11                      NY Listing Status_Special Concern    125.922718
45                      Taxonomic Subgroup_Frogs and Toads    124.593615
105                             Taxonomic Subgroup_Snakes    124.096909
19             Taxonomic Subgroup_Blackbirds and Orioles    112.039126
98                     Taxonomic Subgroup_Salmon and Trout     93.324266
111                           Taxonomic Subgroup_Swallows     93.156818
```

112	Taxonomic Subgroup_Thrushes and Bluebirds	90.984064
86	Taxonomic Subgroup_Owls	83.001131
38	Taxonomic Subgroup_Finches and Crossbills	79.848701
50	Taxonomic Subgroup_Gulls, Terns, Plovers, Shor…	77.654144
117	Taxonomic Subgroup_Woodpeckers	77.204548
13	NY Listing Status_Unknown	74.572754
113	Taxonomic Subgroup_Vireos	70.886092
21	Taxonomic Subgroup_Cardinals and Buntings	67.626961
30	Taxonomic Subgroup_Crows and Jays	58.935946
52	Taxonomic Subgroup_Herons, Bitterns, Egrets, P…	55.276484
49	Taxonomic Subgroup_Grouse, Pheasants, Turkeys	54.798118
118	Taxonomic Subgroup_Wrens	54.731758
83	Taxonomic Subgroup_Other Animals	53.535554
93	Taxonomic Subgroup_Rails, Coots and Cranes	52.416994
69	Taxonomic Subgroup_Mockingbirds and Thrashers	50.244931
96	Taxonomic Subgroup_Rushes	45.484179
23	Taxonomic Subgroup_Carrion Beetles	40.782635
53	Taxonomic Subgroup_Herrings and Shad	40.782635
104	Taxonomic Subgroup_Snails	37.906098
91	Taxonomic Subgroup_Quillworts	37.756941
25	Taxonomic Subgroup_Chickadees and Titmice	35.307249
27	Taxonomic Subgroup_Conifers	35.024214
99	Taxonomic Subgroup_Sculpins	34.503630
90	Taxonomic Subgroup_Pigeons and Doves	33.949278
55	Taxonomic Subgroup_Hummingbirds and Swifts	32.591307
31	Taxonomic Subgroup_Cuckoos	32.591307
77	Taxonomic Subgroup_Nuthatches	32.319712
101	Taxonomic Subgroup_Shrews and Moles	30.586976
44	Taxonomic Subgroup_Freshwater Mussels	30.581590
71	Taxonomic Subgroup_Moths	28.837167
22	Taxonomic Subgroup_Carnivores	28.616791
39	Taxonomic Subgroup_Flies	28.243628
16	Taxonomic Subgroup_Barrens and Woodlands	21.785128
35	Taxonomic Subgroup_Dragonflies	21.751936
59	Taxonomic Subgroup_Kinglets	20.097972
33	Taxonomic Subgroup_Darters and Sunfishes	19.792191
81	Taxonomic Subgroup_Open Uplands	19.077983
32	Taxonomic Subgroup_Damselflies	17.924637
43	Taxonomic Subgroup_Forested Uplands	17.279282
114	Taxonomic Subgroup_Waxwings	16.838842
107	Taxonomic Subgroup_Starlings	16.838842
78	Taxonomic Subgroup_Old World Sparrows	16.838842
54	Taxonomic Subgroup_Horsetails	16.785964
18	Taxonomic Subgroup_Bees	16.112125
48	Taxonomic Subgroup_Grebes	16.024059
58	Taxonomic Subgroup_Kingfishers	15.752465
29	Taxonomic Subgroup_Creepers	15.480871
46	Taxonomic Subgroup_Gnatcatchers	15.209276

```
26                                  Taxonomic Subgroup_Clubmosses     14.282201
108                                 Taxonomic Subgroup_Stoneflies     13.826167
94                                     Taxonomic Subgroup_Rodents     13.786549
80                              Taxonomic Subgroup_Open Peatlands     13.579225
41    Taxonomic Subgroup_Forested Mineral Soil Wetlands     12.643292
62                                       Taxonomic Subgroup_Larks     12.493334
109                  Taxonomic Subgroup_Sturgeons and Paddlefish     10.673893
28                                  Taxonomic Subgroup_Cormorants     10.048986
79                Taxonomic Subgroup_Open Mineral Soil Wetlands      9.771501
20                  Taxonomic Subgroup_Butterflies and Skippers      8.170970
14                        Taxonomic Subgroup_Animal Assemblages      7.128998
85                              Taxonomic Subgroup_Other Mosses      6.897372
64                                       Taxonomic Subgroup_Loons      6.789856
73                 Taxonomic Subgroup_Natural Lakes and Ponds      6.767954
76                                  Taxonomic Subgroup_Nightbirds      6.308303
74              Taxonomic Subgroup_Natural Rivers and Streams      5.759304
57                                 Taxonomic Subgroup_Killifishes      5.097829
42                          Taxonomic Subgroup_Forested Peatlands      4.900427
110                          Taxonomic Subgroup_Subtidal Wetlands      4.073913
7                NY Listing Status_Protected - no open season      3.259131
61                                   Taxonomic Subgroup_Lampreys      2.715942
60                               Taxonomic Subgroup_Lady Beetles      2.715942
56                       Taxonomic Subgroup_Intertidal Wetlands      2.541053
24                                  Taxonomic Subgroup_Catfishes      2.184206
87                     Taxonomic Subgroup_Parrots and Parakeets      2.172754
66                            Taxonomic Subgroup_Marine Subtidal      2.172754
115                      Taxonomic Subgroup_Whales and Dolphins      1.901160
72                              Taxonomic Subgroup_Natural Caves      1.901160
63                                    Taxonomic Subgroup_Lizards      1.826646
88                                Taxonomic Subgroup_Peat Mosses      1.786060
102                                   Taxonomic Subgroup_Shrikes      1.114296
89                                    Taxonomic Subgroup_Perches      1.025882
70                                   Taxonomic Subgroup_Mooneyes      1.025882
65                          Taxonomic Subgroup_Marine Intertidal      0.771089
67                                   Taxonomic Subgroup_Mayflies      0.543188
34                              Taxonomic Subgroup_Diving Beetles      0.271594
95                                Taxonomic Subgroup_Rove Beetles      0.271594
```

21 Display Selected Features

Disply the 100 features which have relatively high scores as per the selectkbest method.

> Add blockquote

[82]:
```
# Display feature scores from SelectKBest in descending order
feature_scores_selected = feature_scores_desc.sort_values(by='Score',␣
↪ascending=False).head(60)
# Display the entire DataFrame without truncation
```

```python
pd.set_option('display.max_rows', None)  # Set option to display all rows
print("Feature Scores from SelectKBest (Descending Order):")
print(feature_scores_selected)
```

```
Feature Scores from SelectKBest (Descending Order):
                                               Feature         Score
3                              State Conservation Rank  20827.181391
1                                             Category   9297.483508
5                         NY Listing Status_Endangered   3215.864893
84            Taxonomic Subgroup_Other Flowering Plants   3154.321431
8                     NY Listing Status_Protected Bird   1843.716007
2                                  Year Last Documented   1084.054021
82                          Taxonomic Subgroup_Orchids    836.513755
12                         NY Listing Status_Threatened    823.292534
100                          Taxonomic Subgroup_Sedges    737.347854
10                               NY Listing Status_Rare    696.668091
15    Taxonomic Subgroup_Asters, Goldenrods and Daisies    578.804113
17                             Taxonomic Subgroup_Bats    407.226703
116                   Taxonomic Subgroup_Wood-Warblers    325.062381
9     NY Listing Status_Protected Bird - Game with o…  308.562995
6            NY Listing Status_Game with no open season    306.607232
103                     Taxonomic Subgroup_Silversides    269.007502
37                             Taxonomic Subgroup_Ferns    243.295561
68          Taxonomic Subgroup_Minnows, Shiners, Suckers    235.781581
75                      Taxonomic Subgroup_Needlefishes    231.240787
0                                               County    229.589078
92                   Taxonomic Subgroup_Rabbits and Hares    201.240243
106             Taxonomic Subgroup_Sparrows and Towhees    172.190736
51    Taxonomic Subgroup_Hawks, Falcons, Eagles, Vul…  166.425545
97                      Taxonomic Subgroup_Salamanders    160.002523
36            Taxonomic Subgroup_Ducks, Geese, Waterfowl    156.472473
47                          Taxonomic Subgroup_Grasses    149.455399
40                      Taxonomic Subgroup_Flycatchers    133.352762
4                             Global Conservation Rank    131.622488
11                      NY Listing Status_Special Concern    125.922718
45                   Taxonomic Subgroup_Frogs and Toads    124.593615
105                          Taxonomic Subgroup_Snakes    124.096909
19            Taxonomic Subgroup_Blackbirds and Orioles    112.039126
98                  Taxonomic Subgroup_Salmon and Trout     93.324266
111                        Taxonomic Subgroup_Swallows     93.156818
112           Taxonomic Subgroup_Thrushes and Bluebirds     90.984064
86                             Taxonomic Subgroup_Owls     83.001131
38            Taxonomic Subgroup_Finches and Crossbills     79.848701
50    Taxonomic Subgroup_Gulls, Terns, Plovers, Shor…   77.654144
117                     Taxonomic Subgroup_Woodpeckers     77.204548
13                         NY Listing Status_Unknown     74.572754
113                          Taxonomic Subgroup_Vireos     70.886092
```

```
21           Taxonomic Subgroup_Cardinals and Buntings     67.626961
30                   Taxonomic Subgroup_Crows and Jays     58.935946
52    Taxonomic Subgroup_Herons, Bitterns, Egrets, P…     55.276484
49        Taxonomic Subgroup_Grouse, Pheasants, Turkeys     54.798118
118                          Taxonomic Subgroup_Wrens     54.731758
83                   Taxonomic Subgroup_Other Animals     53.535554
93           Taxonomic Subgroup_Rails, Coots and Cranes     52.416994
69        Taxonomic Subgroup_Mockingbirds and Thrashers     50.244931
96                          Taxonomic Subgroup_Rushes     45.484179
23                 Taxonomic Subgroup_Carrion Beetles     40.782635
53               Taxonomic Subgroup_Herrings and Shad     40.782635
104                         Taxonomic Subgroup_Snails     37.906098
91                      Taxonomic Subgroup_Quillworts     37.756941
25           Taxonomic Subgroup_Chickadees and Titmice     35.307249
27                        Taxonomic Subgroup_Conifers     35.024214
99                        Taxonomic Subgroup_Sculpins     34.503630
90               Taxonomic Subgroup_Pigeons and Doves     33.949278
31                         Taxonomic Subgroup_Cuckoos     32.591307
55         Taxonomic Subgroup_Hummingbirds and Swifts     32.591307
```

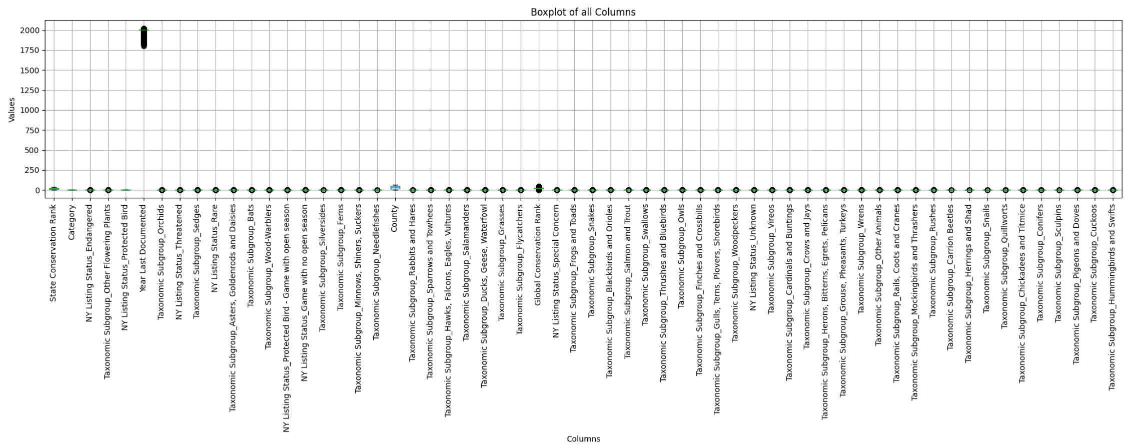
Generate a variable named columns_to_check that holds the column names from df2 excluding 'Distribution Status'(target).

[83]:
```
# Select columns excluding the target variable
selected_features = feature_scores_selected[feature_scores_selected['Feature'] !
↪= 'Distribution Status']['Feature'].tolist()
# Filtering df2 to include only the selected features
df2_selected = df2[selected_features]
#df_selected.head()
```

22 Box Plot Before Applying IQR

Plot boxplots for df2.

[84]:
```
# Plotting the boxplot
plt.figure(figsize=(20, 8))
df2_selected.boxplot()
plt.title('Boxplot of all Columns')
plt.xlabel('Columns')
plt.ylabel('Values')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.tight_layout()  # Adjust layout to prevent overlapping labels
plt.show()
```



23 Outlier Removal using IQR

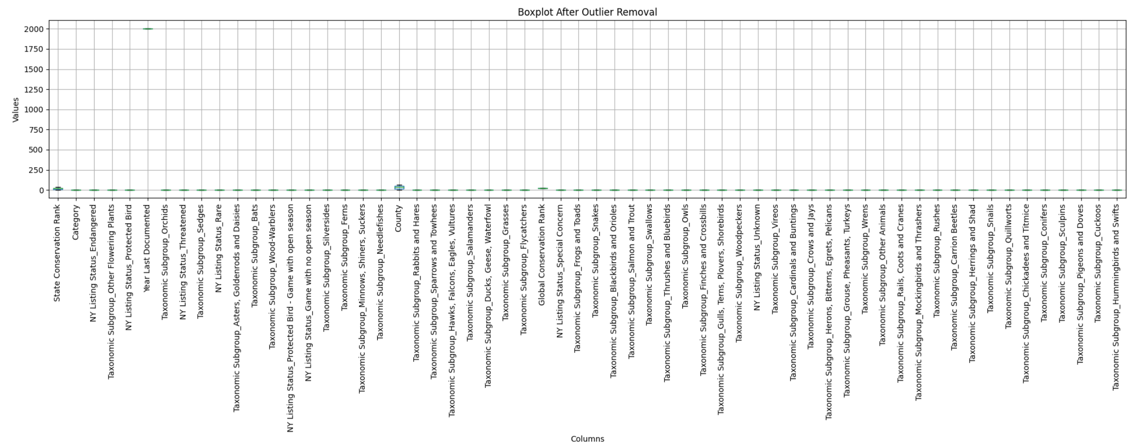
The iqr_rem function performs outlier removal using the interquartile range (IQR) method.And clips the values in the column col to be within the lower and upper bounds, effectively removing outliers.

[85]:
```python
#method to remove outliers
def iqr_rem(df, cols):
    for col in cols:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        iqr = q3 - q1
        upper_bound = q3 + (1.5 * iqr)
        lower_bound = q1 - (1.5 * iqr)
        df.loc[:, col] = df[col].clip(lower_bound, upper_bound) # Got a warning␣
 ↪msg while using df[col] only as
                                                              #Try using .
 ↪loc[row_indexer,col_indexer] = value instead
    return df
df2_cleaned = iqr_rem(df2_selected, selected_features)
```

24 Box Plot After Applying Outlier Removal

[86]:
```python
# Plot boxplot after outlier removal
plt.figure(figsize=(20, 8))
df2_cleaned.boxplot()
plt.title('Boxplot After Outlier Removal')
plt.xlabel('Columns')
plt.ylabel('Values')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
```

```
plt.tight_layout()  # Adjust layout to prevent overlapping labels

plt.show()
```



Observations: * After applying the Interquartile Range (IQR) method for outlier removal, it is observed that the outliers have been successfully removed from the DataFrame.

Separating X and Y with selected features

[87]:
```
X = df2[selected_features].values  # Use selected_features_df from the previous␣
↪code
X
```

[87]:
```
array([[24,  0,  0, …,  0,  0,  0],
       [21,  0,  0, …,  0,  0,  0],
       [24,  0,  0, …,  0,  0,  0],
       …,
       [ 3,  2,  1, …,  0,  0,  0],
       [ 0,  2,  1, …,  0,  0,  0],
       [ 6,  2,  0, …,  0,  0,  0]])
```

[88]:
```
Y = df2['Distribution Status'].values
Y
```

[88]:
```
array([3, 3, 3, …, 1, 1, 3])
```

25 Class Distribution Using SMOTE Oversampling

Applies SMOTE to the feature matrix x and the target variable y, generating synthetic samples for the minority class to balance the class distribution. And converts the oversampled target variable y__sm to a Pandas Series for easier manipulation and analysis. Prints the class distribution of the oversampled target variable y__sm to check the effectiveness of the oversampling technique

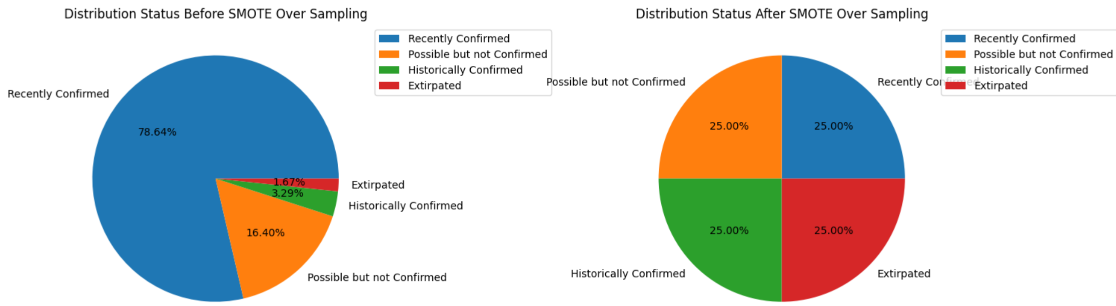
```
[89]: from imblearn.over_sampling import SMOTE
smote=SMOTE(random_state=42)
x_sm,y_sm=smote.fit_resample(X,Y)
# Check the class distribution after oversampling
y_sm=pd.Series(y_sm)
print(y_sm.value_counts())
```

```
3    16127
2    16127
1    16127
0    16127
Name: count, dtype: int64
```

26 Visualize the Target Column Before and After Applying SMOTE

To visualize the 'Distribution Status' column before SMOTE oversampling using a pie chart and adjust the legend's position to the upper left corner.

```
[90]: # Pie chart before SMOTE oversampling
plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1)
plt.pie(df['Distribution Status'].value_counts(), autopct='%1.2f%%',␣
 ↪labels=['Recently Confirmed', 'Possible but not Confirmed', 'Historically␣
 ↪Confirmed', 'Extirpated'])
plt.title('Distribution Status Before SMOTE Over Sampling')
plt.legend(loc='upper left',bbox_to_anchor=(1, 1))

# Pie chart after SMOTE oversampling
plt.subplot(1, 2, 2)
plt.pie(y_sm.value_counts(), autopct='%1.2f%%', labels=['Recently Confirmed',␣
 ↪'Possible but not Confirmed', 'Historically Confirmed', 'Extirpated'])
plt.title('Distribution Status After SMOTE Over Sampling')
plt.legend(loc='upper left',bbox_to_anchor=(1, 1))

plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```



#Train-Test Split for Oversampled Data train_test_split function to split the oversampled data into training and testing sets.

[91]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_sm,y_sm,test_size=0.
 ↪30,random_state=42,stratify=y_sm)
```

27 Decoding Encoded Target Variable

Decodes the encoded target variable y_train using the inverse_transform method from the encoding object (encode)

[92]:
```
y_decode=encode.inverse_transform(y_train)
y_decode[:50]
```

[92]:
```
array(['Possible but not Confirmed', 'Recently Confirmed',
       'Recently Confirmed', 'Possible but not Confirmed', 'Extirpated',
       'Possible but not Confirmed', 'Historically Confirmed',
       'Historically Confirmed', 'Recently Confirmed',
       'Recently Confirmed', 'Historically Confirmed',
       'Recently Confirmed', 'Recently Confirmed',
       'Historically Confirmed', 'Extirpated', 'Recently Confirmed',
       'Historically Confirmed', 'Extirpated', 'Recently Confirmed',
       'Historically Confirmed', 'Extirpated', 'Recently Confirmed',
       'Recently Confirmed', 'Extirpated', 'Historically Confirmed',
       'Possible but not Confirmed', 'Extirpated',
       'Possible but not Confirmed', 'Extirpated', 'Recently Confirmed',
       'Recently Confirmed', 'Historically Confirmed', 'Extirpated',
       'Extirpated', 'Possible but not Confirmed',
       'Possible but not Confirmed', 'Recently Confirmed', 'Extirpated',
       'Possible but not Confirmed', 'Historically Confirmed',
       'Extirpated', 'Recently Confirmed', 'Historically Confirmed',
       'Possible but not Confirmed', 'Extirpated', 'Extirpated',
       'Extirpated', 'Extirpated', 'Recently Confirmed',
```

```
'Historically Confirmed'], dtype=object)
```

28 Feature Scaling using StandardScaler

StandardScaler is ensuring that each feature contributes equally to the analysis and preventing certain features from dominating due to their larger scales.

[93]:
```
from sklearn.preprocessing import StandardScaler
norm=StandardScaler()
norm.fit(x_train)
x_train=norm.transform(x_train)
x_test=norm.transform(x_test)
```

29 Model Creation

Import KNN,Decision Tree, Random Forest, SVM, Naive Bayes Classifiers for model creation and imports various metrics and tools for evaluating classification models, such as accuracy score, confusion matrix, classification report, and ConfusionMatrixDisplay.

[94]:
```
#Model creation
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay
model1=KNeighborsClassifier(n_neighbors=9)
model2=DecisionTreeClassifier(criterion='entropy')
model3=RandomForestClassifier(n_estimators=100,criterion='entropy',random_state=42)
model4=SVC()
model5=BernoulliNB()
lst=[model1,model2,model3,model4,model5]
```

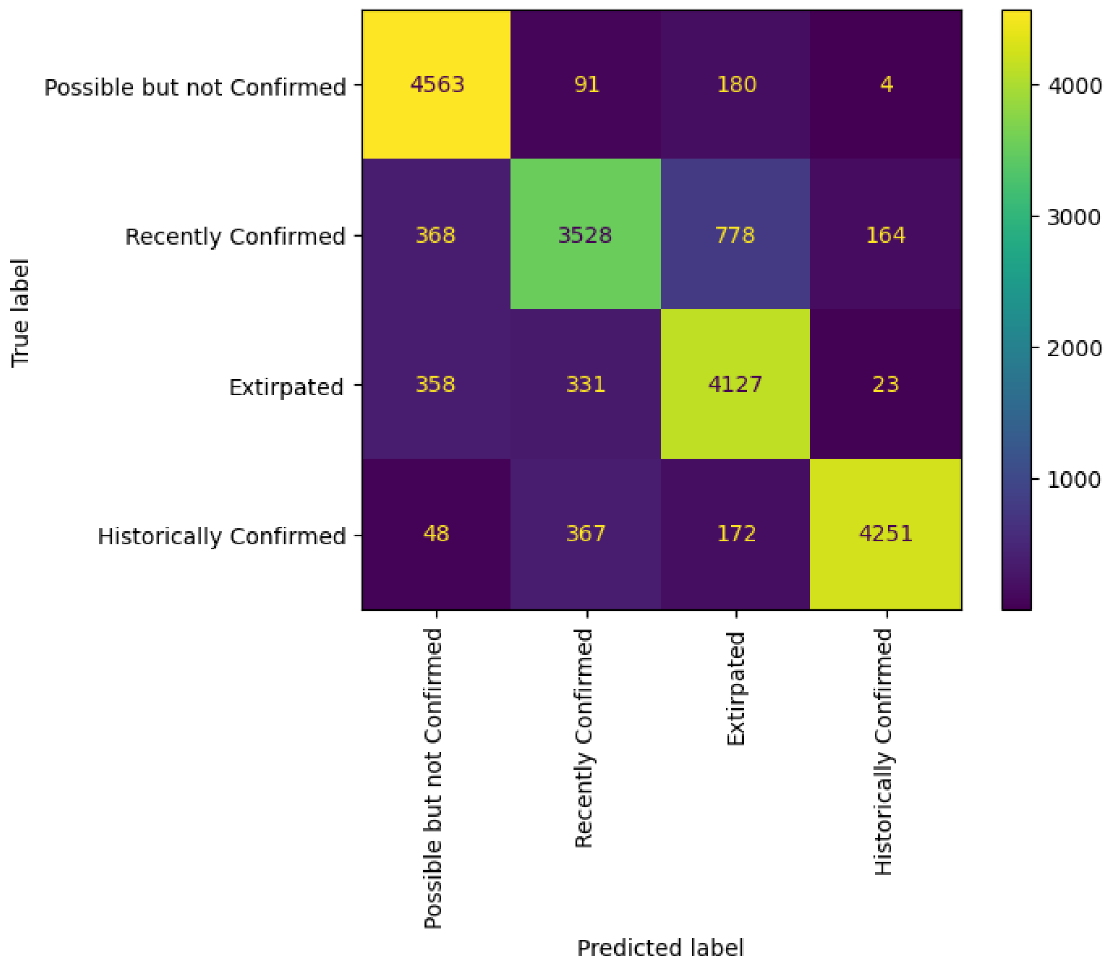
30 Model Evaluation and Comparison for Classification

Compare the performance of multiple classification models by evaluating their accuracy, confusion matrix, and classification report on a testing dataset.

[95]:
```
for i in lst:
  print("model is",i)
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  cm=confusion_matrix(y_test,y_pred)
  print("Accuracy score is",accuracy_score(y_test,y_pred))
  print(cm)
```

```python
labels=['Possible but not Confirmed','Recently Confirmed','Extirpated',␣
↪'Historically Confirmed']
print(classification_report(y_test,y_pred))
cmd=ConfusionMatrixDisplay(cm,display_labels=labels)
cmd.plot(xticks_rotation = 'vertical')
#cmd.plot()
plt.show()
```

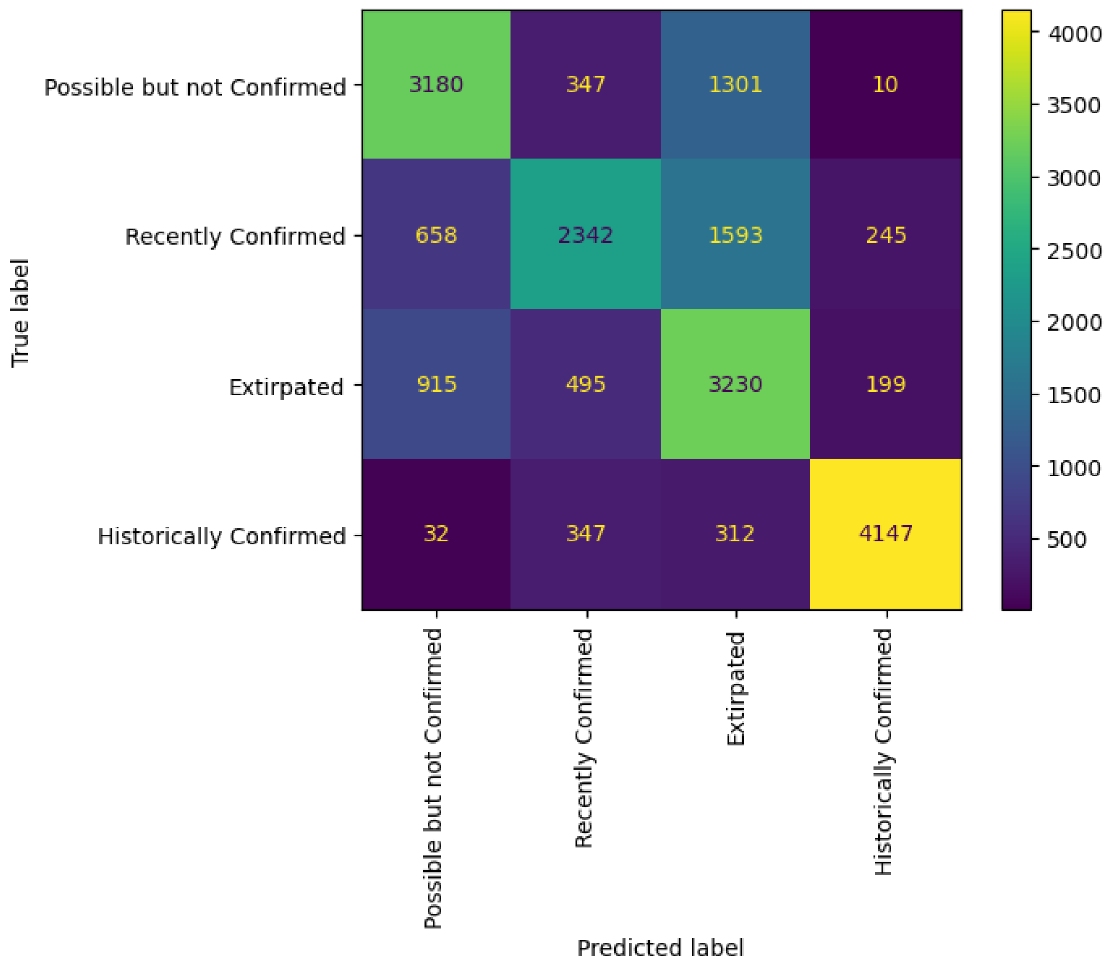
```
model is KNeighborsClassifier(n_neighbors=9)
Accuracy score is 0.850979176355087
[[4563   91  180    4]
 [ 368 3528  778  164]
 [ 358  331 4127   23]
 [  48  367  172 4251]]
              precision    recall  f1-score   support

           0       0.85      0.94      0.90      4838
           1       0.82      0.73      0.77      4838
           2       0.79      0.85      0.82      4839
           3       0.96      0.88      0.92      4838

    accuracy                           0.85     19353
   macro avg       0.85      0.85      0.85     19353
weighted avg       0.85      0.85      0.85     19353
```




```
model is DecisionTreeClassifier(criterion='entropy')
Accuracy score is 0.8990854131142458
[[4543   89  199    7]
 [ 188 3939  583  128]
 [ 193  302 4329   15]
 [  14  169   66 4589]]
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      4838
           1       0.88      0.81      0.84      4838
           2       0.84      0.89      0.86      4839
           3       0.97      0.95      0.96      4838

    accuracy                           0.90     19353
   macro avg       0.90      0.90      0.90     19353
weighted avg       0.90      0.90      0.90     19353
```


```
model is RandomForestClassifier(criterion='entropy', random_state=42)
Accuracy score is 0.9061644189531339
[[4568   72  192    6]
 [ 177 3978  577  106]
 [ 188  249 4374   28]
 [  13  157   51 4617]]
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      4838
           1       0.89      0.82      0.86      4838
           2       0.84      0.90      0.87      4839
           3       0.97      0.95      0.96      4838

    accuracy                           0.91     19353
   macro avg       0.91      0.91      0.91     19353
weighted avg       0.91      0.91      0.91     19353
```


```
model is SVC()
Accuracy score is 0.6665116519402676
[[3180  347 1301   10]
 [ 658 2342 1593  245]
 [ 915  495 3230  199]
 [  32  347  312 4147]]
              precision    recall  f1-score   support

           0       0.66      0.66      0.66      4838
           1       0.66      0.48      0.56      4838
           2       0.50      0.67      0.57      4839
           3       0.90      0.86      0.88      4838

    accuracy                           0.67     19353
   macro avg       0.68      0.67      0.67     19353
weighted avg       0.68      0.67      0.67     19353
```




```
model is BernoulliNB()
Accuracy score is 0.5637368883377254
[[2439  725 1593   81]
 [1010 1504 1891  433]
 [ 874  413 3168  384]
 [ 352  286  401 3799]]
              precision    recall  f1-score   support

           0       0.52      0.50      0.51      4838
           1       0.51      0.31      0.39      4838
           2       0.45      0.65      0.53      4839
           3       0.81      0.79      0.80      4838

    accuracy                           0.56     19353
   macro avg       0.57      0.56      0.56     19353
weighted avg       0.57      0.56      0.56     19353
```

Observations:-

- Random Forest classifier excels with an accuracy score of 90.61% in classification tasks, demonstrating superior predictive power compared to other models.
- Decision Tree classifier achieves competitive accuracy at 89.89%, showcasing its effectiveness in capturing decision boundaries.
- KNeighborsClassifier and SVC exhibit lower accuracies, with KNeighborsClassifier emphasizing local similarities and SVC using hyperplanes for classification, indicating potential limitations in handling the dataset's complexity.
- BernoulliNB model shows the lowest accuracy at 56.37%, underscoring challenges in accurately classifying the dataset compared to Random Forest's robust performance.
- Random Forest maintains balanced precision, recall, and F1-scores across classes, showcasing its reliability in diverse classification scenarios.

31 Hyper Parameter Tuning using GridSearchCV

[96]:
```python
# Define parameter grids for random forest algorithms
param = {'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'max_features': ['sqrt', 'log2']}
```

[97]:
```python
# Initialize models for different algorithms
model_rf = RandomForestClassifier(random_state=42)
```

[98]:
```python
# Initialize GridSearchCV with different parameter grid
from sklearn.model_selection import GridSearchCV
grid_search_rf = GridSearchCV(model_rf, param, cv=10, scoring='accuracy',␣
 ↪n_jobs=-1)
# Fit GridSearchCV on the training data for each algorithm
grid_search_rf.fit(x_train, y_train)
```

[98]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=42),
             n_jobs=-1,
             param_grid={'max_depth': [None, 10, 20],
                         'max_features': ['sqrt', 'log2'],
                         'min_samples_split': [2, 5],
                         'n_estimators': [50, 100]},
             scoring='accuracy')
```

[99]:
```python
# Get best parameters and best scores for each algorithm
best_params_rf = grid_search_rf.best_params_
best_params_rf
```

[99]:
```
{'max_depth': None,
 'max_features': 'sqrt',
 'min_samples_split': 2,
 'n_estimators': 100}
```

[100]:
```python
#Model creation using the best parameters obtained from GridSearchCV
model_rf1=RandomForestClassifier(max_depth=None,max_features='sqrt',min_samples_split=2,n_esti
model_rf1.fit(x_train,y_train)
y_pred2=model_rf1.predict(x_test)
y_pred2
```

[100]:
```
array([3, 1, 3, …, 3, 3, 1])
```

[101]:
```python
#Print the accuracy score for the new model
print("score is",accuracy_score(y_test,y_pred2))
```

```
score is 0.9071461788869942
```

32 Conclusion

Before tuning, the Random Forest model achieved an accuracy score of 90.61%. After tuning with GridSearchCV, the accuracy improved marginally to 90.71%. The Random Forest classifier maintained balanced precision, recall, and F1-scores across all classes both before and after tuning, highlighting its reliability in predicting biodiversity distribution statuses among animals, plants, and natural communities. Compared to other models like KNeighborsClassifier, DecisionTreeClassifier, SVC, and BernoulliNB, Random Forest consistently outperformed them, making it the preferred and robust choice for accurate and dependable biodiversity distribution predictions

33 Using Joblib to Save, Load, and Predict with the Random Forest Model

Joblib particularly useful for saving trained machine learning models to disk and then loading them back into memory for later use without having to retrain the model.

[102]:
```
import joblib
#save the trained Random Forest model (model_rf1) to a file named␣
↪'random_forest_model.joblib'
joblib.dump(model_rf1, 'random_forest_model.joblib')
```

[102]:
```
['random_forest_model.joblib']
```

[103]:
```
#Saved model is loaded back into memory using joblib.load as loaded_model
loaded_model = joblib.load('random_forest_model.joblib')
```

[104]:
```
#Predictions on x_test without retraining the model.
predictions = loaded_model.predict(x_test)
predictions
```

[104]:
```
array([3, 1, 3, …, 3, 3, 1])
```