19.03.2024

ÇAĞLA ÇAĞLAR

IBM – UNSUPERVISED MACHINE LEARNING

A Cluster Analysis of Meteorite Landings

Introduction

This report has been compiled using a dataset from the Kaggle platform, which details over 45,000 meteorite landings recorded by The Meteoritical Society. The dataset is a comprehensive collection of data pertaining to meteorites that have made contact with the Earth, documenting their various characteristics. The complete Python code utilized for the analysis, along with the generated visualizations, are included at the conclusion of this report.

Dataset Overview

The dataset under analysis provides a historical record of meteorite landings, with specifics on location, mass, composition, and the year of fall. Such a rich dataset offers an unparalleled opportunity to study the frequency, distribution, and types of meteorite impacts on Earth.

Upon meticulous examination of the dataset, certain discrepancies were noted. Some entries included dates outside the expected range (before 860 CE or after 2016), likely due to incorrect parsing into the NASA database. These anomalies were interpreted as BCE dates and were corrected accordingly. Additionally, coordinates listed as 0N/0E, which would position the meteorites in an inaccessible region off the western coast of Africa, were identified as erroneous, often representing findings from Antarctica without precise location data. These coordinates were treated as missing data.

The dataset adopts "rec" as a prefix for certain columns, such as "recclass", "reclat", and "reclong", indicating values that are endorsed by the Meteoritical Society. These values reflect society's current recommendations, accounting for any historical reclassifications or adjustments in recovery data.

The dataset paints a portrait of each meteorite with a rich tapestry of variables. The “name” is more than just a label; it is often a nod to the discovery site, adorned with figures and facts that lend identity to each celestial stone. Each one is tagged with an “id”, a numerical fingerprint, ensuring its uniqueness in the cosmic catalogue. The “nametype” bifurcates the collection into “valid”, the standard meteorites, and “relict”, those altered by the embrace of Earth's environment. The'recclass` further delves into a detailed taxonomy, assigning each meteorite to a class based on a suite of physical and chemical attributes.

The “mass” variable weighs the significance of each meteorite in grams, anchoring these travelers from the void in the realm of measurable reality. The “fall” variable captures the drama of their descent: 'Fell' denotes a meteorite's plunge witnessed by human eyes, while “Found” denotes a silent arrival, discovered only after the fact. The “year” of the meteorite's encounter with Earth is chronicled, marking a temporal coordinate in the long history of our planet.

Grounding these events are “'reclat” and “reclong”, geographical coordinates that pinpoint the landing site within the tapestry of our world's geography. Together, they are woven into the “GeoLocation”, a single string that couples the latitude and longitude.

The data was sourced from the following URL: [Kaggle - NASA Meteorite Landings](https://www.kaggle.com/datasets/nasa/meteorite-landings).

Data Preprocessing

In the preprocessing stage of the meteorite landings dataset, initial steps involved removing entries with missing "year" values and converting this feature to an integer type. Additionally, only records from the years 860 to 2016 were kept to eliminate incorrect entries. Entries with geolocation at 0N/0E, which usually indicates missing or inaccurate data, were set to NaN. Remaining missing values were also dropped, leading to a dataset ready for analysis. Numerical features such as mass and year were then standardized using StandardScaler to neutralize the scale effects, and non-critical warnings were silenced to streamline the process. The resulting cleaned dataset contained 31,929 entries with uniform and relevant data for clustering analysis.

Discussion on Anomalies

The dataset contained anomalies such as implausible dates and inaccurate coordinates (0N/0E), which could distort the analysis. Dates outside the expected range were scrutinized and corrected to maintain historical accuracy, while coordinates at 0N/0E were treated as missing data to avoid misrepresenting the spatial distribution of landings. These corrections were crucial for ensuring the integrity of the dataset, enabling a realistic and accurate analysis of meteorite landings.

Exploratory Data Analysis

The visualizations presented provide a rich narrative of the meteorite data. The logarithmic histogram showcases the mass distribution of meteorites, highlighting a right-skewed trend indicating that smaller meteorites are far more common than larger ones. The line graph depicting meteorite findings by year reveals a significant increase in discoveries or observations in recent times, likely due to improved detection methods and greater scientific interest. The scatter plot of meteorite mass versus year illustrates the temporal distribution of meteorites of various sizes, again emphasizing more recent finds. The pie chart clearly shows that most meteorites are found rather than observed falling, underscoring the prevalence of discoveries of meteorites long after their impact. Finally, the bar chart and heatmap collectively emphasize the diversity of meteorite classes and the geographical distribution of landings, with higher concentrations in certain areas, possibly influenced by observation efforts and geographical factors.

Methodology Explanation

The clustering algorithms K-Means, Mean Shift, and DBSCAN were carefully selected for their distinctive strengths in analyzing spatial data. K-Means provides a straightforward approach for partitioning data into a predefined number of clusters, ideal for initial explorations. Mean Shift, being non-parametric, automatically identifies the number of clusters based on data density and is adaptable to the dataset's inherent structure. DBSCAN excels in distinguishing dense clusters from sparse noise and handling arbitrary shapes and outliers effectively. Each algorithm offers a unique lens through which the spatial distribution of meteorite landings can be examined, catering to the diverse and complex nature of the data.

Clustering Analysis

The clustering analysis involved the application of three distinct clustering algorithms to the standardized meteorite landings dataset. The K-Means algorithm was first applied, utilizing the Elbow Method to determine the optimal number of clusters, which was identified as four. Each meteorite was then assigned to one of these four clusters based on its location.

Following K-Means, the Mean Shift clustering algorithm was employed, determining the bandwidth from the data itself using the estimate_bandwidth function. This non-parametric method does not require the number of clusters to be specified in advance, offering a model based solely on the data structure.

Lastly, the DBSCAN algorithm was utilized, which identifies clusters as high-density areas separated by areas of low density. It is especially suited for data with clusters of similar density. For both Mean Shift and DBSCAN, the silhouette score was calculated to quantify the clustering effectiveness.

These clustering methods revealed distinct groupings within the data based on spatial coordinates, providing a comprehensive perspective of the spread and density of meteorite landings. Each method offers unique insights, with K-Means providing defined clusters, Mean Shift adjusting to data density without predefined cluster numbers, and DBSCAN emphasizing dense regions, thus catering to different analytical needs.

Performance Metrics and Model Recommendations

The silhouette score was used as a performance metric for the K-Means clustering model, yielding a score of approximately 0.564. This score suggests moderate separation between clusters. Subsequently, Mean Shift and DBSCAN clustering were also applied to the dataset. The Mean Shift model, which automatically determines the number of clusters based on the data, identified a wide range of clusters with varied sizes. The silhouette score for Mean Shift was approximately 0.647, indicating relatively well-separated clusters. DBSCAN, known for its ability to find arbitrarily shaped clusters and identify outliers, resulted in a silhouette score of approximately 0.582.

Based on the silhouette scores and the ability to automatically determine cluster numbers, the Mean Shift model is recommended. It provided a good balance between cluster separation and the number of clusters identified, allowing for a nuanced understanding of the data.

Visualization and Cluster Count

The visualization for K-Means clustering exhibits a substantial primary cluster that significantly overshadows the smaller clusters, suggesting a predominant region where meteorite landings are more frequent. Meanwhile, the Mean Shift algorithm, as depicted in its respective scatter plot, has revealed an intricate pattern of numerous small clusters, capturing the nuanced differences in the distribution of meteorites across the globe. This indicates not only a fine granularity in the division of the data but also the presence of multiple dense regions of landings. On the other hand, the DBSCAN clustering visualization accentuates a few large clusters interspersed with outliers, effectively distinguishing densely packed landings from solitary or less clustered ones. Together, these clustering analyses provide a comprehensive view of the geographical dispersion of meteorite landings, with K-Means highlighting the most active zones, Mean Shift detailing the subtleties in distribution, and DBSCAN identifying both dense clusters and individual outliers.

Key Insights and Findings

Upon a comprehensive analysis, it was found that the spatial distribution of meteorite landings is not uniform, with some regions exhibiting higher concentrations of landings. The variability in meteorite mass and class types among the clusters indicates a possible correlation between the falling patterns of meteorites and their physical characteristics. Furthermore, the temporal distribution of findings suggests a marked increase in recorded landites over recent years, which may be attributed to improved detection methods or increased human presence in previously remote locations. The assessment of the fall types revealed that an overwhelming majority of meteorites were found rather than observed during their fall, underscoring the notion that many meteorites likely go unnoticed during their descent. The exploratory data analysis has also uncovered that the mass distribution of meteorites follows a logarithmic scale, indicating a prevalence of smaller mass meteorites with occasional larger outliers, which are important for understanding the distribution and impact of meteorite landings on Earth.

Next Steps for Analysis

On the path forward, a multi-faceted approach would be beneficial. Initially, the development of a time series analysis might reveal trends and patterns over periods that are not immediately apparent in a static clustering model. The examination of external factors, such as atmospheric conditions or planetary positions, could provide insights into the landing patterns and assist in the prediction of future landings. Delving into the composition of meteorites in tandem with their landing coordinates might yield findings related to the geographical distribution of certain types or classes of meteorites. To improve the robustness of the model, employing ensemble clustering techniques that consolidate findings from multiple clustering methods could offer a more accurate representation of the data. Although the current feature set is relatively low-dimensional, exploring PCA might provide insights into the most variance-carrying directions and could potentially reveal more distinct clustering patterns, especially if the analysis is extended by including additional meteorite characteristics in the future. Lastly, implementing a deep learning approach, potentially using autoencoders for dimensionality reduction, could reveal complex, non-linear relationships in the dataset that are not captured by traditional clustering algorithms.

unsupervised_machine_learning_project_çağla

March 19, 2024

[1]:
```
import pandas as pd

# Loading the dataset
data_path = 'meteorite-landings.csv'
df = pd.read_csv(data_path)

# Displaying the first few rows of the dataframe and the basic information about the dataset
df_info = df.info()
df_head = df.head()

df_info, df_head
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45716 entries, 0 to 45715
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         45716 non-null  object
 1   id           45716 non-null  int64
 2   nametype     45716 non-null  object
 3   recclass     45716 non-null  object
 4   mass         45585 non-null  float64
 5   fall         45716 non-null  object
 6   year         45428 non-null  float64
 7   reclat       38401 non-null  float64
 8   reclong      38401 non-null  float64
 9   GeoLocation  38401 non-null  object
dtypes: float64(4), int64(1), object(5)
memory usage: 3.5+ MB
```

[1]:
```
(None,
        name   id nametype     recclass      mass  fall    year    reclat  \
 0    Aachen    1    Valid           L5      21.0  Fell  1880.0  50.77500
 1    Aarhus    2    Valid           H6     720.0  Fell  1951.0  56.18333
 2      Abee    6    Valid          EH4  107000.0  Fell  1952.0  54.21667
 3  Acapulco   10    Valid  Acapulcoite    1914.0  Fell  1976.0  16.88333
 4   Achiras  370    Valid           L6     780.0  Fell  1902.0 -33.16667
```

```
      reclong               GeoLocation
 0    6.08333     (50.775000, 6.083330)
 1   10.23333    (56.183330, 10.233330)
 2 -113.00000  (54.216670, -113.000000)
 3  -99.90000   (16.883330, -99.900000)
 4  -64.95000  (-33.166670, -64.950000)  )
```

[2]:
```
import numpy as np

# Dropping NA values in 'year' and converting 'year' to an integer
df_cleaned = df.dropna(subset=['year']).copy()
df_cleaned['year'] = df_cleaned['year'].astype(int)

# Filtering out years outside the 860-2016 range
df_cleaned = df_cleaned[(df_cleaned['year'] >= 860) & (df_cleaned['year'] <=␣
 ↪2016)]

# Addressing the ON/OE coordinates issue by setting them to NaN
df_cleaned.loc[(df_cleaned['reclat'] == 0) & (df_cleaned['reclong'] == 0),␣
 ↪['reclat', 'reclong']] = np.nan

df_cleaned.dropna(inplace=True)

# Re-checking the info and the head of the cleaned dataframe
df_cleaned_info = df_cleaned.info()
df_cleaned_head = df_cleaned.head()


df_cleaned_info, df_cleaned_head
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31929 entries, 0 to 45715
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         31929 non-null  object
 1   id           31929 non-null  int64
 2   nametype     31929 non-null  object
 3   recclass     31929 non-null  object
 4   mass         31929 non-null  float64
 5   fall         31929 non-null  object
 6   year         31929 non-null  int64
 7   reclat       31929 non-null  float64
 8   reclong      31929 non-null  float64
 9   GeoLocation  31929 non-null  object
dtypes: float64(3), int64(2), object(5)
```

```
memory usage: 2.7+ MB
```

[2]:
```
(None,
        name   id nametype     recclass      mass  fall  year    reclat  \
 0    Aachen    1    Valid           L5      21.0  Fell  1880  50.77500
 1    Aarhus    2    Valid           H6     720.0  Fell  1951  56.18333
 2      Abee    6    Valid          EH4  107000.0  Fell  1952  54.21667
 3  Acapulco   10    Valid  Acapulcoite    1914.0  Fell  1976  16.88333
 4   Achiras  370    Valid           L6     780.0  Fell  1902 -33.16667

      reclong               GeoLocation
 0    6.08333     (50.775000, 6.083330)
 1   10.23333    (56.183330, 10.233330)
 2 -113.00000  (54.216670, -113.000000)
 3  -99.90000   (16.883330, -99.900000)
 4  -64.95000  (-33.166670, -64.950000)  )
```

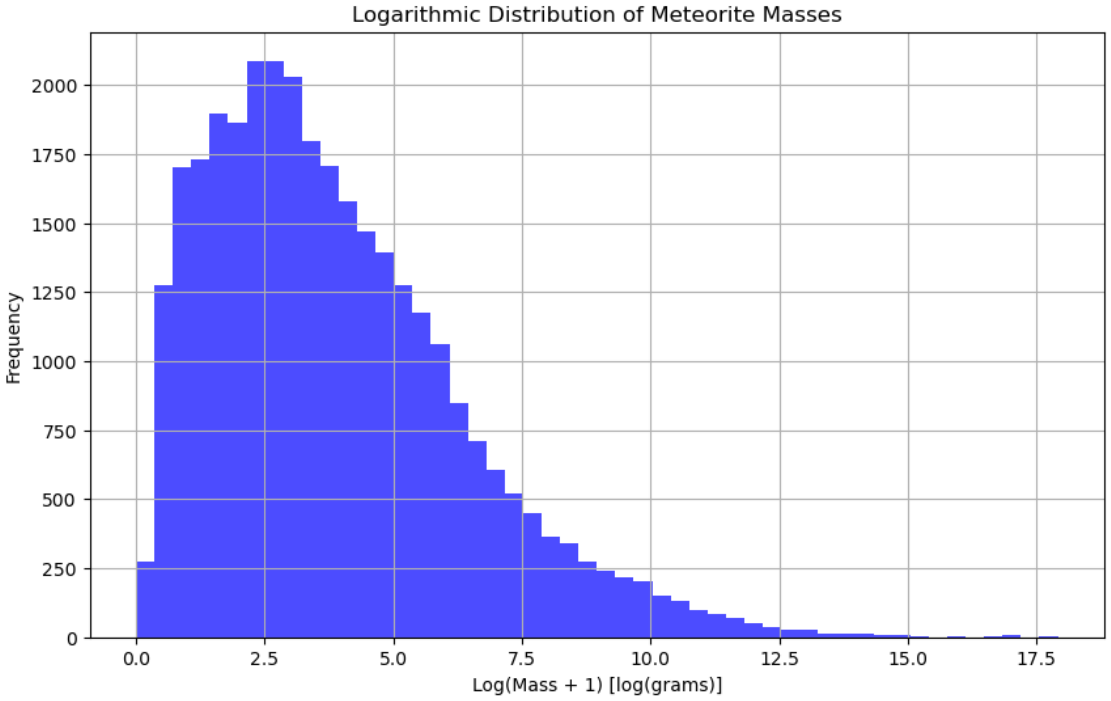
[3]:
```
# Generating a statistical summary of the numerical columns
statistical_summary = df_cleaned.describe()

statistical_summary
```

[3]:

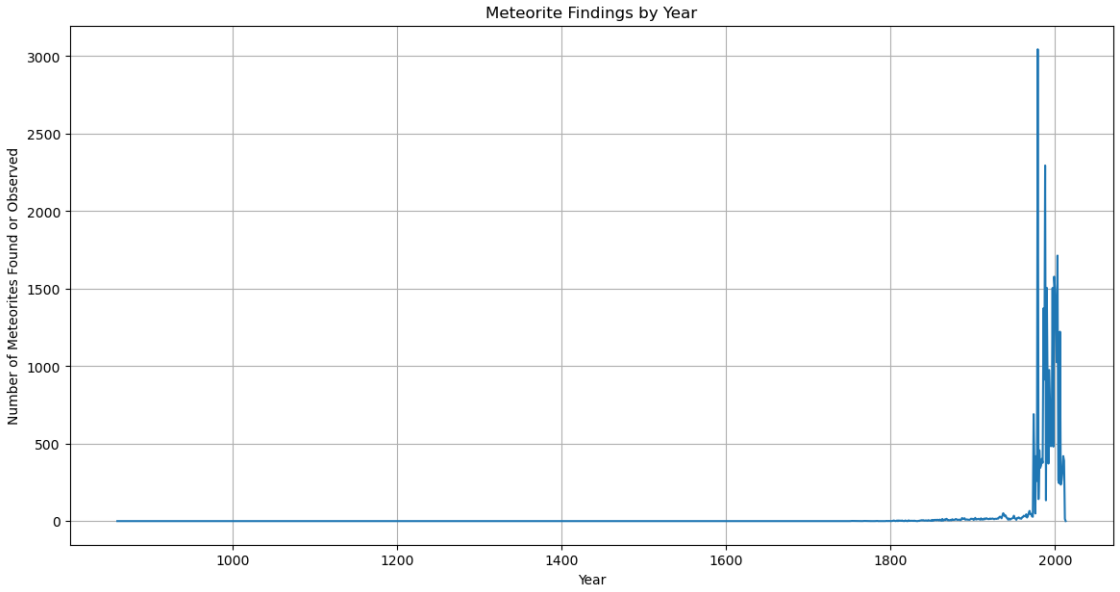
	id	mass	year	reclat	reclong
count	31929.000000	3.192900e+04	31929.000000	31929.000000	31929.000000
mean	20780.386357	1.854289e+04	1986.994362	-47.268054	73.187580
std	14925.593765	6.868495e+05	26.720354	46.719265	83.185722
min	1.000000	0.000000e+00	860.000000	-87.366670	-165.433330
25%	9246.000000	6.500000e+00	1982.000000	-79.683330	26.000000
50%	18622.000000	2.960000e+01	1991.000000	-72.000000	56.822620
75%	27244.000000	2.020000e+02	2000.000000	18.333330	159.393610
max	57455.000000	6.000000e+07	2013.000000	81.166670	178.200000

[4]:
```
import matplotlib.pyplot as plt
import numpy as np

# Visualization of Meteorite Mass Distribution with a log scale due to skewness

# Dropping the entries with missing mass to avoid errors in log scale␣
 ↪visualization
df_mass = df_cleaned.dropna(subset=['mass'])

# Applying a log transformation since the mass distribution is highly skewed
log_masses = np.log(df_mass['mass'] + 1)  # Adding 1 to avoid log(0)

# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.hist(log_masses, bins=50, color='blue', alpha=0.7)
```

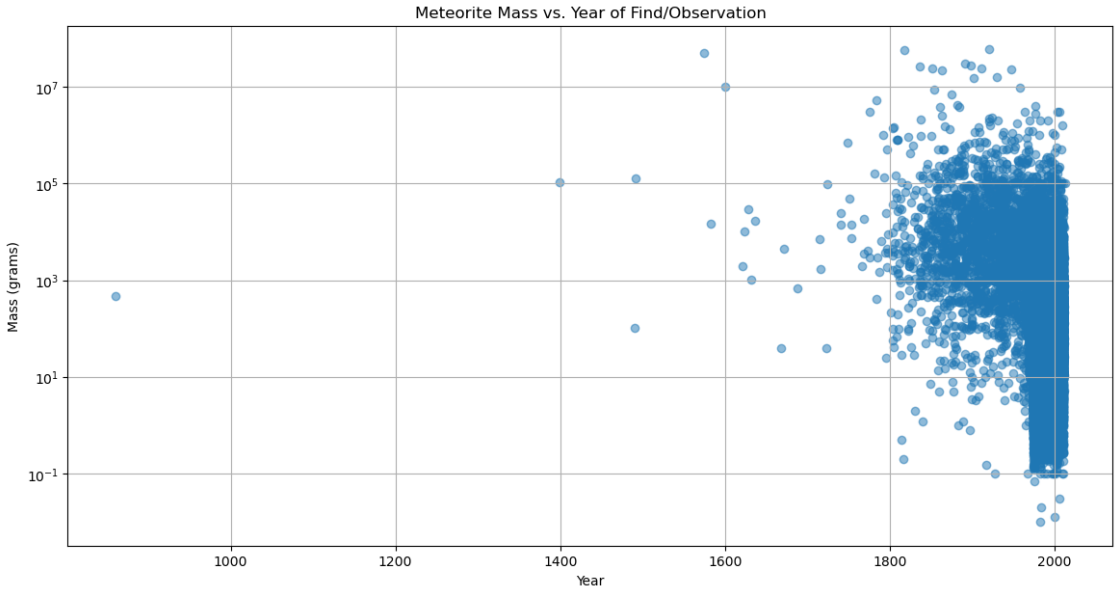
```
plt.title('Logarithmic Distribution of Meteorite Masses')
plt.xlabel('Log(Mass + 1) [log(grams)]')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



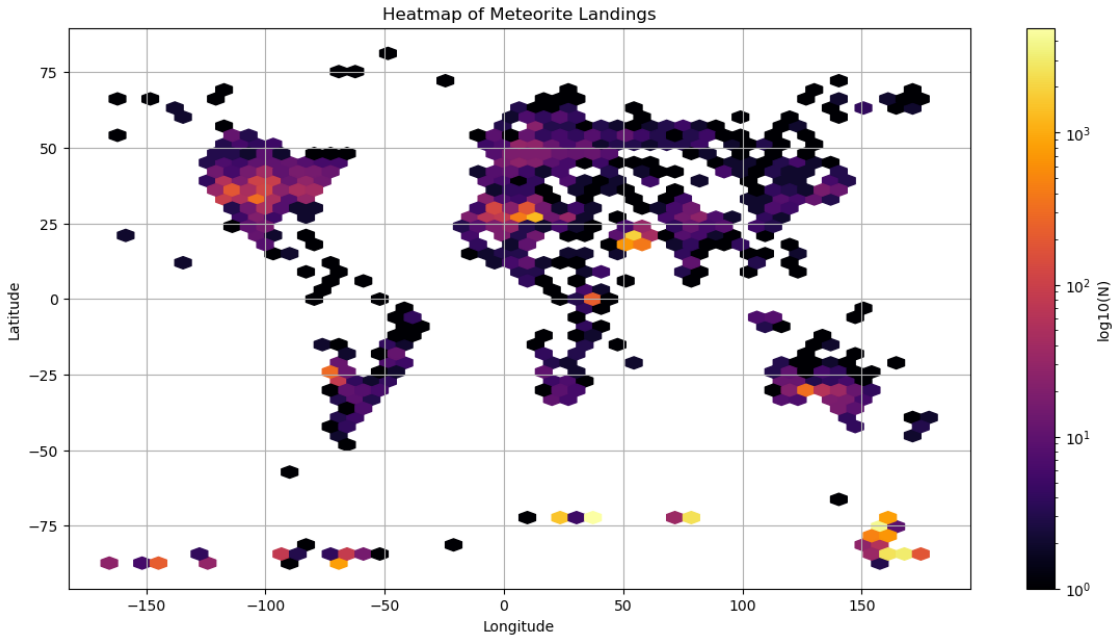
[5]:
```
df_cleaned['year'].value_counts().sort_index().plot(kind='line', figsize=(14,␣
↪7))
plt.title('Meteorite Findings by Year')
plt.xlabel('Year')
plt.ylabel('Number of Meteorites Found or Observed')
plt.grid(True)
plt.show()
```



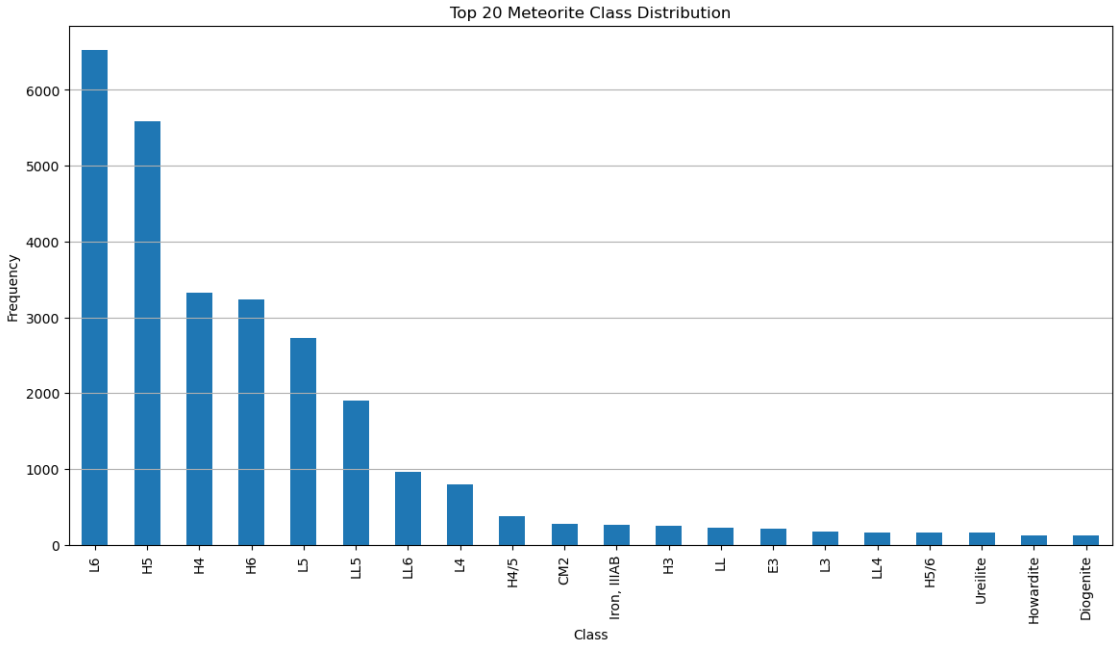
[6]:
```
plt.figure(figsize=(14, 7))
plt.scatter(df_cleaned['year'], df_cleaned['mass'], alpha=0.5)
plt.title('Meteorite Mass vs. Year of Find/Observation')
plt.xlabel('Year')
plt.ylabel('Mass (grams)')
plt.yscale('log')  # Log scale due to large range of masses
plt.grid(True)
plt.show()
```



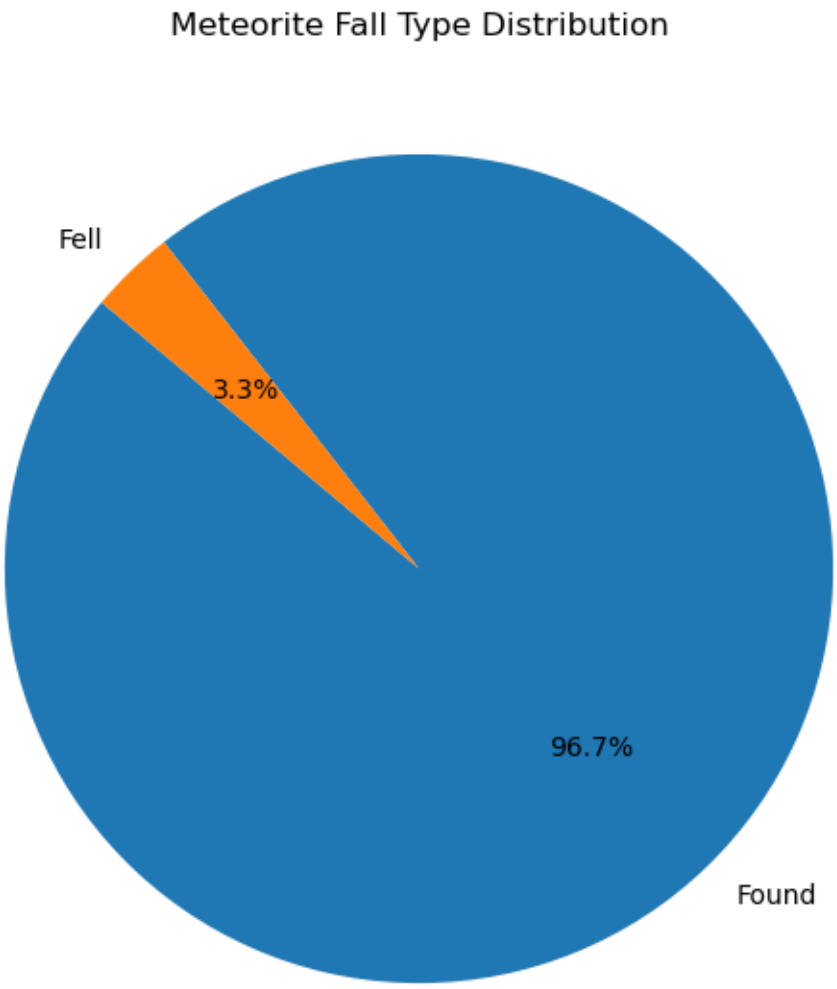
[7]:
```
plt.figure(figsize=(14, 7))
plt.hexbin(df_cleaned['reclong'], df_cleaned['reclat'], gridsize=50,␣
 ↪cmap='inferno', bins='log')
plt.colorbar(label='log10(N)')
plt.title('Heatmap of Meteorite Landings')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



[8]:
```
class_counts = df_cleaned['recclass'].value_counts().head(20)  # Limiting to␣
 ↪top 20 classes for readability
class_counts.plot(kind='bar', figsize=(14, 7))
plt.title('Top 20 Meteorite Class Distribution')
plt.xlabel('Class')
plt.ylabel('Frequency')
plt.grid(axis='y')
plt.show()
```

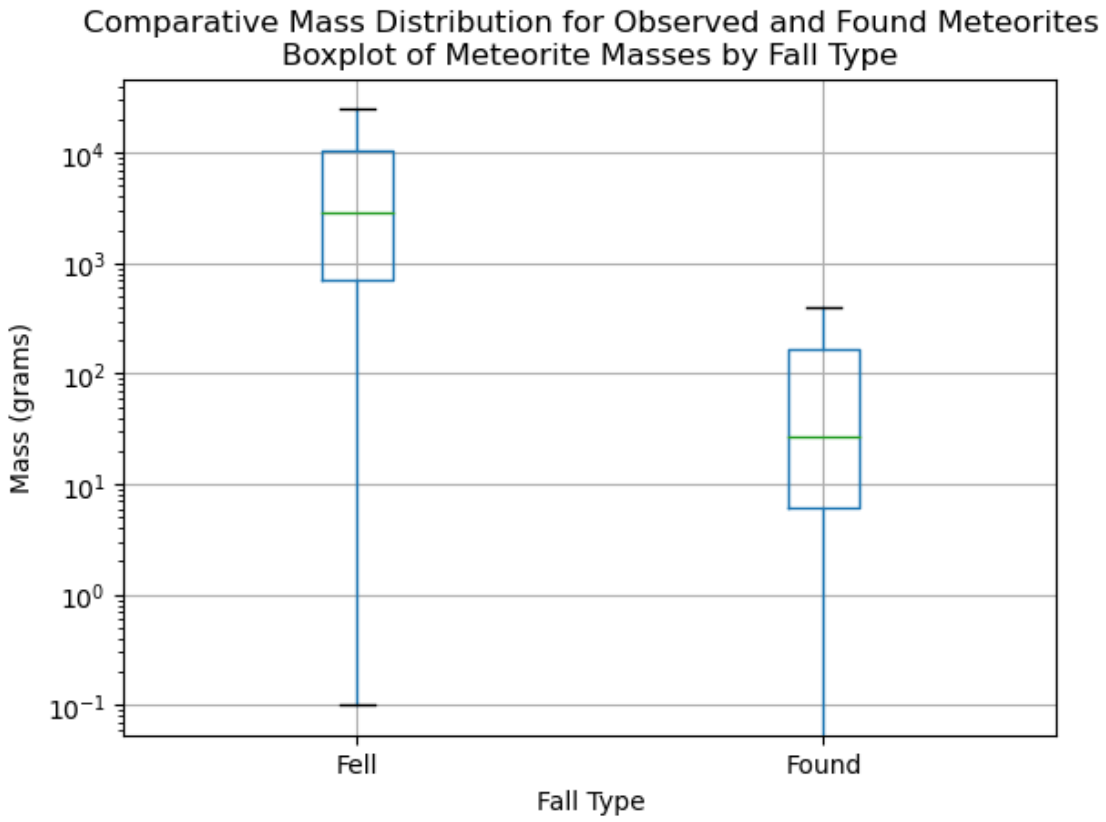


[9]:
```
fall_counts = df_cleaned['fall'].value_counts()
plt.figure(figsize=(7, 7))
plt.pie(fall_counts, labels=fall_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Meteorite Fall Type Distribution')
plt.show()
```



[10]:
```
plt.figure(figsize=(10, 7))
df_cleaned.boxplot(column='mass', by='fall', showfliers=False)  # Excluding␣
 ↪outliers for better visualization
plt.title('Boxplot of Meteorite Masses by Fall Type')
plt.suptitle('Comparative Mass Distribution for Observed and Found Meteorites')
plt.xlabel('Fall Type')
plt.ylabel('Mass (grams)')
plt.yscale('log')  # Log scale due to large range of masses
plt.grid(True)
plt.show()
```

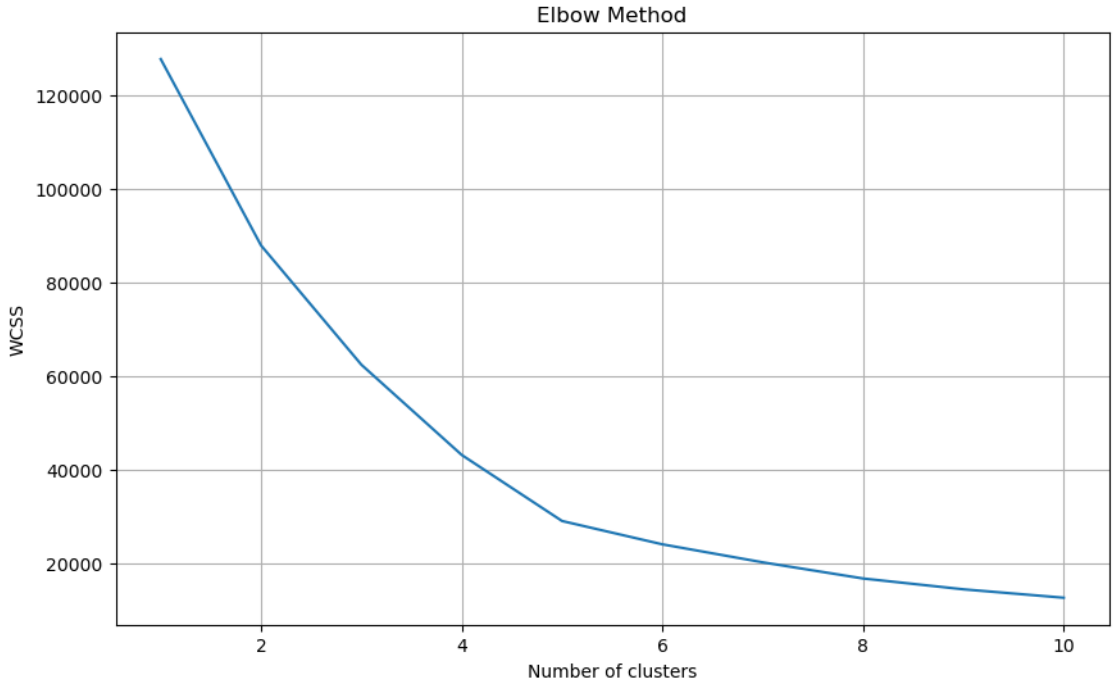
```
<Figure size 1000x700 with 0 Axes>
```



[11]:
```
import warnings
from sklearn.preprocessing import StandardScaler

# Converting all numerical columns to float
numerical_features = df_cleaned[['mass', 'year', 'reclat', 'reclong']].
 ↪astype(float)

# Initializing the StandardScaler
scaler = StandardScaler()

# Suppressing specific sklearn warnings
warnings.filterwarnings(action='ignore', module='sklearn')

# Scaling the features
df_scaled = scaler.fit_transform(numerical_features)
```

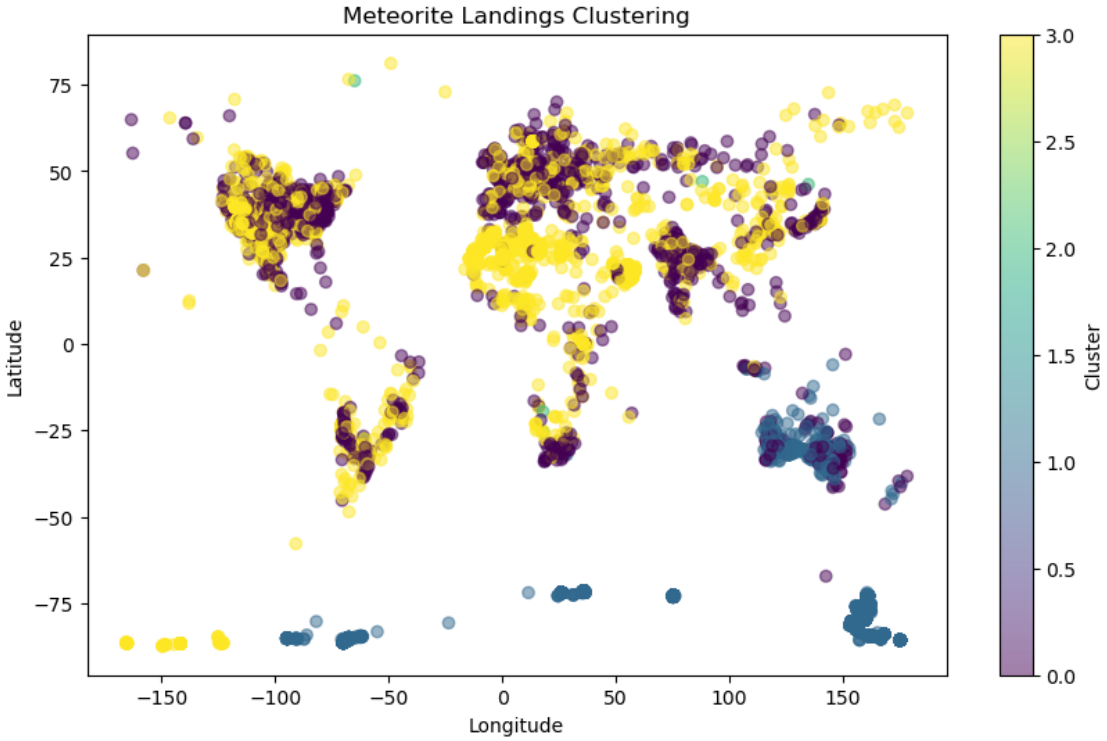
[12]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```



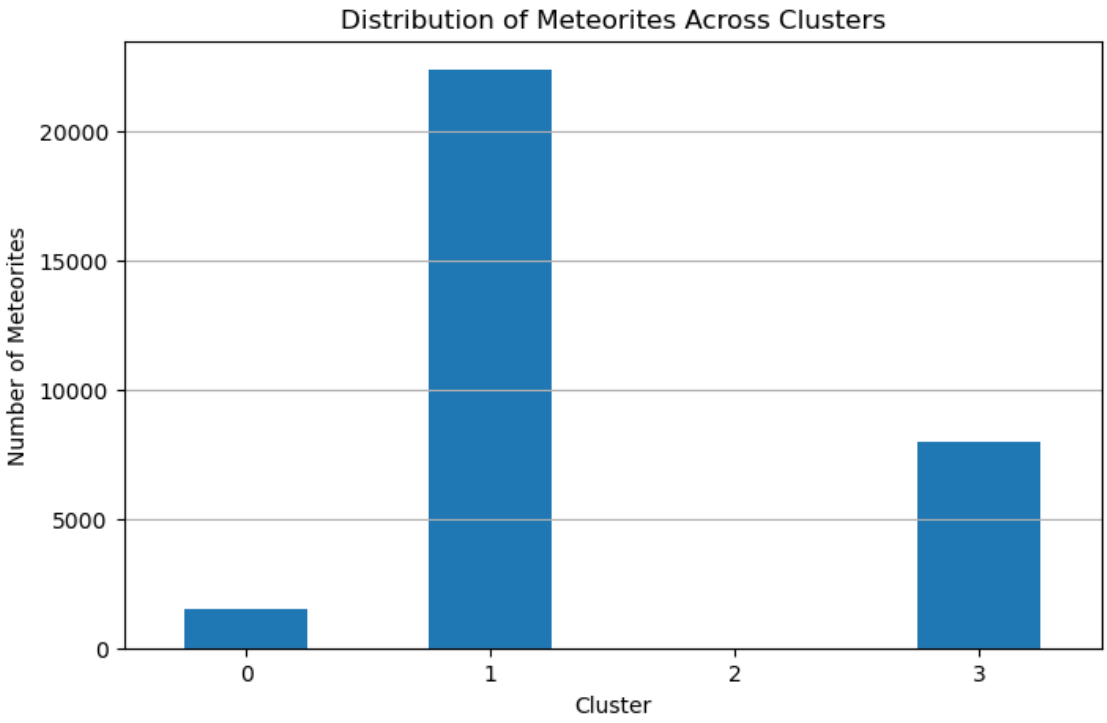
[13]:
```
# k=4 is chosen based on the Elbow Method
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=42)
cluster_labels = kmeans.fit_predict(df_scaled)

# Adding the cluster labels to the cleaned dataframe
df_cleaned['Cluster'] = cluster_labels
```

[14]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(df_cleaned['reclong'], df_cleaned['reclat'],␣
 ↪c=df_cleaned['Cluster'], cmap='viridis', alpha=0.5)
plt.title('Meteorite Landings Clustering')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.colorbar(label='Cluster')
plt.show()
```



[15]:
```
cluster_distribution = df_cleaned['Cluster'].value_counts()
print(cluster_distribution)

# Visualizing this distribution
plt.figure(figsize=(8, 5))
cluster_distribution.sort_index().plot(kind='bar')
plt.title('Distribution of Meteorites Across Clusters')
plt.xlabel('Cluster')
plt.ylabel('Number of Meteorites')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```

```
1    22385
3     8010
0     1524
2       10
Name: Cluster, dtype: int64
```



[16]:
```python
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(df_scaled, cluster_labels)
print(f'Silhouette Score: {silhouette_avg}')
```

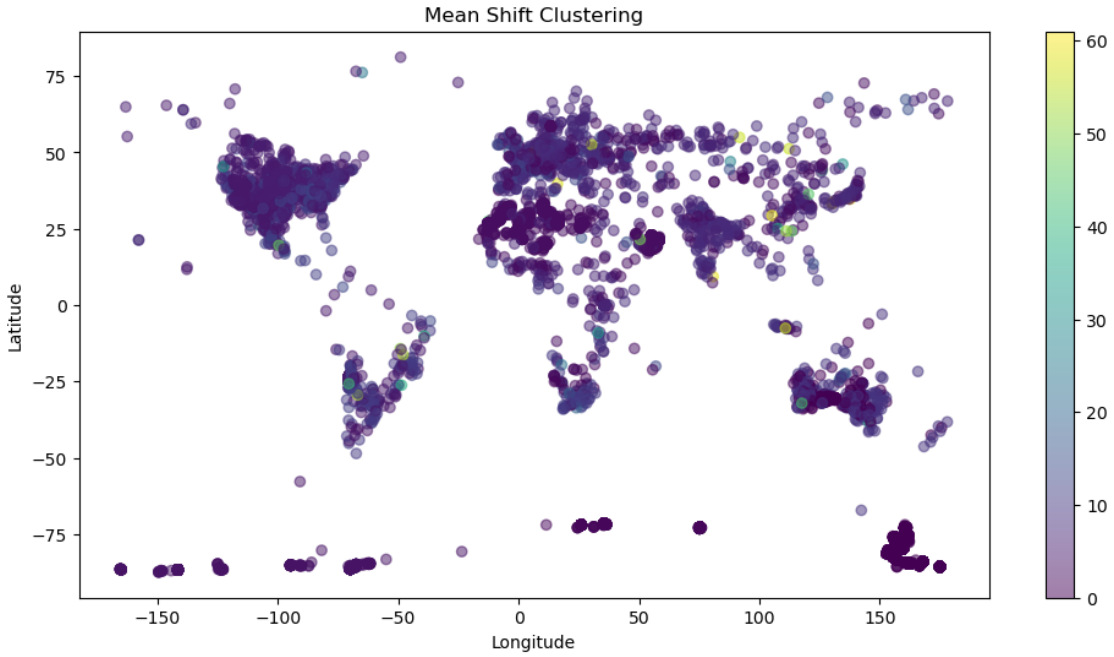
```
Silhouette Score: 0.564418075924552
```

[17]:
```python
from sklearn.cluster import MeanShift, estimate_bandwidth
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score


# Mean Shift Clustering
bandwidth = estimate_bandwidth(df_scaled, quantile=0.2, n_samples=500)
mean_shift = MeanShift(bandwidth=bandwidth, bin_seeding=True)
mean_shift_labels = mean_shift.fit_predict(df_scaled)

# Silhouette Score for Mean Shift
```

```
silhouette_avg_ms = silhouette_score(df_scaled, mean_shift_labels)
print(f'Silhouette Score for Mean Shift: {silhouette_avg_ms}')

# DBSCAN Clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(df_scaled)

# Silhouette Score for DBSCAN
silhouette_avg_dbscan = silhouette_score(df_scaled, dbscan_labels)
print(f'Silhouette Score for DBSCAN: {silhouette_avg_dbscan}')
```

```
Silhouette Score for Mean Shift: 0.6467091181744343
Silhouette Score for DBSCAN: 0.581549933674657
```

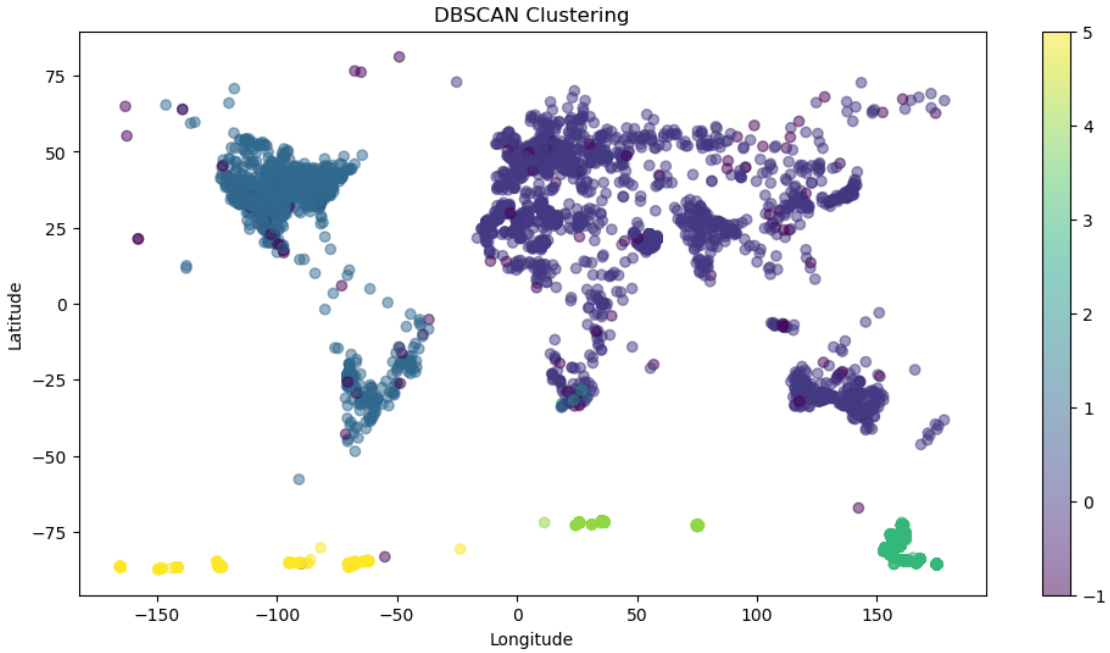
[22]:
```
import matplotlib.pyplot as plt

# Visualizing the clusters for Mean Shift
plt.figure(figsize=(12, 6))

plt.scatter(df_cleaned['reclong'], df_cleaned['reclat'], c=mean_shift_labels,␣
↪cmap='viridis', alpha=0.5)
plt.colorbar()
plt.title('Mean Shift Clustering')
plt.xlabel('Longitude')
plt.ylabel('Latitude')


plt.show()

# Getting the count of meteorites in each cluster for Mean Shift
unique, counts = np.unique(mean_shift_labels, return_counts=True)
mean_shift_cluster_counts = dict(zip(unique, counts))
print(f"Mean Shift - Number of meteorites in each cluster:␣
↪{mean_shift_cluster_counts}")
```




```
Mean Shift - Number of meteorites in each cluster: {0: 12375, 1: 8915, 2: 5662,
3: 1292, 4: 978, 5: 887, 6: 490, 7: 226, 8: 342, 9: 183, 10: 163, 11: 207, 12:
14, 13: 106, 14: 8, 15: 9, 16: 3, 17: 9, 18: 2, 19: 4, 20: 2, 21: 2, 22: 2, 23:
1, 24: 1, 25: 1, 26: 1, 27: 1, 28: 1, 29: 1, 30: 1, 31: 1, 32: 1, 33: 1, 34: 1,
35: 1, 36: 1, 37: 1, 38: 1, 39: 1, 40: 1, 41: 1, 42: 1, 43: 1, 44: 1, 45: 1, 46:
1, 47: 1, 48: 1, 49: 1, 50: 1, 51: 1, 52: 2, 53: 1, 54: 2, 55: 2, 56: 5, 57: 1,
58: 1, 59: 3, 60: 1, 61: 1}
```

[24]:
```
import matplotlib.pyplot as plt


# Visualizing the clusters for DBSCAN
plt.figure(figsize=(12, 6))

plt.scatter(df_cleaned['reclong'], df_cleaned['reclat'], c=dbscan_labels,␣
 ↪cmap='viridis', alpha=0.5)
plt.colorbar()
plt.title('DBSCAN Clustering')
plt.xlabel('Longitude')
plt.ylabel('Latitude')


plt.show()

# Getting the count of meteorites in each cluster for DBSCAN
unique, counts = np.unique(dbscan_labels, return_counts=True)
```

```
dbscan_cluster_counts = dict(zip(unique, counts))
print(f"DBSCAN - Number of meteorites in each cluster: {dbscan_cluster_counts}")
```




```
DBSCAN - Number of meteorites in each cluster: {-1: 132, 0: 7390, 1: 2304, 2: 9,
3: 11918, 4: 8896, 5: 1280}
```