In [1]:
```
import os
current_directory = os.getcwd()
print(current_directory)
```

```
C:\Users\kevin4tx\Saved Games\ICT
```

In [6]:
```
# df = pd.get_dummies(df, columns=['recclass'], prefix='recclass')

from sklearn.preprocessing import LabelEncoder

# Load your meteorite dataset
# df = pd.read_csv('your_dataset.csv')

# Use label encoding for the 'recclass' column
le = LabelEncoder()
df['recclass_encoded'] = le.fit_transform(df['recclass'])

# Printing the first two rows
print(df.head(6))
```

```
       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
5  Adhi Kot  379    Valid          EH4    4239.0  Fell  1919.0  32.10000

     reclong                   geo  recclass_encoded
0    6.08333     (50.775, 6.08333)               254
1   10.23333  (56.18333, 10.23333)               144
2 -113.00000    (54.21667, -113.0)                63
3  -99.90000     (16.88333, -99.9)                 0
4  -64.95000   (-33.16667, -64.95)               259
5   71.80000          (32.1, 71.8)                63
```

In [8]:
```
import pandas as pd
from sklearn.ensemble import IsolationForest

# Load your meteorite dataset (replace 'your_dataset.csv' with the actual file path)
# df = pd.read_csv('your_dataset.csv')

# Select the features for anomaly detection (e.g., 'mass' and 'reclat')
X = df[['mass', 'reclong']]

# Create an Isolation Forest model
model = IsolationForest(contamination=0.05)  # Adjust the contamination parameter as r

# Fit the model to the data
model.fit(X)

# Predict anomalies (1 for inliers, -1 for outliers)
anomalies1 = model.predict(X)

# Add the anomaly scores to the original DataFrame
df['anomaly1'] = anomalies1

# Filter the anomalies
anomalous_records1 = df[df['anomaly1'] == -1]
```

```
# Display the anomalous records
print(anomalous_records1)
```

```
             name     id nametype           recclass       mass   fall  \
2            Abee      6    Valid                EH4   107000.0   Fell
7            Agen    392    Valid                 H5    30000.0   Fell
11           AÃ¯r    424    Valid                 L6    24000.0   Fell
15        Akyumak    433    Valid          Iron, IVA    50000.0   Fell
26     Alfianello    466    Valid                 L6   228000.0   Fell
...           ...    ...      ...                ...        ...    ...
20945     Yandama  30347    Valid                 L6     5800.0  Found
20950      Yudoma  30376    Valid          Iron, IVA     7600.0  Found
20953      Zagora  30387    Valid      Iron, IAB-ung    50000.0  Found
20956     Zerkaly  31354    Valid                 H5    16000.0  Found
20957    Zhaoping  54609    Valid  Iron, IAB complex  2000000.0  Found

         year    reclat    reclong                   geo  recclass_encoded  \
2      1952.0  54.21667 -113.00000     (54.21667, -113.0)                63
7      1814.0  44.21667    0.61667    (44.21667, 0.61667)               137
11     1925.0  19.08333    8.38333    (19.08333, 8.38333)               259
15     1981.0  39.91667   42.81667   (39.91667, 42.81667)               182
26     1883.0  45.26667   10.15000      (45.26667, 10.15)               259
...       ...       ...        ...                   ...               ...
20945  1914.0 -29.75000  141.03333    (-29.75, 141.03333)               259
20950  1946.0  60.00000  140.00000          (60.0, 140.0)               182
20953  1987.0  30.36667   -5.85000      (30.36667, -5.85)               163
20956  1956.0  52.13333   81.96667   (52.13333, 81.96667)               137
20957  1983.0  24.23333  111.18333  (24.23333, 111.18333)               155

       anomaly  anomaly1
2           -1        -1
7           -1        -1
11          -1        -1
15          -1        -1
26          -1        -1
...        ...       ...
20945        1        -1
20950       -1        -1
20953       -1        -1
20956       -1        -1
20957       -1        -1

[1048 rows x 13 columns]
```

In [9]:

```
# Assuming you have a DataFrame 'df'

# Count the rows where both 'anomaly' and 'anomaly1' columns have the value -1
count_both = len(df[(df['anomaly'] == -1) & (df['anomaly1'] == -1)])

print(f"Number of rows with -1 in both 'anomaly' and 'anomaly1': {count_both}")
```

```
Number of rows with -1 in both 'anomaly' and 'anomaly1': 852
```

In [10]:

```
from sklearn.cluster import KMeans

# Select the features for clustering
features = df[['mass', 'reclat', 'reclong']]

# Remove rows with 'None Type' (NaN) values in any of the selected columns
```

```
features_cleaned = features.dropna()

# Now, 'features_cleaned' contains the rows with valid values in the selected columns.


# Choose the number of clusters (K)
k = 4  # You can change this based on your analysis

# Create and fit the K-Means model
kmeans = KMeans(n_clusters=k, random_state=42)
clusters = kmeans.fit_predict(features_cleaned)

# Add the cluster labels to your DataFrame
df['cluster'] = clusters
```

```
C:\Users\ICT\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarni
ng: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 17
     15 # Create and fit the K-Means model
     16 kmeans = KMeans(n_clusters=k, random_state=42)
---> 17 clusters = kmeans.fit_predict(features_cleaned)
     19 # Add the cluster labels to your DataFrame
     20 df['cluster'] = clusters

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1068, in _BaseKMeans.fit_predict(self, X, y, sample_weight)
   1045 def fit_predict(self, X, y=None, sample_weight=None):
   1046     """Compute cluster centers and predict cluster index for each sample.
   1047
   1048     Convenience method; equivalent to calling fit(X) followed by
   (...)
   1066         Index of the cluster each sample belongs to.
   1067     """
-> 1068     return self.fit(X, sample_weight=sample_weight).labels_

File ~\anaconda3\Lib\site-packages\sklearn\base.py:1151, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1526, in KMeans.fit(self, X, y, sample_weight)
   1523     print("Initialization complete")
   1525 # run a k-means once
-> 1526 labels, inertia, centers, n_iter_ = kmeans_single(
   1527     X,
   1528     sample_weight,
   1529     centers_init,
   1530     max_iter=self.max_iter,
   1531     verbose=self.verbose,
   1532     tol=self._tol,
   1533     n_threads=self._n_threads,
   1534 )
   1536 # determine if these results are the best so far
   1537 # we chose a new run if it has a better inertia and the clustering is
   1538 # different from the best so far (it's possible that the inertia is
   1539 # slightly better even if the clustering is the same with potentially
   1540 # permuted labels, due to rounding errors)
   1541 if best_inertia is None or (
   1542     inertia < best_inertia
   1543     and not _is_same_clustering(labels, best_labels, self.n_clusters)
   1544 ):

File ~\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:688, in _kmeans_single_lloyd(X, sample_weight, centers_init, max_iter, verbose, tol, n_threads)
    684 strict_convergence = False
    686 # Threadpoolctl context to limit the number of threads in second level of
    687 # nested parallelism (i.e. BLAS) to avoid oversubscription.
--> 688 with threadpool_limits(limits=1, user_api="blas"):
    689     for i in range(max_iter):
```

```
    690         lloyd_iter(
    691             X,
    692             sample_weight,
   (...)
    698             n_threads,
    699         )

File ~\anaconda3\Lib\site-packages\sklearn\utils\fixes.py:72, in threadpool_limits(limits, user_api)
     70     return controller.limit(limits=limits, user_api=user_api)
     71 else:
---> 72     return threadpoolctl.threadpool_limits(limits=limits, user_api=user_api)

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:171, in threadpool_limits.__init__(self, limits, user_api)
    167 def __init__(self, limits=None, user_api=None):
    168     self._limits, self._user_api, self._prefixes = \
    169         self._check_params(limits, user_api)
--> 171     self._original_info = self._set_threadpool_limits()

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:268, in threadpool_limits._set_threadpool_limits(self)
    265 if self._limits is None:
    266     return None
--> 268 modules = _ThreadpoolInfo(prefixes=self._prefixes,
    269                            user_api=self._user_api)
    270 for module in modules:
    271     # self._limits is a dict {key: num_threads} where key is either
    272     # a prefix or a user_api. If a module matches both, the limit
    273     # corresponding to the prefix is chosed.
    274     if module.prefix in self._limits:

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:340, in _ThreadpoolInfo.__init__(self, user_api, prefixes, modules)
    337     self.user_api = [] if user_api is None else user_api
    339     self.modules = []
--> 340     self._load_modules()
    341     self._warn_if_incompatible_openmp()
    342 else:

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:373, in _ThreadpoolInfo._load_modules(self)
    371     self._find_modules_with_dyld()
    372 elif sys.platform == "win32":
--> 373     self._find_modules_with_enum_process_module_ex()
    374 else:
    375     self._find_modules_with_dl_iterate_phdr()

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:485, in _ThreadpoolInfo._find_modules_with_enum_process_module_ex(self)
    482         filepath = buf.value
    484         # Store the module if it is supported and selected
--> 485         self._make_module_from_path(filepath)
    486 finally:
    487     kernel_32.CloseHandle(h_process)

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:515, in _ThreadpoolInfo._make_module_from_path(self, filepath)
    513 if prefix in self.prefixes or user_api in self.user_api:
    514     module_class = globals()[module_class]
```

```
--> 515     module = module_class(filepath, prefix, user_api, internal_api)
    516     self.modules.append(module)

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:606, in _Module.__init__(self, fi
lepath, prefix, user_api, internal_api)
    604 self.internal_api = internal_api
    605 self._dynlib = ctypes.CDLL(filepath, mode=_RTLD_NOLOAD)
--> 606 self.version = self.get_version()
    607 self.num_threads = self.get_num_threads()
    608 self._get_extra_info()

File ~\anaconda3\Lib\site-packages\threadpoolctl.py:646, in _OpenBLASModule.get_versi
on(self)
    643 get_config = getattr(self._dynlib, "openblas_get_config",
    644                      lambda: None)
    645 get_config.restype = ctypes.c_char_p
--> 646 config = get_config().split()
    647 if config[0] == b"OpenBLAS":
    648     return config[1].decode("utf-8")

AttributeError: 'NoneType' object has no attribute 'split'
```

In [38]:

```python
import pandas as pd

# Load your meteorite dataset
#df = pd.read_csv('your_dataset.csv')

# Identify missing values in the 'mass' column
missing_year = df['year'].isna()

# Calculate the mean mass
mean_year = df['year'].mean()

# Impute missing values with the mean mass
df.loc[missing_year, 'year'] = mean_year
```

In [3]:

```python
# Uploading dataset and checking

import pandas as pd

# Read the csv data file in to dataframe name "df"
df = pd.read_csv('C:/Users/ICT/dataforkdd.csv')

# Printing the first two rows
print(df.head(6))
```

```
       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
5  Adhi Kot  379    Valid          EH4    4239.0  Fell  1919.0  32.10000

     reclong                   geo
0    6.08333     (50.775, 6.08333)
1   10.23333  (56.18333, 10.23333)
2 -113.00000    (54.21667, -113.0)
3  -99.90000     (16.88333, -99.9)
4  -64.95000   (-33.16667, -64.95)
5   71.80000          (32.1, 71.8)
```

In [5]:
```python
import pandas as pd

# Assuming you have a DataFrame 'df' with a 'year' column

# Replace NaN values in the 'year' column with 0
data['reclong'].fillna(0, inplace=True)
```

In [6]:
```python
import pandas as pd
from sklearn.neighbors import NearestNeighbors
import numpy as np

# Load the dataset
#data = pd.read_csv('C:/Users/kevin4tx/Saved Games/ICT/dataforkdd.csv')

# Select the features to be used for anomaly detection
features = ['mass', 'year', 'reclat', 'reclong']

# Create a DataFrame with the selected features
X = data[features]

# Initialize and fit the K-nearest neighbors model
n_neighbors = 5  # You can adjust the number of neighbors as needed
knn = NearestNeighbors(n_neighbors=n_neighbors)
knn.fit(X)

# Calculate the distance to the kth nearest neighbor for each data point
distances, _ = knn.kneighbors(X)

# Calculate the anomaly score as the mean distance to k-nearest neighbors
anomaly_scores = np.mean(distances, axis=1)

# Add the anomaly scores to the original DataFrame
data['anomaly_score'] = anomaly_scores

# Define a threshold for anomaly detection (you can adjust this threshold)
anomaly_threshold = 2.0

# Identify anomalies based on the threshold
anomalies = data[data['anomaly_score'] > anomaly_threshold]

# Display the anomalies
print("Anomalies:")
print(anomalies)
```

```
# You can save the anomalies to a CSV file if needed
anomalies.to_csv('anomalies.csv', index=False)
```

```
Anomalies:
             name     id nametype              recclass       mass   fall  \
0          Aachen      1    Valid                    L5       21.0   Fell
1          Aarhus      2    Valid                    H6      720.0   Fell
2            Abee      6    Valid                   EH4   107000.0   Fell
3        Acapulco     10    Valid           Acapulcoite     1914.0   Fell
4         Achiras    370    Valid                    L6      780.0   Fell
...           ...    ...      ...                   ...        ...    ...
20955       Zerga  30402    Valid                   LL6       76.0  Found
20956     Zerkaly  31354    Valid                    H5    16000.0  Found
20957    Zhaoping  54609    Valid     Iron, IAB complex  2000000.0  Found
20958      Zinder  30409    Valid  Pallasite, ungrouped       46.0  Found
20959  Zulu Queen  30414    Valid                 L3.7      200.0  Found

         year    reclat    reclong                    geo  anomaly_score
0      1880.0  50.77500    6.08333       (50.775, 6.08333)      13.433988
1      1951.0  56.18333   10.23333    (56.18333, 10.23333)      27.590704
2      1952.0  54.21667 -113.00000      (54.21667, -113.0)     846.103219
3      1976.0  16.88333  -99.90000       (16.88333, -99.9)      25.976231
4      1902.0 -33.16667  -64.95000     (-33.16667, -64.95)      52.517823
...       ...       ...        ...                     ...            ...
20955  1973.0  20.25000  -12.68333      (20.25, -12.68333)      19.184315
20956  1956.0  52.13333   81.96667    (52.13333, 81.96667)      58.565190
20957  1983.0  24.23333  111.18333   (24.23333, 111.18333)   20091.414385
20958  1999.0  13.78333    8.96667     (13.78333, 8.96667)      11.600048
20959  1976.0  33.98333 -115.68333  (33.98333, -115.68333)      19.183654

[7520 rows x 11 columns]
```

In [23]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Load your meteorite dataset (replace 'your_dataset.csv' with the actual file path)
# df = pd.read_csv('your_dataset.csv')

# Select the features (independent variables) and the target (dependent variable)
X = df[['recclass_encoded', 'reclat', 'reclong']]  # Features
y = df['mass']  # Target variable

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the R-squared (R2) score to measure accuracy
r2 = r2_score(y_test, y_pred)

print(f"R-squared (R2) Score: {r2:.2f}")
```

```
R-squared (R2) Score: 0.00
```