# Занятие 5. Анализ данных. Подготовка и предобработка данных

Данные не всегда поступают в форме, готовой для анализа. Например, они могут иметь неправильный формат, быть ошибочными, а то и вовсе отсутствовать. Опыт показывает, что специалисты по data science тратят до 75% своего времени на подготовку данных перед началом анализа. Подготовка данных называется *первичной обработкой*, два важнейших шага которой заключаются в *очистке данных* и последующем *преобразовании данных* в форматы, оптимальные для ваших систем баз данных и аналитических программ. Несколько типичных примеров очистки данных:

- удаление наблюдений с отсутствующими значениями;
- замена отсутствующих значений подходящими значениями;
- удаление наблюдений с некорректными значениями;
- замена некорректных значений подходящими значениями;
- исключение выбросов (хотя в некоторых случаях их лучше оставить);
- устранение дубликатов (хотя некоторые дубликаты содержат действительную информацию);
- обработка с данными, целостность которых была нарушена
- и так далее.

Вероятно, вы уже думаете, что очистка данных — сложный и хлопотный процесс, в котором легко можно принять ошибочные решения, отрицательно сказывающиеся на ваших результатах. Да, все правильно. Когда мы дойдем до практических примеров data science в следующих главах, вы увидите, что data science в большей степени является *дисциплиной эмпирической*, как медицина, и в меньшей степени — дисциплиной теоретической, как теоретическая физика. В эмпирических дисциплинах выводы делаются на основании наблюдений и практического опыта. Например, многие лекарства, эффективно решающие современные медицинские проблемы, были получены на основе наблюдения за воздействием ранних версий этих лекарств на лабораторных животных, а в конечном итоге и на людей и постепенного уточнения ингредиентов и дозировок. Действия специалистов data science могут изменяться в зависимости от проекта, могут зависеть от качества и природы данных, а также от развивающихся организационных и профессиональных стандартов.

Примеры типичных преобразований данных:

- удаление необязательных данных и признаков;
- объединение взаимосвязанных признаков;
- формирование выборок данных для получения репрезентативного подмножества (в практических примерах data science вы увидите, что случайная выборка особенно эффективна в этом отношении);
- стандартизация форматов данных;
- группировка данных

## 1.1 Загрузка данных

In [1]:

```
# подгрузка необходимых модулей для визуализации и работы с данными
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
/home/agat.local/s.bulganin/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/masked.py:60: UserWar
ning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [4]:

```
# загрузка данных
data_all = pd.read_csv("data/mammal_atlas.csv")
data_all.head(4).T
```

Out[4]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **Scientific name** | Capreolus capreolus | Erinaceus europaeus | Halichoerus grypus | Talpa europaea |
| **Common name** | Roe Deer | West European Hedgehog | Grey Seal | European Mole |
| **Species ID (TVK)** | NHMSYS0000080203 | NBNSYS0000005078 | NBNSYS0000005137 | NBNSYS0000005079 |
| **Taxon Rank** | species | species | species | species |
| **Start date** | 31/01/2018 | 20/01/2021 | 06/01/2016 | 25/01/2017 |
| **Start date day** | 31 | 20 | 6 | 25 |
| **Start date month** | 1 | 1 | 1 | 1 |
| **Start date year** | 2018 | 2021 | 2016 | 2017 |
| **Latitude (WGS84)** | 57.4002 | 51.519214 | 52.8 | 53.322473 |
| **Longitude (WGS84)** | -4.2679 | -0.738969 | 1.6 | -1.731987 |
| **Coordinate uncertainty (m)** | 50.0 | 57.0 | 7071.1 | 70.7 |
| **Identification verification status** | Accepted - correct | Accepted - correct | Accepted | Accepted |
| **Basis of record** | HumanObservation | HumanObservation | HumanObservation | HumanObservation |
| **Survey key** | iRecord \| Mammals | iRecord \| App | iRecord \| Mammals | iRecord \| Import |
| **Kingdom** | Animalia | Animalia | Animalia | Animalia |
| **Phylum** | Chordata | Chordata | Chordata | Chordata |
| **Class** | Mammalia | Mammalia | Mammalia | Mammalia |
| **Order** | Artiodactyla | Insectivora | Carnivora | Insectivora |
| **Family** | Cervidae | Erinaceidae | Phocidae | Talpidae |
| **Genus** | Capreolus | Erinaceus | Halichoerus | Talpa |
| **Country** | United Kingdom | United Kingdom | United Kingdom | United Kingdom |
| **State/Province** | Scotland | England | England | England |
| **Vitality** | NaN | NaN | NaN | NaN |

## 1.2 Разведочный анализ и очистка данных

### 1.2.1 Информация о признаках

In [5]:

```
data_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227322 entries, 0 to 227321
Data columns (total 23 columns):
 #   Column                              Non-Null Count   Dtype  
---  ------                              --------------   -----  
 0   Scientific name                     227322 non-null  object 
 1   Common name                         227123 non-null  object 
 2   Species ID (TVK)                    227322 non-null  object 
 3   Taxon Rank                          227322 non-null  object 
 4   Start date                          227322 non-null  object 
 5   Start date day                      227322 non-null  int64  
 6   Start date month                    227322 non-null  int64  
 7   Start date year                     227322 non-null  int64  
 8   Latitude (WGS84)                    227321 non-null  float64
 9   Longitude (WGS84)                   227321 non-null  float64
 10  Coordinate uncertainty (m)          227321 non-null  float64
 11  Identification verification status  227322 non-null  object 
 12  Basis of record                     227322 non-null  object 
 13  Survey key                          227322 non-null  object 
 14  Kingdom                             227322 non-null  object 
 15  Phylum                              227322 non-null  object 
 16  Class                               227322 non-null  object 
 17  Order                               227186 non-null  object 
 18  Family                              225849 non-null  object 
 19  Genus                               225328 non-null  object 
 20  Country                             227092 non-null  object 
 21  State/Province                      227092 non-null  object 
 22  Vitality                            0 non-null       float64
dtypes: float64(4), int64(3), object(16)
memory usage: 39.9+ MB
```

В представленном наборе содержатся данные о млекопитающих, которые обитают на территории Великобритании. Датасет содержит следующие признаки:

1. **Scientific name и Common name** -- научное и обиходное название животного, тип данных object;
2. **Taxon Rank, Kingdom, Phylum, Class, Order, Family, Genus** -- данные о месте млекопитающего в таксономической системе (царство животных, класс, род, семейство и т.д.), тип данных object;
3. **Latitude (WGS84), Longitude (WGS84), Coordinate uncertainty (m), Country, State/Province** -- географические данные, говорящие о месте нахождения животного, т.е. географические широта и долгота, погрешность координат и страна; тип данных у всех, кроме Country и State/Province, float;
4. содержится также некоторая техническая информация: **Species ID (TVK)** -- идентификатор вида, **Basis of Record** -- учреждение, которое ведёт запись/учёт, **Identification verification status** -- статус подтверждения идентификации, а также дата начала записи (**Start Date**)

In [6]:

```
# посмотреть основные статистики численных признаков
data_all.describe().T
```

Out[6]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Start date day** | 227322.0 | 15.539239 | 8.833260 | 1.000000 | 8.000000 | 16.000000 | 23.0000 | 31.000000 |
| **Start date month** | 227322.0 | 6.109365 | 3.017320 | 1.000000 | 4.000000 | 6.000000 | 8.0000 | 12.000000 |
| **Start date year** | 227322.0 | 2016.904202 | 5.187149 | 1905.000000 | 2015.000000 | 2018.000000 | 2020.0000 | 2023.000000 |
| **Latitude (WGS84)** | 227321.0 | 52.931642 | 1.774928 | 49.174050 | 51.517338 | 52.622681 | 53.8654 | 60.829075 |
| **Longitude (WGS84)** | 227321.0 | -1.734658 | 1.586204 | -10.461502 | -2.810713 | -1.753089 | -0.4302 | 1.900000 |
| **Coordinate uncertainty (m)** | 227321.0 | 659.041594 | 2088.721995 | 0.300000 | 7.100000 | 50.000000 | 70.7000 | 70710.700000 |
| **Vitality** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

В статистических характеристиках численных признаков можно заметить признак Vitality, в котором отсутствуют все значения. От него необходимо избавиться.

### 1.2.2 Пропуски в данных

Некорректные и отсутствующие значения в данных могут оказать значительное влияние на анализ данных. Некоторые специалисты data science протестуют против любых попыток вставки «разумных значений». Вместо этого они рекомендуют четко пометить отсутствующие данные и предоставить решение проблемы пакету аналитики данных. Другие советуют действовать более осторожно.

In [7]:

```
all_records = data_all.shape[0]
data_all.isnull().sum()/all_records * 100
```

Out[7]:

```
Scientific name                        0.000000
Common name                            0.087541
Species ID (TVK)                       0.000000
Taxon Rank                             0.000000
Start date                             0.000000
Start date day                         0.000000
Start date month                       0.000000
Start date year                        0.000000
Latitude (WGS84)                       0.000440
Longitude (WGS84)                      0.000440
Coordinate uncertainty (m)             0.000440
Identification verification status     0.000000
Basis of record                        0.000000
Survey key                             0.000000
Kingdom                                0.000000
Phylum                                 0.000000
Class                                  0.000000
Order                                  0.059827
Family                                 0.647980
Genus                                  0.877170
Country                                0.101178
State/Province                         0.101178
Vitality                             100.000000
dtype: float64
```

In [8]:

```
irretrievable_cols = list()
for column in data_all.columns:
    if data_all[column].isnull().sum() / all_records * 100 > 60:
        irretrievable_cols.append(column)
print(f"Невосстановимые признаки, у которых пропусков более 60%: {irretrievable_cols}")
data_all.drop(columns=irretrievable_cols, inplace=True)
```

```
Невосстановимые признаки, у которых пропусков более 60%: ['Vitality']
```

In [9]:

```
print(f"Всего записей: {all_records}")
```

```
Всего записей: 227322
```

Выше приведены в процентном отношении данные о наличии пропусков. В данных есть пропуски, но менее 1%. Был отобран признак с полностью отсутствующими значениями и удалён из датасета. С остальными признаками можно поступить следующими способами:

1. Избавиться от записей с пропусками в датасете;
2. Избавиться от всех атрибутов, которые содержат пропуски;
3. Установить недостающие значения в некоторую величину (медиану, среднее, моду и т.д.)

При подстановке “разумных значений” на место отсутствующих или некорректных данных необходимо действовать очень осторожно. “Подставлять” значения, повышающие статистическую значимость или обеспечивающие более “разумные” или “лучшие” результаты, недопустимо. “Подстановка” данных не должна превратиться в “подтасовку” данных. Первое, чему должен научиться исследователь, — не исключать и не изменять значения, противоречащие гипотезам. “Подстановка” разумных значений не означает, что исследователь может изменять данные, чтобы получить нужный результат.

В данной ситуации подойдёт удаление отдельных записей с пропусками, поскольку пропусков менее 1%.

In [10]:

```
data_all.dropna(axis=0, inplace=True)
print(f"Было записей: {all_records}\nСтало записей: {data_all.shape[0]}")
```

```
Было записей: 227322
Стало записей: 224934
```

In [11]:

```
data_all.isnull().sum()/data_all.shape[0] * 100
```

Out[11]:

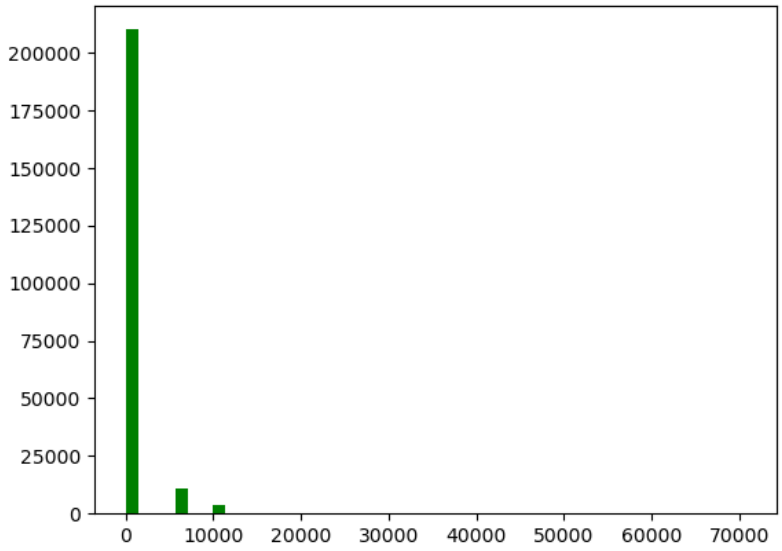
```
Scientific name                      0.0
Common name                          0.0
Species ID (TVK)                     0.0
Taxon Rank                           0.0
Start date                           0.0
Start date day                       0.0
Start date month                     0.0
Start date year                      0.0
Latitude (WGS84)                     0.0
Longitude (WGS84)                    0.0
Coordinate uncertainty (m)           0.0
Identification verification status   0.0
Basis of record                      0.0
Survey key                           0.0
Kingdom                              0.0
Phylum                               0.0
Class                                0.0
Order                                0.0
Family                               0.0
Genus                                0.0
Country                              0.0
State/Province                       0.0
dtype: float64
```

### 1.2.3 Поиск аномалий и выбросов в данных

Ещё при просмотре основных статистик датасета можно было заметить неадекватное распределение признака Coordinate uncertainty (m), у которого наблюдается завышенное значение: 75-процентиль показывает 70.7 метров, а максимальное значение 70710.7 метров (70км!). Исследуем этот признак детальнее.

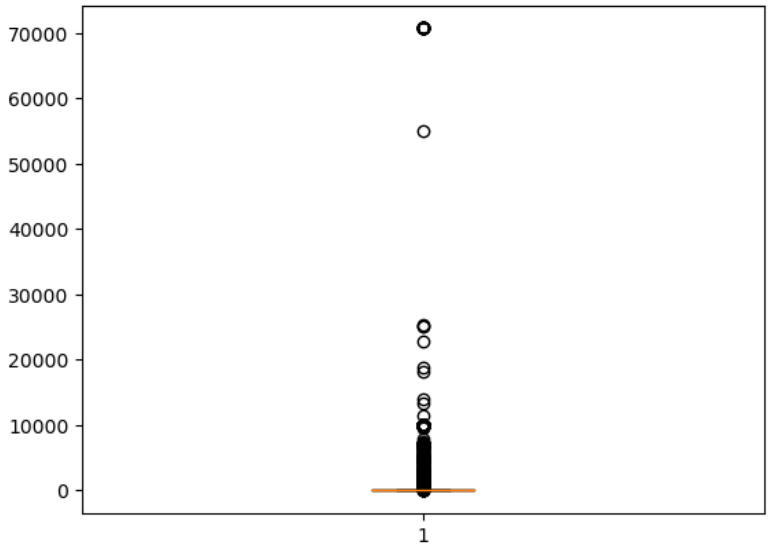
In [12]:

```
data_all['Coordinate uncertainty (m)'].hist(bins=50,color='green');
plt.grid(False)
```




In [13]:

```
plt.boxplot(data_all['Coordinate uncertainty (m)']);
```




Признак сильно зашумлён, нужно удалить записи, в которых содержится слишком высокая погрешность измерения координат.

In [14]:

```
print(f"Количество записей, в которых погрешность более 1км: {len(data_all[data_all['Coordinate uncertainty (m)'] > 1_
print(f"Количество записей, в которых погрешность более 2км: {len(data_all[data_all['Coordinate uncertainty (m)'] > 2_
```

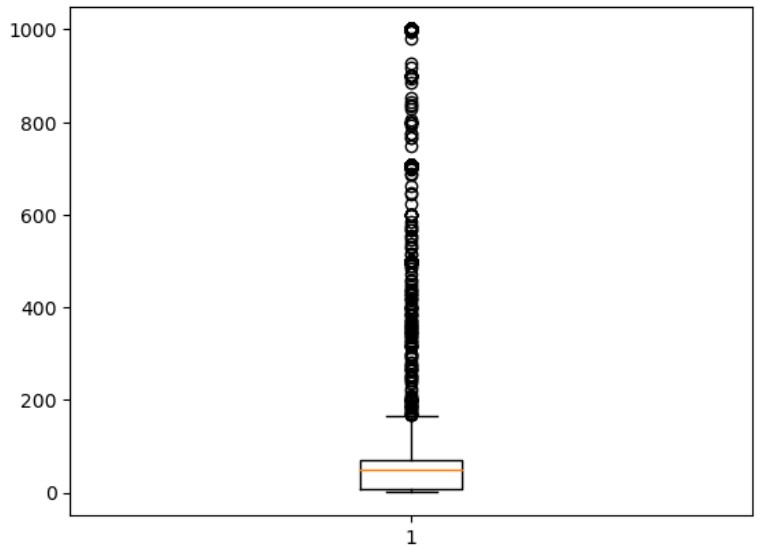
```
Количество записей, в которых погрешность более 1км: 16586
Количество записей, в которых погрешность более 2км: 14928
```

In [15]:

```
data_all.drop(index=data_all[data_all['Coordinate uncertainty (m)'] > 1_000].index, inplace=True)
```

In [16]:

```
plt.boxplot(data_all['Coordinate uncertainty (m)']);
```




In [17]:

```
for i in range(200, 1050, 50):
    print(f"Количество записей, в которых погрешность более {i}м: {len(data_all[data_all['Coordinate uncertainty (m)']
```

```
Количество записей, в которых погрешность более 200м: 33119
Количество записей, в которых погрешность более 250м: 33109
Количество записей, в которых погрешность более 300м: 33095
Количество записей, в которых погрешность более 350м: 33077
Количество записей, в которых погрешность более 400м: 33055
Количество записей, в которых погрешность более 450м: 33044
Количество записей, в которых погрешность более 500м: 32534
Количество записей, в которых погрешность более 550м: 32527
Количество записей, в которых погрешность более 600м: 32512
Количество записей, в которых погрешность более 650м: 32509
Количество записей, в которых погрешность более 700м: 32495
Количество записей, в которых погрешность более 750м: 720
Количество записей, в которых погрешность более 800м: 709
Количество записей, в которых погрешность более 850м: 704
Количество записей, в которых погрешность более 900м: 690
Количество записей, в которых погрешность более 950м: 687
Количество записей, в которых погрешность более 1000м: 0
```

In [18]:

```
data_all.drop(index=data_all[data_all['Coordinate uncertainty (m)'] > 750].index, inplace=True)
```

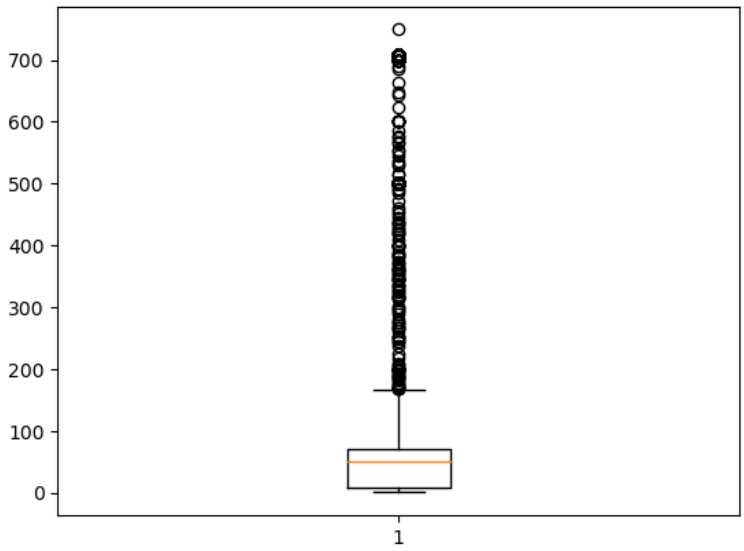
In [19]:

```
print(f"Количество записей в наборе данных: {data_all.shape[0]}")
```

```
Количество записей в наборе данных: 207628
```

In [20]:

```
plt.boxplot(data_all['Coordinate uncertainty (m)']);
```




Избавился от аномально высоких показаний погрешности. Да, выбросы остались, но решение оставить некоторое количество записей с завышенными показаниями обусловлено тем, что удаление такого большого количества записей (примерно 30к) сильно урежет размер данных.

### 1.2.4 Неинформативные и избыточные признаки

Очевидно, что колонки (они же *атрибуты* или *признаки*), которые содержат всего одно значение, не дадут какого-либо прироста информации ни для модели машинного обучения, ни для вас как исследователя. Можно по каждому признаку пройтись и вызвать у него команду, которая показывает количество уникальных значений, но зачем, если можно **автоматизировать** отбор таких признаков.

In [21]:

```
useless_features = list() # так можно объявить пустой список
for column in data_all.columns:
    if len(data_all[column].unique()) == 1:
        useless_features.append(column)
        print(f"{column} has {len(data_all[column].unique())} уникальных значений: {data_all[column].unique()}")
```

```
Basis of record has 1 уникальных значений: ['HumanObservation']
Kingdom has 1 уникальных значений: ['Animalia']
Phylum has 1 уникальных значений: ['Chordata']
Class has 1 уникальных значений: ['Mammalia']
Country has 1 уникальных значений: ['United Kingdom']
```

На мой взгляд, результаты очевидны. Поскольку датасет посвящён млекопитающим, обитающим на территории Великобритании, информации о растениях мы в нём не встретим, поэтому царство всего одно -- Animalia. Это же касается и признаков Тип (Phylum) и Класс (Class), поскольку млекопитающие относятся к типу хордовых и классу млекопитающих в таксономической системе.
Избавимся от этих признаков.

In [22]:

```
data_all.drop(columns=useless_features, inplace=True)
data_all.head(5)
```

Out[22]:

| | Scientific name | Common name | Species ID (TVK) | Taxon Rank | Start date | Start date day | Start date month | Start date year | Latitude (WGS84) | Longitude (WGS84) | Coordinate uncertainty (m) | Identification verification status | Survey key |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Capreolus capreolus | Roe Deer | NHMSYS0000080203 | species | 31/01/2018 | 31 | 1 | 2018 | 57.400200 | -4.267900 | 50.0 | Accepted - correct | iRecord \| Mammals |
| **1** | Erinaceus europaeus | West European Hedgehog | NBNSYS0000005078 | species | 20/01/2021 | 20 | 1 | 2021 | 51.519214 | -0.738969 | 57.0 | Accepted - correct | iRecord \| App |
| **3** | Talpa europaea | European Mole | NBNSYS0000005079 | species | 25/01/2017 | 25 | 1 | 2017 | 53.322473 | -1.731987 | 70.7 | Accepted | iRecord \| Import |
| **4** | Sciurus carolinensis | Eastern Grey Squirrel | NHMSYS0000332764 | species | 06/01/2016 | 6 | 1 | 2016 | 52.581700 | -1.717600 | 50.0 | Accepted | iRecord \| Mammals |
| **5** | Talpa europaea | European Mole | NBNSYS0000005079 | species | 18/01/2022 | 18 | 1 | 2022 | 50.740415 | -0.775410 | 13.0 | Accepted - considered correct | iRecord \| App |

Обратим внимание на признаки Start date, Start date day, Start date month, Start date year -- информация в них несколько избыточна. Зачем иметь отдельные столбцы с частями даты начала наблюдения, если она уже есть в отдельном столбце, но в виде строки? Достаточно этот признак из типа object перевести в тип **datetime**, тогда появится возможность получить доступ к частям даты начала наблюдения "из коробки". Проделаем эти действия:

In [23]:

```
# удаляем избыточную информацию
data_all.drop(columns=['Start date day', 'Start date month', 'Start date year'], inplace=True)
# приводим признак Start date к типу datetime
data_all['Start date'] = pd.to_datetime(data_all['Start date'])
# проверяю, что получил
print(data_all['Start date'].dtype)
data_all.head(3)
```

```
datetime64[ns]
```

```
/tmp/ipykernel_20628/3621885612.py:4: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (th
e default) was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  data_all['Start date'] = pd.to_datetime(data_all['Start date'])
```

Out[23]:

| | Scientific name | Common name | Species ID (TVK) | Taxon Rank | Start date | Latitude (WGS84) | Longitude (WGS84) | Coordinate uncertainty (m) | Identification verification status | Survey key | Order | Family | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Capreolus capreolus | Roe Deer | NHMSYS0000080203 | species | 2018-01-31 | 57.400200 | -4.267900 | 50.0 | Accepted - correct | iRecord \| Mammals | Artiodactyla | Cervidae | Ca |
| **1** | Erinaceus europaeus | West European Hedgehog | NBNSYS0000005078 | species | 2021-01-20 | 51.519214 | -0.738969 | 57.0 | Accepted - correct | iRecord \| App | Insectivora | Erinaceidae | Er |
| **3** | Talpa europaea | European Mole | NBNSYS0000005079 | species | 2017-01-25 | 53.322473 | -1.731987 | 70.7 | Accepted | iRecord \| Import | Insectivora | Talpidae | |

Это явно не всё. Существует ещё такое понятие как **техническая информация**. Техническая информация -- это всё, что даёт представление о носителе информации, но не об исследуемом объекте. Например, сейчас мы исследуем данные и хотим получить на их основе информацию о **млекопитающих**, но признаки Identification verification status, Survey key и Species ID (TVK), кажется, ничего не говорят о том, какое это млекопитающее, но несут информацию о том, как были собраны данные в том или ином случае. Исследуем эти признаки более подробно.

In [24]:

```python
def feature_summary(df, column):
    print(column)
    print(f"Количество уникальных значений признака: {len(df[column].unique())}")
    print(f"Уникальные значения признака: {df[column].unique()}")
    print(f"Тип данных признака: {df[column].dtype}")
    print("-" * 20)

technical_features = ['Identification verification status', 'Survey key', 'Species ID (TVK)']
for feature in technical_features:
    feature_summary(data_all, feature)
```

```
Identification verification status
Количество уникальных значений признака: 3
Уникальные значения признака: ['Accepted - correct' 'Accepted' 'Accepted - considered correct']
Тип данных признака: object
--------------------
Survey key
Количество уникальных значений признака: 67
Уникальные значения признака: ['iRecord | Mammals' 'iRecord | App' 'iRecord | Import'
 'iRecord | General data' 'Wild Sheffield | General records'
 'Mammal Society | Mammal Atlas' 'iRecord | LERC Wales'
 'iRecord | Multi-site records'
 'iRecord | Moors for the Future Mountain Hare postcard'
 'MammalNet-Europe | IMammalia App' 'iRecord | National Trust Bioblitzes'
 'iRecord | ISpot' 'Merseyside BioBank | General records'
 'iRecord | App General Survey' 'NNSS | RISC Mammal'
 'Nature Counts - Nottinghamshire WT | General records' 'iRecord | Bats'
 'YWT INNS Mapper | Observations' 'UKBMS | Transects'
 'iRecord | National Trust for Scotland'
 "BRC website's forms | Mammal Atlas"
 'iRecord | National Trust Bioblitzes 2016'
 'iRecord | National Trust record lists' 'iRecord | Dorset Mammals'
 'iRecord | Garden Bioblitz 2013' 'iRecord | Garden BioBlitz 2014'
 'iRecord | SEWeb INNS' 'Merseyside BioBank | NW Brown Hare Project'
 'UKBMS | WCBS-BBS' 'YNU | Submit a Sighting' 'YNU | Terrestrial'
 'iRecord | AquaInvaders' 'iRecord | Boat of Garten Wildlife Group'
 'iRecord | UK lakes portal'
 'Corfe Mullen Nature Watch | Corfe Mullen Bioblitz 2011'
 'iRecord | National Trust Sites' 'iRecord | Rinse' 'iRecord | GBNNSIP'
 'iRecord | Suffolk WT' 'iRecord | Cairngorms NP'
 'Marine Biological Association | Marine Sightings Network'
 'iRecord | Garden Bioblitz 2015' 'Merseyside BioBank | Local Sites'
 'iRecord | Woodland Trust' 'iRecord | National Trust'
 'Corfe Mullen Nature Watch | Corfe Mullen NatureWatch'
 'iRecord | Garden Bioblitz 2012' 'iRecord | Garden Bioblitz 2016'
 'Heathland Surveillance Network | General' 'York University | Records'
 'UKBMS | WCBS-BC' 'Wild Sheffield | Records sourced from iRecord App'
 'iRecord | ARC Explorer'
 'Nature Counts - Northumberland WT | General records'
 'iRecord | Natural England Surveys' 'iRecord | Wytham'
 'Nature Counts - Herefordshire WT | General records' 'iRecord | SEPA'
 'iRecord | Glenlivet & Tomintoul Area Recording'
 'iRecord | Garden Bioblitz 2017'
 'Wild Sheffield | Records sourced from iRecord general data'
 'iRecord | Moths' 'Surrey BIC | Surrey Mammal Group'
 'NFBR | General records' 'YNU | Intertidal Recording'
 'YWT INNS Mapper | Imported infection points format 2'
 'Surrey BIC | General records']
Тип данных признака: object
--------------------
Species ID (TVK)
Количество уникальных значений признака: 116
Уникальные значения признака: ['NHMSYS0000080203' 'NBNSYS0000005078' 'NBNSYS0000005079'
 'NHMSYS0000332764' 'NHMSYS0000080210' 'NBNSYS0000005108'
 'NHMSYS0000080219' 'NBNSYS0000005127' 'NHMSYS0000080184'
 'NHMSYS0000080204' 'NBNSYS0000005128' 'NHMSYS0000080218'
 'NHMSYS0000080212' 'NBNSYS0000005106' 'NBNSYS0000005133'
 'NHMSYS0000080211' 'NHMSYS0000080188' 'NBNSYS0000005135'
 'NBNSYS0000136687' 'NHMSYS0020546253' 'NHMSYS0020755138'
 'NBNSYS0000040775' 'NHMSYS0020774297' 'NBNSYS0000005143'
 'NHMSYS0000080191' 'NHMSYS0000080207' 'NBNSYS0000005121'
 'NBNSYS0000136578' 'NBNSYS0000005148' 'NBNSYS0000005145'
 'NHMSYS0000080186' 'NBNSYS0000005102' 'NBNSYS0000005144'
 'NBNSYS0000040730' 'NBNSYS0000005081' 'NBNSYS0000005137'
 'NBNSYS0000005082' 'NHMSYS0020001355' 'NBNSYS0100004720'
 'NHMSYS0000080214' 'NHMSYS0000080178' 'NHMSYS0001699414'
 'NHMSYS0000528008' 'NHMSYS0000080209' 'NHMSYS0000080190'
 'NHMSYS0000080174' 'NBNSYS0000138842' 'NBNSYS0000005129'
 'NHMSYS0000080189' 'NHMSYS0020001356' 'NHMSYS0000080187'
 'NBNSYS0000005179' 'NHMSYS0000528028' 'NHMSYS0000080177'
 'NBNSYS0000037278' 'NHMSYS0001699416' 'NHMSYS0000080183'
 'NHMSYS0000376191' 'NBNSYS0000005142' 'NBNSYS0000005107'
 'NHMSYS0000080201' 'NBNSYS0000005172' 'NHMSYS0000080176'
 'NHMSYS0000528030' 'NBNSYS0000135668' 'NHMSYS0000080185'
 'NBNSYS0000005183' 'NHMSYS0000528026' 'NBNSYS0000163066'
 'NHMSYS0000080202' 'NBNSYS0000040788' 'NBNSYS0000188733'
 'NHMSYS0021109725' 'NHMSYS0001699418' 'NHMSYS0020636762'
 'NBNSYS0000137324' 'NBNSYS0000040117' 'NBNSYS0000005159'
 'NHMSYS0020975282' 'NBNSYS0000135345' 'NHMSYS0000376470'
 'NHMSYS0000528024' 'NHMSYS0020774295' 'NBNSYS0000005181'
 'NBNSYS0000005173' 'NBNSYS0000135957' 'NHMSYS0000080192'
 'NHMSYS0000376187' 'NHMSYS0020774285' 'NBNSYS0000130775'
 'NHMSYS0000080200' 'NHMSYS0000080208' 'NBNSYS0000005160'
 'NHMSYS0000080194' 'NHMSYS0000080171' 'NBNSYS0000042516'
 'NHMSYS0000080215' 'NHMSYS0000376773' 'NBNSYS0000005167'
 'NHMSYS0020755426' 'NHMSYS0000080173' 'NHMSYS0020774282'
 'NBNSYS0000132921' 'NHMSYS0000080193' 'NHMSYS0000377289'
 'NBNSYS0000005103' 'NHMSYS0000080213' 'NHMSYS0020975131'
 'NBNSYS0000130945' 'NBNSYS0000005139' 'NBNSYS0000005166'
 'NHMSYS0001699417' 'NHMSYS0000080196' 'NHMSYS0000376462'
```

```
 'NHMSYS0000377112' 'NBNSYS0000040193']
Тип данных признака: object
--------------------
```

Безусловно, это информация технического характера (метаданные). Однако глядя на них, тоже можно сделать выводы:

- Неподтверждённых записей в датасете нет (все они Accepted), можно избавиться;
- Признак Ключ опроса (Survey key) не несёт информации о млекопитающем, поскольку говорит о том, откуда получены данные, можно избавиться;
- Признак Species ID (TVK) -- это идентификаторы каждого вида. От него пока избавляться не станем, исследуем дальше

In [25]:

```
data_all.drop(columns=['Identification verification status', 'Survey key'], inplace=True)
```

Чтобы исследовать идентификаторы видов, необходимо рассмотреть признаки, отвечающие за наименования видов. Их у нас два: Scientific name и Common name.

In [26]:

```
names_features = ['Scientific name', 'Common name']
for feature in names_features:
    feature_summary(data_all, feature)
```

```
Scientific name
Количество уникальных значений признака: 115
Уникальные значения признака: ['Capreolus capreolus' 'Erinaceus europaeus' 'Talpa europaea'
 'Sciurus carolinensis' 'Apodemus sylvaticus' 'Sciurus vulgaris'
 'Oryctolagus cuniculus' 'Mustela erminea' 'Myotis nattereri'
 'Muntiacus reevesi' 'Mustela nivalis' 'Lepus europaeus'
 'Rattus norvegicus' 'Lepus timidus' 'Lutra lutra' 'Micromys minutus'
 'Vulpes vulpes' 'Phoca vitulina' 'Plecotus' 'Arvicola amphibius'
 'Myodes glareolus' 'Felis silvestris x catus' 'Neovison vison'
 'Cervus elaphus' 'Meles meles' 'Microtus agrestis' 'Mus musculus'
 'Pipistrellus' 'Hydropotes inermis' 'Dama dama' 'Nyctalus noctula'
 'Plecotus auritus' 'Cervus nippon' 'Myotis' 'Sorex araneus'
 'Halichoerus grypus' 'Sorex minutus' 'Pipistrellus pipistrellus'
 'Pipistrellus pygmaeus' 'Muscardinus avellanarius'
 'Barbastella barbastellus' 'Lepus timidus subsp. scoticus'
 'Eptesicus serotinus' 'Apodemus flavicollis' 'Martes martes'
 'Neomys fodiens' 'Vespertilio' 'Mustela putorius' 'Felis catus'
 'Pipistrellus nathusii' 'Tursiops truncatus' 'Myotis daubentonii'
 'Rhinolophus hipposideros' 'Sciurus' 'Mustela erminea subsp. erminea'
 'Myotis mystacinus' 'Rattus' 'Sus scrofa' 'Castor fiber' 'Capra hircus'
 'Phocoena phocoena' 'Rhinolophus ferrumequinum'
 'Myotis mystacinus/brandtii' 'Nyctalus' 'Nyctalus leisleri'
 'Rangifer tarandus' 'Myotis brandtii' 'Physeter macrocephalus'
 'Ovis aries' 'Felis' 'Odobenus rosmarus' 'Muntiacus muntjak'
 'Microtus arvalis subsp. orcadensis' 'Myotis alcathoe' 'Rhinolophus'
 'Mustela furo x putorius' 'Megaptera novaeangliae' 'Equus asinus' 'Mus'
 'Capra aegagrus' 'Myotis bechsteinii' 'Mustela putorius subsp. furo'
 'Delphinus delphis' 'Orcinus orca' 'Ovis' 'Balaenoptera acutorostrata'
 'Capra' 'Equus ferus' 'Barbastella' 'Bos taurus' 'Microtus arvalis'
 'Balaenoptera physalus' 'Grampus griseus' 'Macropus rufogriseus'
 'Lepus timidus subsp. hibernicus' 'Glis glis' 'Homo sapiens'
 'Ziphius cavirostris' 'Myodes glareolus subsp. skomerensis'
 'Crocidura suaveolens' 'Canis lupus subsp. familiaris' 'Eptesicus'
 'Globicephala melas' 'Sus' 'Plecotus austriacus' 'Rattus rattus'
 'Sus domesticus' 'Bos' 'Erignathus barbatus' 'Hyperoodon ampullatus'
 'Mustela erminea subsp. hibernica' 'Lagenorhynchus albirostris' 'Canis'
 'Pipistrellus kuhlii' 'Ondatra zibethicus']
Тип данных признака: object
--------------------
Common name
Количество уникальных значений признака: 114
Уникальные значения признака: ['Roe Deer' 'West European Hedgehog' 'European Mole'
 'Eastern Grey Squirrel' 'Wood Mouse' 'Eurasian Red Squirrel'
 'European Rabbit' 'Stoat' "Natterer's Bat" 'Chinese Muntjac' 'Weasel'
 'Brown Hare' 'Brown Rat' 'Mountain Hare' 'Eurasian Otter' 'Harvest Mouse'
 'Red Fox' 'Harbour Seal' 'Long-eared Bat species' 'European Water Vole'
 'Bank Vole' 'Wild Cat Hybrid' 'American Mink' 'Red Deer'
 'Eurasian Badger' 'Field Vole' 'House Mouse' 'Pipistrelle Bat species'
 'Chinese Water Deer' 'Fallow Deer' 'Noctule Bat' 'Brown Long-eared Bat'
 'Sika Deer' 'Myotis Bat species' 'Eurasian Common Shrew' 'Grey Seal'
 'Eurasian Pygmy Shrew' 'Common Pipistrelle' 'Soprano Pipistrelle'
 'Hazel Dormouse' 'Western Barbastelle' 'Serotine' 'Yellow-necked Mouse'
 'Pine Marten' 'Eurasian Water Shrew' 'Vespertilio Bat species' 'Polecat'
 'Feral Cat' 'Pipistrelle' "Nathusius's Pipistrelle"
 'Bottle-nosed Dolphin' "Daubenton's Bat" 'Lesser Horseshoe Bat'
 'Squirrel' 'Whiskered Bat' 'Rat spp.' 'Wild Boar' 'Beaver' 'Feral Goat'
 'Common Porpoise' 'Greater Horseshoe Bat' "Whiskered/Brandt's Bat"
 'Nyctalus Bat species' 'Lesser Noctule' 'Reindeer' "Brandt's Bat"
 'Sperm Whale' 'Feral Sheep' 'Cat' 'Walrus' 'Indian Muntjac' 'Orkney Vole'
 'Alcathoe Bat' 'Horseshoe Bat species' 'Polecat-Ferret' 'Humpback Whale'
 'Donkey' 'Mouse' 'Wild Goat' "Bechstein's Bat" 'Feral Ferret'
 'Common Dolphin' 'Killer Whale' 'Sheep' 'Minke Whale' 'Goat' 'Horse'
 'Barbastelle Bat species' 'Domestic Cattle' 'Common Vole' 'Fin Whale'
 "Risso's Dolphin" 'Red-necked Wallaby' 'Irish Hare' 'Fat Dormouse'
 'Human' "Cuvier's Beaked Whale" 'Skomer Vole'
 'Lesser White-toothed Shrew' 'Domestic Dog' 'Eptesicus Bat species'
 'Long-finned Pilot Whale' 'Pig' 'Grey Long-eared Bat' 'Black Rat'
 'Domestic Pig' 'Cattle' 'Bearded Seal' 'Northern Bottlenose Whale'
 'Irish Stoat' 'White-beaked Dolphin' 'Dog' "Kuhl's Pipistrelle"
 'Musk Rat']
Тип данных признака: object
--------------------
```

Что видим: количество уникальных значений во всех трёх признаках не совпадает. Необходимо посмотреть, какое научное наименование какому обиходному наименованию и идентификатору соответствует. Можно попробовать сделать это через словарь, установив соответствие между каждым идентификатором и парой из научного и обиходного наименований млекопитающих.

In [27]:

```
# заведу пустой словарь
association_dict = {}
# в этом словаре ключами будут выступать идентификаторы
for id_tvk in data_all['Species ID (TVK)'].unique():
    temp_data = data_all[data_all['Species ID (TVK)'] == id_tvk]
    # добавляю запись словарь вида идентификатор: (научное имя, распространенное имя)
    association_dict[id_tvk] = (temp_data['Scientific name'].unique(),  temp_data['Common name'].unique())
# проверка на наличие в кортеже с именами случаев, где
# какому-то идентификатору соответствует больше одного из наименований
for idx, names in association_dict.items():
    if len(association_dict[idx][0]) > 1 or len(association_dict[idx][1]) > 1:
        print(association_dict[idx])
else:
    print('Ничего не найдено')
```

```
Ничего не найдено
```

In [28]:

```
# разделим кортеж с наименованиями на два списка
scientific, common = [], []
for value in association_dict.values():
    scientific.append(value[0])
    common.append(value[1])

print("Длина первого списка: ", len(scientific))
print("Длина второго списка: ", len(common))
```

```
Длина первого списка:  116
Длина второго списка:  116
```

Посмотрим дубликаты среди обиходных наименований:

In [30]:

```
[element for element in common if common.count(element) > 1]
```

Out[30]:

```
[array(['Stoat'], dtype=object),
 array(['Mountain Hare'], dtype=object),
 array(['Mountain Hare'], dtype=object),
 array(['Stoat'], dtype=object)]
```

Посмотрим дубликаты среди научных наименований:

In [31]:

```
[element for element in scientific if scientific.count(element) > 1]
```

Out[31]:

```
[array(['Pipistrellus pipistrellus'], dtype=object),
 array(['Pipistrellus pipistrellus'], dtype=object)]
```

А теперь выведем записи с дублирующимся научным наименованием животного.

In [32]:

```
for idx, names in association_dict.items():
    if 'Pipistrellus pipistrellus' in names[0]:
        print(f"{idx} : {association_dict[idx]}")
```

```
NHMSYS0020001355 : (array(['Pipistrellus pipistrellus'], dtype=object), array(['Common Pipistrelle'], dtype
=object))
NHMSYS0020001356 : (array(['Pipistrellus pipistrellus'], dtype=object), array(['Pipistrelle'], dtype=objec
t))
```

Можно сделать вывод, что одному виду животного с научным наименованием Pipistrellus pipistrellus соответствуют два обиходных наименования 'Common Pipistrelle' и просто 'Pipistrelle'. Однако проверим, что указано для записей со значением 'Pipistrelle' в атрибуте Taxon Rank:

In [33]:

```
data_all[data_all['Common name'] == 'Pipistrelle']['Taxon Rank'].unique()
```

Out[33]:

```
array(['species aggregate'], dtype=object)
```

Делаем вывод, что Pipistrelle -- это совокупность разных видов, а Common Pipistrelle -- это отдельный вид. Лишними признаками здесь будут научное и обиходное названия млекопитающих в силу того, что среди них есть записи с дублирующимися значениями. Идентификаторы, используемые в атласе млекопитающих Великобритании, более точны и позволяют определить вид более точно.

In [34]:

```
# удаляю ненужные признаки
data_all.drop(columns=names_features, inplace=True)
```

In [35]:

```
data_all.head(3)
```

Out[35]:

| | Species ID (TVK) | Taxon Rank | Start date | Latitude (WGS84) | Longitude (WGS84) | Coordinate uncertainty (m) | Order | Family | Genus | State/Province |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NHMSYS0000080203 | species | 2018-01-31 | 57.400200 | -4.267900 | 50.0 | Artiodactyla | Cervidae | Capreolus | Scotland |
| **1** | NBNSYS0000005078 | species | 2021-01-20 | 51.519214 | -0.738969 | 57.0 | Insectivora | Erinaceidae | Erinaceus | England |
| **3** | NBNSYS0000005079 | species | 2017-01-25 | 53.322473 | -1.731987 | 70.7 | Insectivora | Talpidae | Talpa | England |

### 1.2.5 Остальные признаки

У нас остались неисследованными признаки Taxon Rank, Order, Family, Genus и State/Province. Посмотрим на признак Taxon Rank.

In [36]:

```
feature_summary(data_all, 'Taxon Rank')
```

```
Taxon Rank
Количество уникальных значений признака: 5
Уникальные значения признака: ['species' 'genus' 'species hybrid' 'subspecies' 'species aggregate']
Тип данных признака: object
--------------------
```

In [37]:

```
data_all['Taxon Rank'].value_counts()/data_all.shape[0] * 100
```

Out[37]:

```
Taxon Rank
species              97.039898
genus                 1.608646
species aggregate     0.901131
subspecies            0.251411
species hybrid        0.198913
Name: count, dtype: float64
```

Видим, что есть несколько категорий у признака Taxon Rank, среди которых большую долю (97%) занимает категория species (виды). Помимо этой категории, встречаются записи, в которых ранг в таксономической системе -- род и обощённые виды, то есть не отдельный вид. Так как записей с такими категориями признака немного, их можно удалить.

In [38]:

```
data_all.drop(index=data_all[(data_all['Taxon Rank'] == 'genus') | (data_all['Taxon Rank'] == 'species aggregate')].in
```

Выделим отдельные атрибуты с таксономическими категориями и проанализируем их.

In [39]:

```
misunderstanded_features = ['Order', 'Family', 'Genus']
for feat in misunderstanded_features:
    feature_summary(data_all, feat)
```

```
Order
Количество уникальных значений признака: 10
Уникальные значения признака: ['Artiodactyla' 'Insectivora' 'Rodentia' 'Lagomorpha' 'Carnivora'
 'Chiroptera' 'Cetartiodactyla' 'Perissodactyla' 'Diprotodontia'
 'Primates']
Тип данных признака: object
--------------------
Family
Количество уникальных значений признака: 27
Уникальные значения признака: ['Cervidae' 'Erinaceidae' 'Talpidae' 'Sciuridae' 'Muridae' 'Leporidae'
 'Mustelidae' 'Vespertilionidae' 'Cricetidae' 'Canidae' 'Phocidae'
 'Felidae' 'Soricidae' 'Myoxidae' 'Delphinidae' 'Rhinolophidae' 'Suidae'
 'Castoridae' 'Bovidae' 'Phocoenidae' 'Physeteridae' 'Odobenidae'
 'Balaenopteridae' 'Equidae' 'Macropodidae' 'Hominidae' 'Ziphiidae']
Тип данных признака: object
--------------------
Genus
Количество уникальных значений признака: 63
Уникальные значения признака: ['Capreolus' 'Erinaceus' 'Talpa' 'Sciurus' 'Apodemus' 'Oryctolagus'
 'Mustela' 'Myotis' 'Muntiacus' 'Lepus' 'Rattus' 'Lutra' 'Micromys'
 'Vulpes' 'Phoca' 'Arvicola' 'Myodes' 'Felis' 'Neovison' 'Cervus' 'Meles'
 'Microtus' 'Mus' 'Hydropotes' 'Dama' 'Nyctalus' 'Plecotus' 'Sorex'
 'Halichoerus' 'Pipistrellus' 'Muscardinus' 'Barbastella' 'Eptesicus'
 'Martes' 'Neomys' 'Tursiops' 'Rhinolophus' 'Sus' 'Castor' 'Capra'
 'Phocoena' 'Rangifer' 'Physeter' 'Ovis' 'Odobenus' 'Megaptera' 'Equus'
 'Delphinus' 'Orcinus' 'Balaenoptera' 'Bos' 'Grampus' 'Macropus' 'Glis'
 'Homo' 'Ziphius' 'Crocidura' 'Canis' 'Globicephala' 'Erignathus'
 'Hyperoodon' 'Lagenorhynchus' 'Ondatra']
Тип данных признака: object
--------------------
```

Признаки можно оставить неизменёнными.

### 1.2.7 Кодирование категориальных переменных

Под *категориальными данными* мы понимаем данные, которые не имеют численного представления, они могут иметь как и два уникальных значения (бинарные признаки), так и более. Для работы с признаками надо произвести *кодирование категориальных признаков* - процедуру, которая представляет собой некоторое преобразование категориальных признаков в численное представление по некоторым оговоренным ранее правилам. Есть несколько возможностей для кодирования.

#### 1.2.7.1 Label encoder

Данный тип кодирования является наиболее часто используемым, преобразование представляет собой однозначное соответствие число <-> уникальное значение категориального признака. Первое (выбранное каким-то образом) уникальное значение кодируется нулем, второе единицей, и так далее, последнее кодируется числом, равным количеству уникальных значений минус единица.

In [40]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
categorical_features = ['Taxon Rank', 'Order', 'Family', 'Genus', 'Species ID (TVK)', 'State/Province']
data_labled = data_all.copy()
for cat_feat in categorical_features:
    data_labled[cat_feat] = labelencoder.fit_transform(data_labled[cat_feat].values)
    print(labelencoder.classes_) # покажем, в каком порядке кодировались категории признака
data_labled
```

```
['species' 'species hybrid' 'subspecies']
['Artiodactyla' 'Carnivora' 'Cetartiodactyla' 'Chiroptera' 'Diprotodontia'
 'Insectivora' 'Lagomorpha' 'Perissodactyla' 'Primates' 'Rodentia']
['Balaenopteridae' 'Bovidae' 'Canidae' 'Castoridae' 'Cervidae'
 'Cricetidae' 'Delphinidae' 'Equidae' 'Erinaceidae' 'Felidae' 'Hominidae'
 'Leporidae' 'Macropodidae' 'Muridae' 'Mustelidae' 'Myoxidae' 'Odobenidae'
 'Phocidae' 'Phocoenidae' 'Physeteridae' 'Rhinolophidae' 'Sciuridae'
 'Soricidae' 'Suidae' 'Talpidae' 'Vespertilionidae' 'Ziphiidae']
['Apodemus' 'Arvicola' 'Balaenoptera' 'Barbastella' 'Bos' 'Canis' 'Capra'
 'Capreolus' 'Castor' 'Cervus' 'Crocidura' 'Dama' 'Delphinus' 'Eptesicus'
 'Equus' 'Erignathus' 'Erinaceus' 'Felis' 'Glis' 'Globicephala' 'Grampus'
 'Halichoerus' 'Homo' 'Hydropotes' 'Hyperoodon' 'Lagenorhynchus' 'Lepus'
 'Lutra' 'Macropus' 'Martes' 'Megaptera' 'Meles' 'Micromys' 'Microtus'
 'Muntiacus' 'Mus' 'Muscardinus' 'Mustela' 'Myodes' 'Myotis' 'Neomys'
 'Neovison' 'Nyctalus' 'Odobenus' 'Ondatra' 'Orcinus' 'Oryctolagus' 'Ovis'
 'Phoca' 'Phocoena' 'Physeter' 'Pipistrellus' 'Plecotus' 'Rangifer'
 'Rattus' 'Rhinolophus' 'Sciurus' 'Sorex' 'Sus' 'Talpa' 'Tursiops'
 'Vulpes' 'Ziphius']
['NBNSYS0000005078' 'NBNSYS0000005079' 'NBNSYS0000005081'
 'NBNSYS0000005082' 'NBNSYS0000005102' 'NBNSYS0000005103'
 'NBNSYS0000005106' 'NBNSYS0000005107' 'NBNSYS0000005108'
 'NBNSYS0000005121' 'NBNSYS0000005127' 'NBNSYS0000005128'
 'NBNSYS0000005129' 'NBNSYS0000005133' 'NBNSYS0000005135'
 'NBNSYS0000005137' 'NBNSYS0000005139' 'NBNSYS0000005142'
 'NBNSYS0000005143' 'NBNSYS0000005144' 'NBNSYS0000005145'
 'NBNSYS0000005148' 'NBNSYS0000005159' 'NBNSYS0000005160'
 'NBNSYS0000005166' 'NBNSYS0000005167' 'NBNSYS0000005172'
 'NBNSYS0000005173' 'NBNSYS0000005179' 'NBNSYS0000005181'
 'NBNSYS0000005183' 'NBNSYS0000040117' 'NBNSYS0000040193'
 'NBNSYS0000040775' 'NBNSYS0000042516' 'NBNSYS0000163066'
 'NBNSYS0000188733' 'NBNSYS0100004720' 'NHMSYS0000080171'
 'NHMSYS0000080173' 'NHMSYS0000080174' 'NHMSYS0000080176'
 'NHMSYS0000080177' 'NHMSYS0000080178' 'NHMSYS0000080183'
 'NHMSYS0000080184' 'NHMSYS0000080185' 'NHMSYS0000080186'
 'NHMSYS0000080187' 'NHMSYS0000080188' 'NHMSYS0000080189'
 'NHMSYS0000080190' 'NHMSYS0000080191' 'NHMSYS0000080192'
 'NHMSYS0000080193' 'NHMSYS0000080194' 'NHMSYS0000080196'
 'NHMSYS0000080200' 'NHMSYS0000080201' 'NHMSYS0000080202'
 'NHMSYS0000080203' 'NHMSYS0000080204' 'NHMSYS0000080207'
 'NHMSYS0000080208' 'NHMSYS0000080209' 'NHMSYS0000080210'
 'NHMSYS0000080211' 'NHMSYS0000080212' 'NHMSYS0000080213'
 'NHMSYS0000080214' 'NHMSYS0000080215' 'NHMSYS0000080218'
 'NHMSYS0000080219' 'NHMSYS0000332764' 'NHMSYS0000376470'
 'NHMSYS0000376773' 'NHMSYS0000377112' 'NHMSYS0000528008'
 'NHMSYS0000528024' 'NHMSYS0000528026' 'NHMSYS0000528028'
 'NHMSYS0001699414' 'NHMSYS0001699416' 'NHMSYS0001699417'
 'NHMSYS0001699418' 'NHMSYS0020001355' 'NHMSYS0020546253'
 'NHMSYS0020636762' 'NHMSYS0020755138' 'NHMSYS0020755426'
 'NHMSYS0020774282' 'NHMSYS0020774285' 'NHMSYS0020774295'
 'NHMSYS0020774297' 'NHMSYS0020975131' 'NHMSYS0020975282'
 'NHMSYS0021109725']
['England' 'Isle of Man' 'Northern Ireland' 'Scotland' 'Wales']
```

Out[40]:

| | Species ID (TVK) | Taxon Rank | Start date | Latitude (WGS84) | Longitude (WGS84) | Coordinate uncertainty (m) | Order | Family | Genus | State/Province |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 60 | 0 | 2018-01-31 | 57.400200 | -4.267900 | 50.0 | 0 | 4 | 7 | 3 |
| **1** | 0 | 0 | 2021-01-20 | 51.519214 | -0.738969 | 57.0 | 5 | 8 | 16 | 0 |
| **3** | 1 | 0 | 2017-01-25 | 53.322473 | -1.731987 | 70.7 | 5 | 24 | 59 | 0 |
| **4** | 73 | 0 | 2016-01-06 | 52.581700 | -1.717600 | 50.0 | 9 | 21 | 56 | 0 |
| **5** | 1 | 0 | 2022-01-18 | 50.740415 | -0.775410 | 13.0 | 5 | 24 | 59 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **227317** | 1 | 0 | 1972-05-21 | 50.676231 | -1.329116 | 707.1 | 5 | 24 | 59 | 0 |
| **227318** | 49 | 0 | 2013-05-04 | 50.953207 | -0.138821 | 7.1 | 1 | 2 | 61 | 0 |
| **227319** | 73 | 0 | 2017-05-26 | 52.038682 | -0.286097 | 7.1 | 9 | 21 | 56 | 0 |
| **227320** | 60 | 0 | 2022-05-26 | 53.518050 | -2.566450 | 5.0 | 0 | 4 | 7 | 0 |
| **227321** | 6 | 0 | 2018-05-14 | 53.412563 | -1.888709 | 7.1 | 6 | 11 | 26 | 0 |

202417 rows × 10 columns

Главный недостаток Label Encoder'a - создание избыточных зависимостей в данных. После преобразования получилось, что в атрибуте State/Province значение Wales имеет численное значение 4, а Isle of Man -- 1, что дает нам право говорить, что Wales в 4 раз больше (круче и тд.) чем Isle of Man по признаку State\Province. Однако, в исходных данных таких зависимостей не было, что и является существенным недостатком данного варианта кодирования.

#### 1.2.7.2 One-Hot Encoder

Данный тип кодирования, основывается на создании бинарных признаков, которые показывают принадлежность к уникальному значению. Проще говоря, на примере нашего признака State/Province, мы создаем бинарные признаки для всех уникальных значений: England, Northern Ireland, ..., где признак принадлежности к категории England имеет значение 1, если объект в признаке State/Province имеет значение England и нуль при всех других. Давайте посмотрим на примере признака State/Province:

In [41]:

```
data_ohe = data_all.copy()

data_new = pd.get_dummies(data_ohe, columns=categorical_features)
data_new
```

Out[41]:

| | Start date | Latitude (WGS84) | Longitude (WGS84) | Coordinate uncertainty (m) | Taxon Rank_NBNSYS0000005078 | Taxon Rank_NBNSYS0000005079 | Taxon Rank_NBNSYS0000005081 | Taxon Rank_NBNSYS000 |
|---|---|---|---|---|---|---|---|---|
| **0** | 2018-01-31 | 57.400200 | -4.267900 | 50.0 | False | False | False | |
| **1** | 2021-01-20 | 51.519214 | -0.738969 | 57.0 | True | False | False | |
| **3** | 2017-01-25 | 53.322473 | -1.731987 | 70.7 | False | True | False | |
| **4** | 2016-01-06 | 52.581700 | -1.717600 | 50.0 | False | False | False | |
| **5** | 2022-01-18 | 50.740415 | -0.775410 | 13.0 | False | True | False | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **227317** | 1972-05-21 | 50.676231 | -1.329116 | 707.1 | False | True | False | |
| **227318** | 2013-05-04 | 50.953207 | -0.138821 | 7.1 | False | False | False | |
| **227319** | 2017-05-26 | 52.038682 | -0.286097 | 7.1 | False | False | False | |
| **227320** | 2022-05-26 | 53.518050 | -2.566450 | 5.0 | False | False | False | |
| **227321** | 2018-05-14 | 53.412563 | -1.888709 | 7.1 | False | False | False | |

202417 rows × 209 columns

Главный недостаток One-Hot Encoder'a заключается в существенном увеличении объема данных, так как большие по количеству уникальных значений признаки кодируются большим количеством бинарных признаков.

In [42]:

```
print(len(data_new.columns))
data_new.columns
```

209

Out[42]:

```
Index(['Start date', 'Latitude (WGS84)', 'Longitude (WGS84)',
       'Coordinate uncertainty (m)', 'Taxon Rank_NBNSYS0000005078',
       'Taxon Rank_NBNSYS0000005079', 'Taxon Rank_NBNSYS0000005081',
       'Taxon Rank_NBNSYS0000005082', 'Taxon Rank_NBNSYS0000005102',
       'Taxon Rank_NBNSYS0000005103',
       ...
       'Species ID (TVK)_Sus', 'Species ID (TVK)_Talpa',
       'Species ID (TVK)_Tursiops', 'Species ID (TVK)_Vulpes',
       'Species ID (TVK)_Ziphius', 'State/Province_England',
       'State/Province_Isle of Man', 'State/Province_Northern Ireland',
       'State/Province_Scotland', 'State/Province_Wales'],
      dtype='object', length=209)
```

Есть и другие способы кодирования: https://habr.com/ru/articles/666234/ (https://habr.com/ru/articles/666234/)

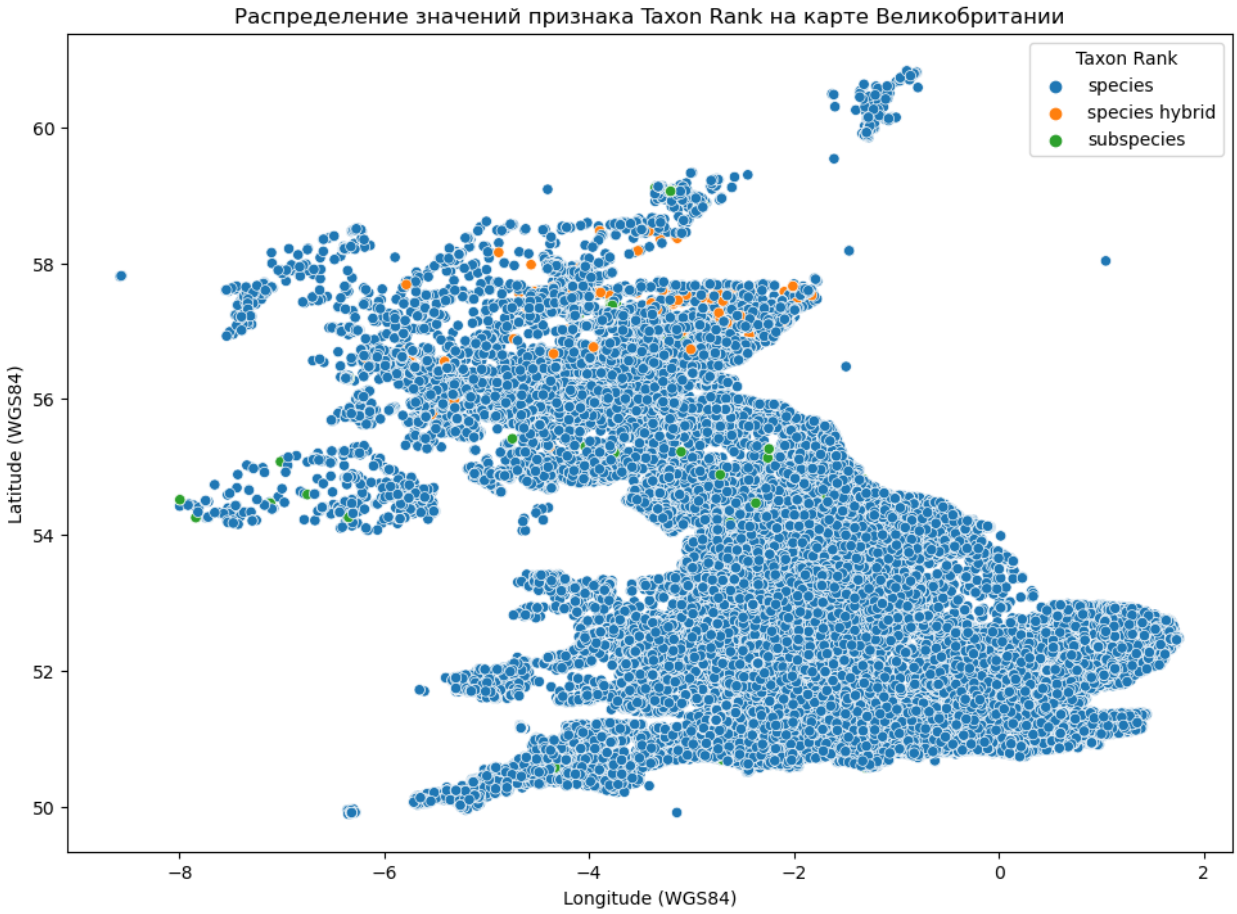
### 1.2.6 Визуализация данных для лучшего понимания данных

In [43]:

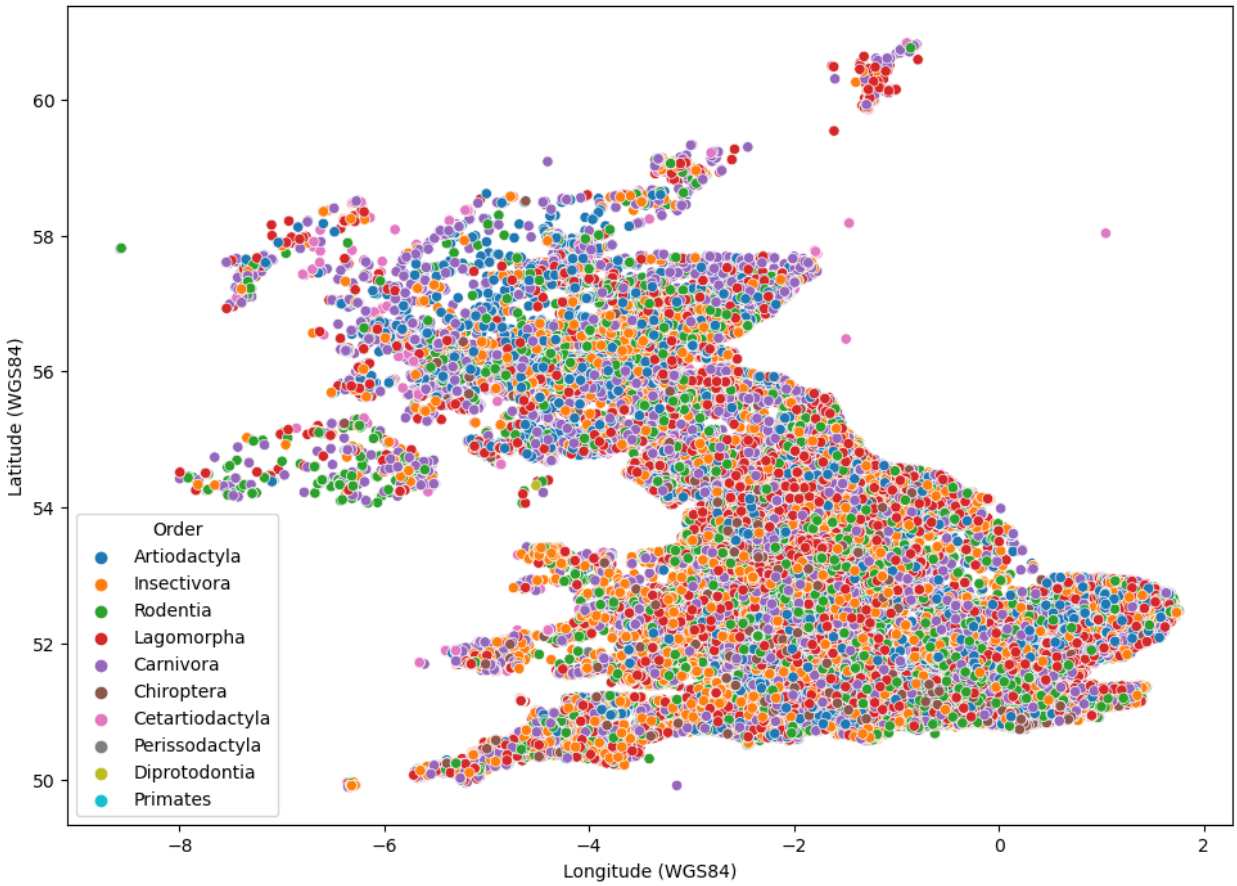
```
from matplotlib import rcParams

# figure size in inches
rcParams['figure.figsize'] = 11.7,8.27

for feature in ['Taxon Rank', 'Order', 'Family', 'Genus', 'Species ID (TVK)', 'State/Province']:
    plt.title(f"Распределение значений признака {feature} на карте Великобритании")
    sns.scatterplot(data=data_all, x='Longitude (WGS84)', y='Latitude (WGS84)', hue=feature)
    #data_all.plot.scatter(x='Longitude (WGS84)', y='Latitude (WGS84)',
    #         c=feature, colormap='viridis', figsize=(6, 10),)
    plt.show();
```

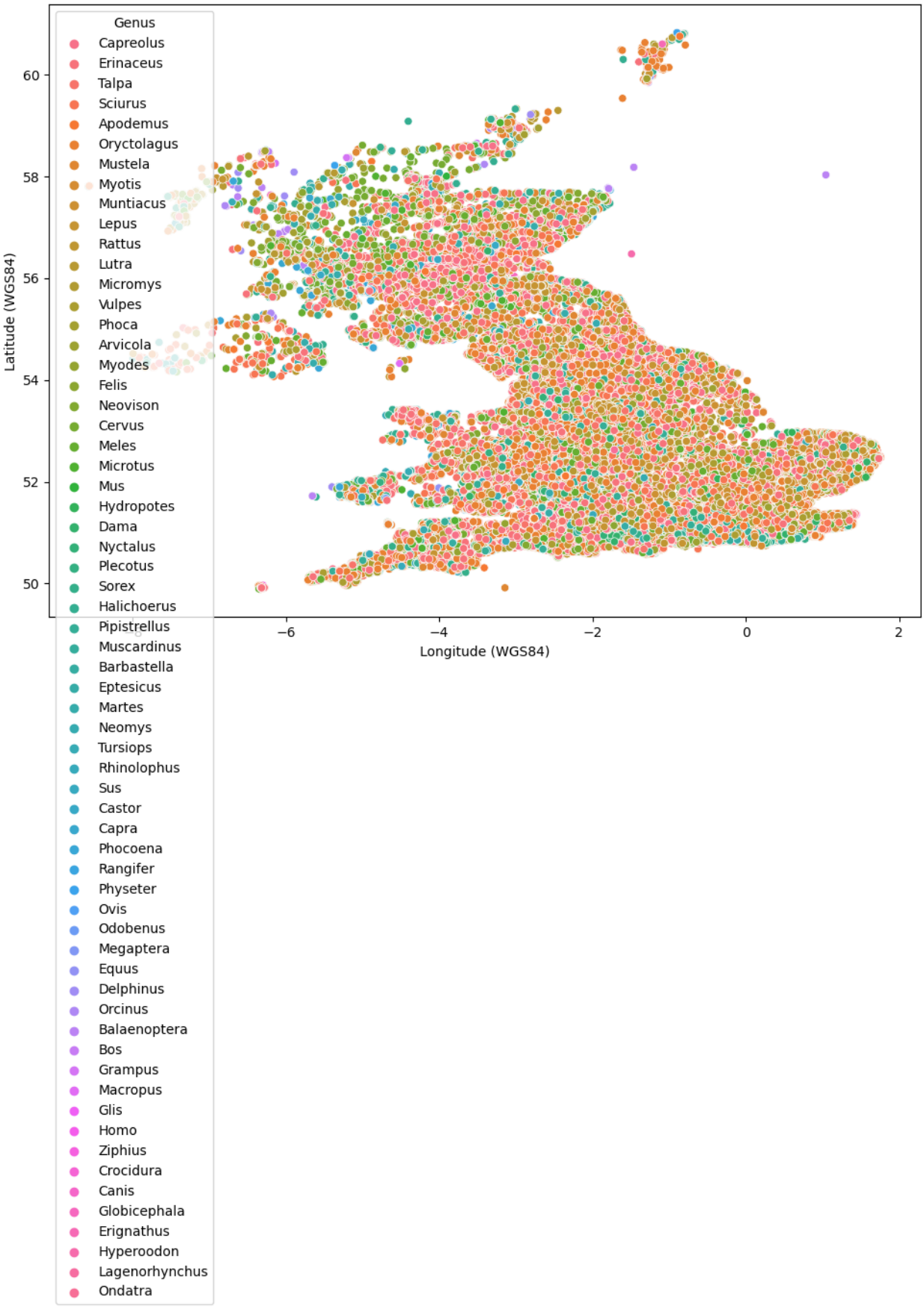


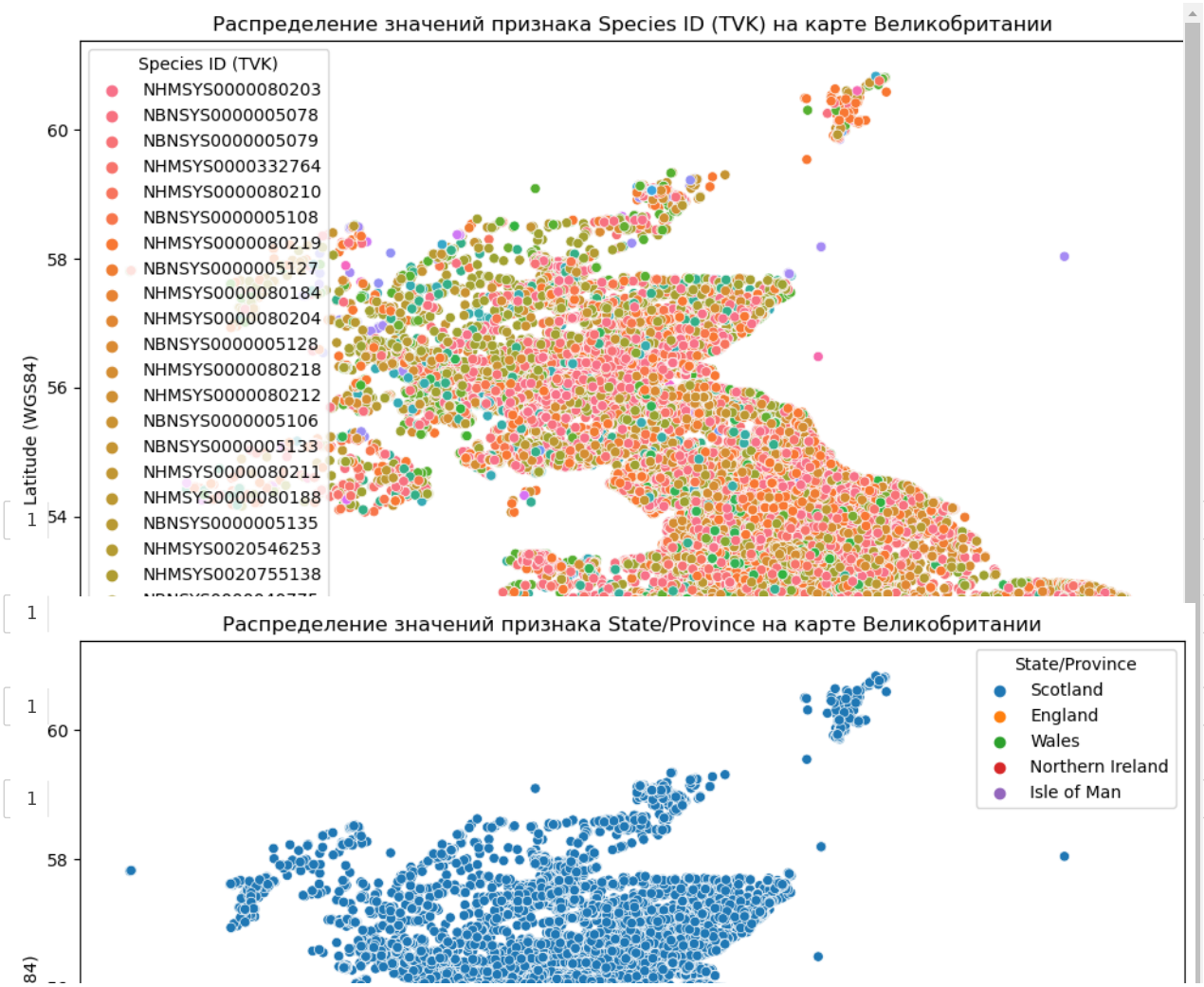
Распределение значений признака Order на карте Великобритании




Распределение значений признака Family на карте Великобритании

Распределение значений признака Genus на карте Великобритании




Распределение значений признака Species ID (TVK) на карте Великобритании
Species ID (TVK)
NHMSYS0000080203
NBNSYS0000005078
NBNSYS0000005079
NHMSYS0000332764
NHMSYS0000080210
NBNSYS0000005108
NHMSYS0000080219
NBNSYS0000005127
NHMSYS0000080184
NHMSYS0000080204
NBNSYS0000005128
NHMSYS0000080218
NHMSYS0000080212
NBNSYS0000005106
NBNSYS0000005133
NHMSYS0000080211
NHMSYS0000080188
NBNSYS0000005135
NHMSYS0020546253
NHMSYS0020755138
Latitude (WGS84)
60
58
56
54


Распределение значений признака State/Province на карте Великобритании
State/Province
Scotland
England
Wales
Northern Ireland
Isle of Man
60
58