Nutridata: Exploring USDA Food Nutrient Database for Data Analysis and Insights

NutriData is a project focused on exploring the US Department of Agriculture Food Nutrient Database for comprehensive data analysis and deriving valuable insights. Through data preprocessing, cleaning, and exploratory analysis techniques, this project aims to uncover patterns, correlations, and trends in the nutritional content of various food items. By leveraging statistical analysis, visualization, and machine learning methods, it seeks to provide data-driven insights into nutrition, dietary choices, and health implications. The project aims to contribute to the field of nutrition science and enable evidence-based recommendations for individuals, healthcare professionals, and researchers in promoting healthy eating habits and making informed decisions related to nutrition and well-being.

Dataset:https://raw.githubusercontent.com/wesm/pydatabook/2ndedition/datasets/usda_food/database.json

This dataset is a JSON file containing information about various food items from the United States Department of Agriculture (USDA). Each food item is represented as a dictionary with multiple attributes, such as description, group, nutrients, and manufacturer.

```
import json
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from tabulate import tabulate

url = "https://raw.githubusercontent.com/wesm/pydata-book/2nd-edition/datasets/usda_food/database.json"
response = requests.get(url)
if response.status_code == 200:
    with open("./usda_food/database.json", "wb") as file:
        file.write(response.content)
        print("File downloaded successfully.")
else:
   print("Failed to download the file.")
with open("./usda_food/database.json", "r") as file:
    db = json.load(file)

print("Number of records in the database:", len(db))
sns.set_style("whitegrid")
print("Type of database:", type(db))
print("Type of first element:", type(db[0]))
# db is a list of dictionaries
print("keys of the first dictionary:" , db[0].keys())
print("First nutrient of the first record:", db[0]['nutrients'][0])
print("Second nutrient of the first record:", db[0]['nutrients'][1])
```

```
File downloaded successfully.
Number of records in the database: 6636
Type of database: <class 'list'>
Type of first element: <class 'dict'>
keys of the first dictionary: dict_keys(['id', 'description', 'tags', 'manufacturer', 'group', 'portions', 'nutrients'])
First nutrient of the first record: {'value': 25.18, 'units': 'g', 'description': 'Protein', 'group': 'Composition'}
Second nutrient of the first record: {'value': 29.2, 'units': 'g', 'description': 'Total lipid (fat)', 'group': 'Composition'}
```

```
nutrients = pd.DataFrame(db[0]['nutrients'])
# Extract the nutrients data from the first record
nutrients = nutrients[['description', 'group', 'units', 'value']]
```

```
# Select specific columns from the nutrients DataFrame
print(nutrients.head(10))
```

```
                    description        group units    value
0                       Protein  Composition     g    25.18
1             Total lipid (fat)  Composition     g    29.20
2  Carbohydrate, by difference  Composition     g     3.06
3                           Ash        Other     g     3.28
4                        Energy       Energy  kcal   376.00
5                         Water  Composition     g    39.28
6                        Energy       Energy    kJ  1573.00
7          Fiber, total dietary  Composition     g     0.00
8                   Calcium, Ca     Elements    mg   673.00
9                      Iron, Fe     Elements    mg     0.64
```

```
info_keys = ['description', 'group', 'id', 'manufacturer']
# Specify the columns to include in the 'info' DataFrame
info = pd.DataFrame(db, columns=info_keys)
# Create the 'info' DataFrame with the specified columns
print(info[:5])
# Display the first 5 rows of the 'info' DataFrame
```

```
                         description  ... manufacturer
0                     Cheese, caraway  ...
1                     Cheese, cheddar  ...
2                        Cheese, edam  ...
3                        Cheese, feta  ...
4  Cheese, mozzarella, part skim milk  ...
```

```
info.info()
```

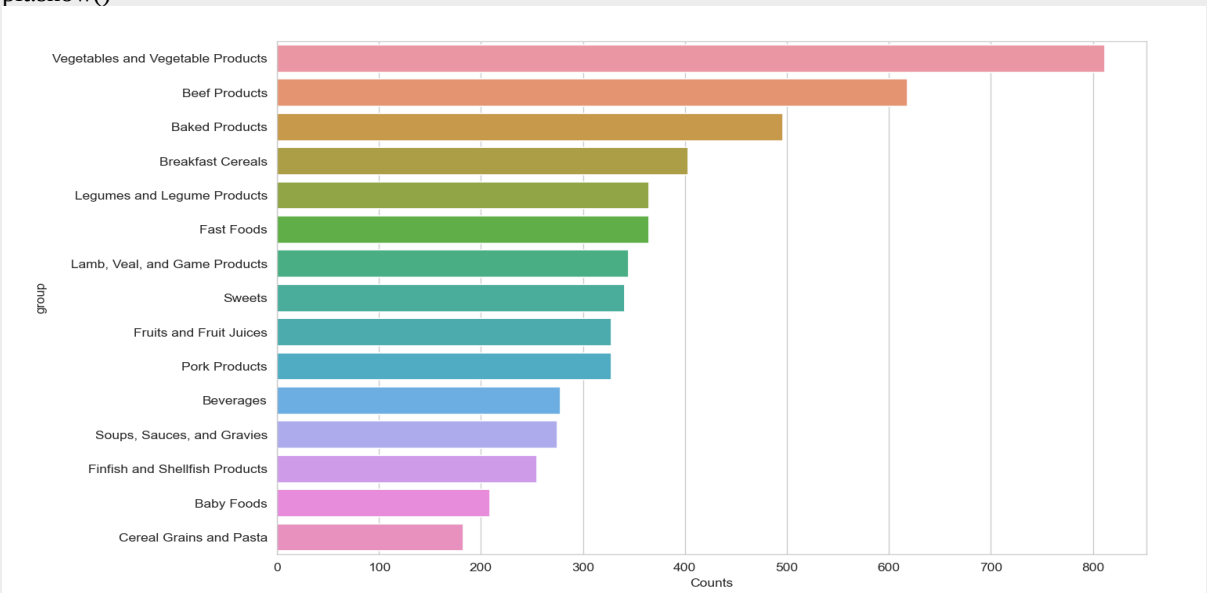
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6636 entries, 0 to 6635
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   description   6636 non-null   object
 1   group         6636 non-null   object
 2   id            6636 non-null   int64
 3   manufacturer  5195 non-null   object
dtypes: int64(1), object(3)
memory usage: 207.5+ KB
```

```
df = pd.DataFrame(db)
print("keys of the first dictionary:" , db[0].keys())
table = tabulate([df.iloc[0]], headers='keys', tablefmt='psql')
#To print all records from your DataFrame df in a table format
print(table)
```

```
keys of the first dictionary: dict_keys(['id', 'description', 'tags', 'manufacturer', 'group', 'portions', 'nutrients'])
+------+----------------+--------+----------------+-----------------------+-----------------------------------------+-----------------------------------
|   id | description    | tags   | manufacturer   | group                 | portions                                | nutrients
|------+----------------+--------+----------------+-----------------------+-----------------------------------------+-----------------------------------
| 1008 | Cheese, caraway | []     |                | Dairy and Egg Products | [{'amount': 1, 'unit': 'oz', 'grams': 28.35}] | [{'value': 25.18, 'units': 'g', 'desc
+------+----------------+--------+----------------+-----------------------+-----------------------------------------+-----------------------------------
```

```
plt.figure(figsize=(12, 8))
_ = sns.barplot(x=pd.value_counts(info.description)[:15], y=pd.value_counts(info.description)[:15].index)
```

```
_ = plt.xlabel("Counts")
plt.show()
```




```
print(db[0]['nutrients'][0]['value'])
# for analysis of nutrients
nutrients =[]
for i in db:
    fnut = pd.DataFrame(i['nutrients'])
    fnut["id"] = i['id']
    nutrients.append(fnut)
print(type(nutrients))
nutrients = pd.concat(nutrients,ignore_index=True)
nutrients = nutrients[["description", "group", "units", "value", "id"]]
print(type(nutrients))
print(nutrients.head(20))
```

```
25.18
<class 'list'>
<class 'pandas.core.frame.DataFrame'>
                   description       group units     value    id
0                      Protein  Composition     g    25.180  1008
1            Total lipid (fat)  Composition     g    29.200  1008
2   Carbohydrate, by difference  Composition     g     3.060  1008
3                           Ash        Other     g     3.280  1008
4                        Energy       Energy  kcal   376.000  1008
5                         Water  Composition     g    39.280  1008
6                        Energy       Energy    kJ  1573.000  1008
7          Fiber, total dietary  Composition     g     0.000  1008
8                   Calcium, Ca     Elements    mg   673.000  1008
9                      Iron, Fe     Elements    mg     0.640  1008
10                Magnesium, Mg     Elements    mg    22.000  1008
11                Phosphorus, P     Elements    mg   490.000  1008
12                 Potassium, K     Elements    mg    93.000  1008
13                   Sodium, Na     Elements    mg   690.000  1008
14                     Zinc, Zn     Elements    mg     2.940  1008
15                   Copper, Cu     Elements    mg     0.024  1008
16                Manganese, Mn     Elements    mg     0.021  1008
17                 Selenium, Se     Elements   mcg    14.500  1008
18                Vitamin A, IU     Vitamins    IU  1054.000  1008
19                      Retinol     Vitamins   mcg   262.000  1008
```

```
print(nutrients.duplicated().sum())
nutrients = nutrients.drop_duplicates()
```

```
14179
```

```
col_mapping = {'description': 'food', 'group': 'fgroup'}
info = info.rename(columns=col_mapping, copy=False)
info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6636 entries, 0 to 6635
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   food          6636 non-null   object
 1   fgroup        6636 non-null   object
 2   id            6636 non-null   int64 
 3   manufacturer  5195 non-null   object
dtypes: int64(1), object(3)
memory usage: 207.5+ KB
```

```
col_mapping = {'description': 'nutrient', 'group': 'nutgroup'}
nutrients = nutrients.rename(columns=col_mapping, copy=False)
nutrients.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 375176 entries, 0 to 389354
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   nutrient  375176 non-null  object 
 1   nutgroup  375176 non-null  object 
 2   units     375176 non-null  object 
 3   value     375176 non-null  float64
 4   id        375176 non-null  int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 17.2+ MB
```

```
ndata = pd.merge(nutrients,info,on='id', how='outer')
ndata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375176 entries, 0 to 375175
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   nutrient      375176 non-null  object 
 1   nutgroup      375176 non-null  object 
 2   units         375176 non-null  object 
 3   value         375176 non-null  float64
 4   id            375176 non-null  int64  
 5   food          375176 non-null  object 
 6   fgroup        375176 non-null  object 
 7   manufacturer  293054 non-null  object 
dtypes: float64(1), int64(1), object(6)
memory usage: 22.9+ MB
```

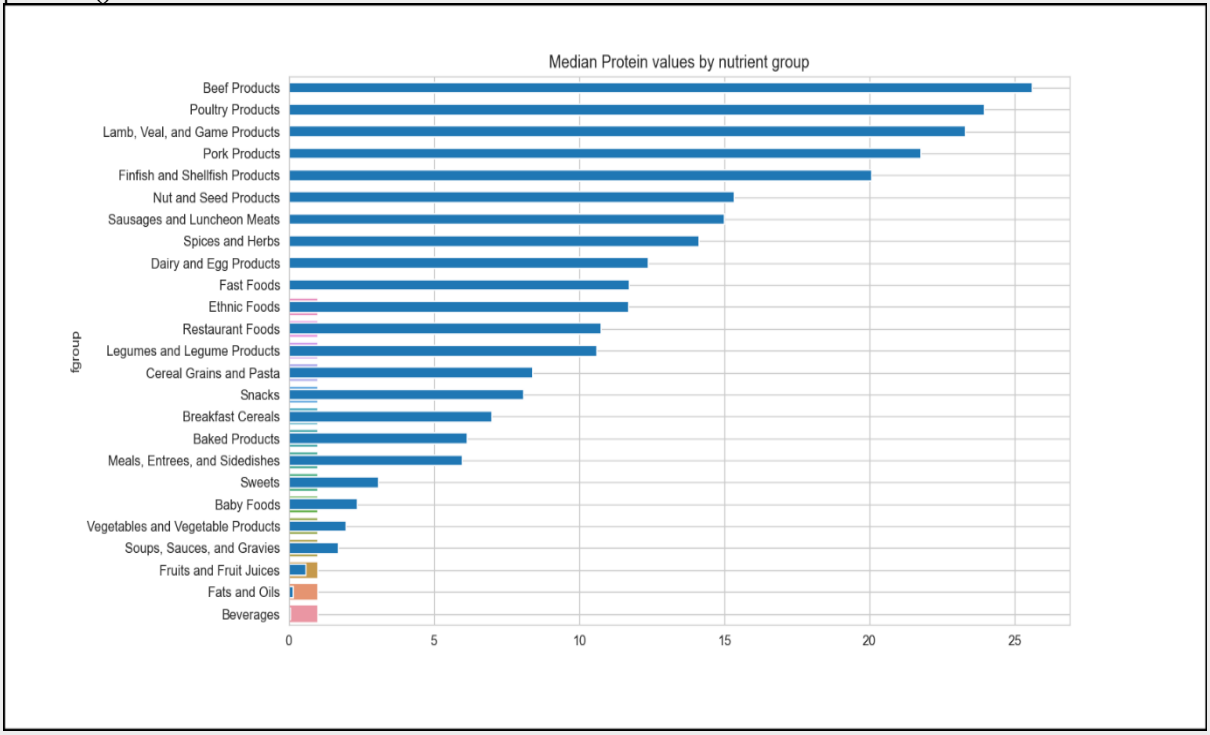
```
print(ndata.iloc[30000])
```

```
nutrient                                   Glycine
nutgroup                                Amino Acids
units                                             g
value                                          0.04
id                                             6158
food           Soup, tomato bisque, canned, condensed
fgroup                     Soups, Sauces, and Gravies
manufacturer
Name: 30000, dtype: object
```

```
result = ndata.groupby(['nutrient', 'fgroup'])['value'].quantile(0.5)
print(result.index.values)
```

```
[('Adjusted Protein', 'Sweets')
 ('Adjusted Protein', 'Vegetables and Vegetable Products')
 ('Alanine', 'Baby Foods') ... ('Zinc, Zn', 'Spices and Herbs')
 ('Zinc, Zn', 'Sweets') ('Zinc, Zn', 'Vegetables and Vegetable Products')]
```

```
_ = result['Protein'].sort_values().plot(kind="barh", figsize=(15, 10),title="Median Protein values by nutrient group")
plt.show()
```




```
by_nutrient = ndata.groupby(['nutgroup','nutrient'])
get_max = lambda x: x.loc[x.value.idxmax()]
get_min = lambda x: x.loc[x.value.idxmin()]
max_foods = by_nutrient.apply(get_max)[['value','food']]
max_foods.food = max_foods.food.str[:50]
print(max_foods)
```

```
                                        value                                        food
nutgroup    nutrient
Amino Acids Alanine                     8.009                Gelatins, dry powder, unsweetened
            Arginine                    7.436                     Seeds, sesame flour, low-fat
            Aspartic acid              10.203                             Soy protein isolate
            Cystine                     1.307       Seeds, cottonseed flour, low fat (glandless)
            Glutamic acid              17.452                             Soy protein isolate
...                                       ...                                             ...
Vitamins    Vitamin D2 (ergocalciferol) 28.100                        Mushrooms, maitake, raw
            Vitamin D3 (cholecalciferol)27.400                 Fish, halibut, Greenland, raw
            Vitamin E (alpha-tocopherol)149.400                                Oil, wheat germ
            Vitamin E, added            46.550  Cereals ready-to-eat, GENERAL MILLS, Multi-Gra...
            Vitamin K (phylloquinone)  1714.500                           Spices, sage, ground

[94 rows x 2 columns]
```

max_foods_amino = max_foods.loc[**'Amino Acids'**][**'food'**]
print(max_foods_amino)
#print(max_foods.loc[**'Amino Acids'**][**'food'**])

```
nutrient
Alanine                          Gelatins, dry powder, unsweetened
Arginine                              Seeds, sesame flour, low-fat
Aspartic acid                                 Soy protein isolate
Cystine               Seeds, cottonseed flour, low fat (glandless)
Glutamic acid                                 Soy protein isolate
Glycine                          Gelatins, dry powder, unsweetened
Histidine              Whale, beluga, meat, dried (Alaska Native)
Hydroxyproline    KENTUCKY FRIED CHICKEN, Fried Chicken, ORIGINA...
Isoleucine        Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Leucine           Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Lysine            Seal, bearded (Oogruk), meat, dried (Alaska Na...
Methionine                   Fish, cod, Atlantic, dried and salted
Phenylalanine     Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Proline                          Gelatins, dry powder, unsweetened
Serine            Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Threonine         Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Tryptophan         Sea lion, Steller, meat with fat (Alaska Native)
Tyrosine          Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Valine            Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Name: food, dtype: object
```