# A Deep Dive into the Open Food Facts Dataset

Achraf Elkhanjari · Follow

11 min read · Mar 30, 2024

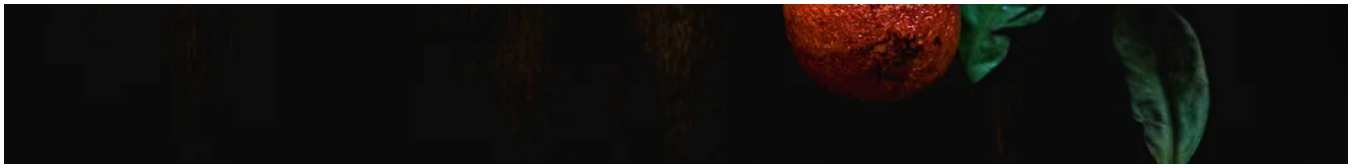


Photo by Edgar Castrejon on Unsplash

It has never been more important to know how the food we eat affects our health and the planet, since the food comes from a lot of different cultures.

The Open Food Facts database may be described as the best collection of data about any food products from around the world, produced by **20,000+** dedicated volunteer workers.

In this context, an important indicator for assessing the environmental impact of food products is the **eco-score.** It offers a measurable indicator of sustainability that helps consumers make more ecological choices when making their purchases.


Picture of eco-score by Olivier le Goaër

We use data analysis to explore the Open Food Facts dataset and gain insights into eco-score coverage.

## Data Collection and Preparation

Our main source of data is a MongoDB dump that we downloaded from the Open Food Facts data website.

After the data was obtained as a Mongo dump, we used the `mongoexport` command line to enable the direct exportation of data from the database into a CSV file format.

The command that is used to export data is:

```
mongoexport --uri="mongodb://"your_mongodb_host":your_port/FoodFacts" --collection=produ
--fields=<your_field_names> --out="/path/to/your/exported_data.csv"
```

**export_data.mongodb** hosted with ❤️ by **GitHub** **view raw**

> *You can replace **'your_field_names'** with any columns you want to retrieve in the database. for example: code, creator, created_t, etc..*

## Loading the Data into a DataFrame

The next step is importing the necessary libraries and reading the CSV file into a Pandas DataFrame. I always use **Jupyter Notebook** for my analysis.

```
import pandas as pd # data manipulation and analysis
import seaborn as sns # data visualisation
import matplotlib.pyplot as plt # data visualisation
%matplotlib inline # Enable inline display of matplotlib plots in the notebook.
```

**librairies.py** hosted with ❤️ by **GitHub** **view raw**

After importing the necessary libraries, we named our data `df` :

```
df = pd.read_csv('exported_data.csv',encoding='utf-8',low_memory=False)
```

**pandas.py** hosted with ❤️ by **GitHub** **view raw**

Get the first 5 rows of the dataset:

Different columns in our dataset have unique data about the food products, like names and brands to more important features such as eco-scores.

A brief explanation of the columns in our dataset is provided below:

- **code:** Barcode of the product.
- **creator:** Contributor who first added the product
- **created_t:** Date that the product was added in UNIX format.
- **last_modified_t:** Date that the product page was last modified.
- **product_name:** Name of the product.
- **generic_name:** A generic description of the product.
- **quantity:** The amount of product in the packaging with unit.
- **packaging:** Information about packaging material.
- **brands:** Product's brand name.
- **categories:** Product categories like groceries, and biscuits.
- **origins**: Origins of ingredients.
- **manufacturing_places:** Locations where the product was manufactured.
- **labels:** Certifications or labels of the product.

- **emb_codes:** Official manufacturing site codes.
- **purchases_places:** Locations where the product can be purchased.
- **stores:** Retail stores that carry the product.
- **countries:** list of countries where the product is sold.
- **ingredients_text:** List of ingredients used in the product.
- **allergens:** Known allergens present in the product.
- **nutriments:** Nutritional information per serving.
- **additives_n:** Number of food additives in the product.
- **ingredients_from_palm_oil_n:** Number of ingredients made from palm oil.
- **nutrition_grades:** nutrition grade (‘a’ to ‘e’).
- **eco_score_score:** Numerical score that quantifies the product’s environmental impact.
- **eco_score_grade:** Letter grade derived from ecoscore, used as a reference for environmental impact.

### Statistical summary of the dataset:

```
1  df.describe()
```

**description.py** hosted with ❤️ by **GitHub** **view raw**

| | created_t | last_modified_t | additives_n | ingredients_from_palm_oil_n | ingredients_that_may_be_from_palm_oil_n | ecoscore_score |
|---|---|---|---|---|---|---|
| count | 1.277375e+06 | 1.277375e+06 | 634658.000000 | 634658.000000 | 634658.000000 | 462715.000000 |
| mean | 1.532772e+09 | 1.625641e+09 | 2.012849 | 0.022705 | 0.073146 | 49.488573 |
| std | 4.463280e+07 | 5.338893e+07 | 2.862717 | 0.150958 | 0.311746 | 25.358566 |
| min | 1.328021e+09 | 1.353582e+09 | 0.000000 | 0.000000 | 0.000000 | -30.000000 |
| 25% | 1.495295e+09 | 1.587576e+09 | 0.000000 | 0.000000 | 0.000000 | 32.000000 |
| 50% | 1.539007e+09 | 1.627200e+09 | 1.000000 | 0.000000 | 0.000000 | 49.000000 |
| 75% | 1.570185e+09 | 1.678274e+09 | 3.000000 | 0.000000 | 0.000000 | 72.000000 |
| max | 1.683912e+09 | 1.707735e+09 | 46.000000 | 3.000000 | 6.000000 | 125.000000 |

summary of our dataset

> ***df.describe()*** *method returns descriptive statistics about a dataframe. For numeric data, the result's index will include* `count` *,* `mean` *,* `std` *,* `min` *,* `max` *, as well as lower,* `50(median)` *and upper percentiles.*

When we look at the eco-score distribution, the average is about **49.49,** where the products had a medium amount of environmental impact. In addition, the standard deviation is about **25.36,** indicating a diverse set of products with different eco-scores.

To give an example, a product with an eco-score close to **125** would be considered to have a very low environmental impact, and products close to the minimum score (**-30**) are not preferred, likely due to a high carbon footprint or significant pollution.

## Data Cleaning

It is widely known that data scientists spend a significant amount of time cleaning data; in fact, you may have heard that data scientists spend **80%** of their time finding, cleaning, and structuring data, with just **20%** analysing it and providing insights.

Using messy data to do analysis will result in erroneous predictions, leading to poor decisions and negative outcomes. Furthermore, the majority of machine learning algorithms need your data cleaned and suitable for modelling.

With all that said, we will discover the missing values, data formatting and what is the best approach to fix these.

### Missing data

```python
missing_percentage = df.isnull().mean() * 100
# Sort the percentages in descending order
missing_percentage_sorted = missing_percentage.sort_values(ascending=False)
# Print the sorted percentages
print("Columns with the highest percentages of missing values:")
print(missing_percentage_sorted)
```

**missing_values.py** hosted with ❤️ by **GitHub** **view raw**

```
Columns with the highest percentages of missing values:
origins                                      93.672571
traces                                       92.533281
emb_codes                                    91.665016
manufacturing_places                         90.752833
generic_name                                 90.697642
purchase_places                              88.490381
allergens                                    84.564674
stores                                       80.545729
packaging                                    79.858303
serving_size                                 68.295998
labels                                       65.365065
ecoscore_score                               63.776103
quantity                                     62.333222
additives_n                                  50.315452
ingredients_that_may_be_from_palm_oil_n      50.315452
ingredients_text                             50.315452
ingredients_from_palm_oil_n                  50.315452
compared_to_category                         36.122360
categories                                   35.711831
brands                                       32.791545
product_name                                  3.424445
ingredients_that_may_be_from_palm_oil_tags    1.521245
ingredients_from_palm_oil_tags                1.521245
additives_tags                                1.521245
countries                                     0.222409
nutrition_grades                              0.131441
ecoscore_grade                                0.025756
creator                                       0.000313
pnns_groups_1                                 0.000078
nutriments                                    0.000000
last_modified_t                               0.000000
pnns_groups_2                                 0.000000
states                                        0.000000
created_t                                     0.000000
code                                          0.000000
dtype: float64
```

Percentage of missing values in each column

**Columns with >90% missing values:** `origins`, `traces`, `emb_codes`, `manufacturing_places`, and `generic_name` are the columns with the highest percentage of missing data. We will drop these columns because they have a minimal impact on our analysis.

```
columns_to_drop = ['origins', 'traces', 'emb_codes', 'manufacturing_places', 'generic_na
df.drop(columns=columns_to_drop, inplace=True)
```

**drop.py** hosted with ❤️ by **GitHub** **view raw**

**Moderate to high missing values:** The `ecoscore_score` and `ecoscore_grade` columns are essential to our analysis. Missing values in `ecoscore_score` will be imputed with **-1** to show that there is no data, while `ecoscore_grade` will be filled with '**unknown**' as it is a categorical column.

```
df['ecoscore_score']=df['ecoscore_score'].fillna(-1)
df['ecoscore_grade']=df['ecoscore_grade'].fillna('unknown')
```

**fill1.py** hosted with ❤️ by **GitHub** **view raw**

**Filling other columns:** For columns like `categories`, `compared_to_category`, `brands`, and `product_name` will be replaced with '**unknown**'.

```
key_columns = ['categories', 'brands', 'product_name','compared_to_category']
for column in key_columns:
    df[column].fillna('unknown', inplace=True)
```

**other_columns.py** hosted with ❤️ by **GitHub** **view raw**

**Data Formatting and Normalisation:**

To facilitate analysis and ensure consistency, the following formatting procedures will be implemented:

- **Unix Timestamps to Datetime:** `created_t` and `last_modified_t` will be converted from UNIX timestamps to a more interpretable datetime format.

```
df['created_t'] = pd.to_datetime(df['created_t'], unit='s')
df['last_modified_t'] = pd.to_datetime(df['last_modified_t'], unit='s')
```

**unix_to_normal_date.py** hosted with ❤️ by **GitHub** **view raw**

- **Text Normalisation:** All textual data, particularly in columns such as `product_name`, `brands`, `categories` and `compared_to_category`, will be converted to lowercase to maintain consistency.

```
text_columns = ['product_name', 'brands', 'categories','compared_to_category']
for column in text_columns:
    df[column] = df[column].str.lower()
```

**lowercase.py** hosted with ❤️ by **GitHub** **view raw**

Now let's check if there are any missing values in other columns.

```
missing_percentage = df.isnull().mean() * 100
# Sort the percentages in descending order
missing_percentage_sorted = missing_percentage.sort_values(ascending=False)
# Print the sorted percentages
print("Columns with the highest percentages of missing values:")
print(missing_percentage_sorted)
```

**missing_values.py** hosted with ❤️ by **GitHub** **view raw**

```
Columns with the highest percentages of missing values:
purchase_places                             88.490381
allergens                                   84.564674
stores                                      80.545729
packaging                                   79.858303
serving_size                                68.295998
labels                                      65.365065
quantity                                    62.333222
ingredients_text                            50.315452
ingredients_that_may_be_from_palm_oil_n     50.315452
additives_n                                 50.315452
ingredients_from_palm_oil_n                 50.315452
ingredients_that_may_be_from_palm_oil_tags   1.521245
ingredients_from_palm_oil_tags               1.521245
additives_tags                               1.521245
countries                                    0.222409
nutrition_grades                             0.131441
creator                                      0.000313
pnns_groups_1                                0.000078
pnns_groups_2                                0.000000
states                                       0.000000
ecoscore_score                               0.000000
code                                         0.000000
nutriments                                   0.000000
compared_to_category                         0.000000
categories                                   0.000000
brands                                       0.000000
product_name                                 0.000000
last_modified_t                              0.000000
created_t                                    0.000000
ecoscore_grade                               0.000000
dtype: float64
```

The remaining percentage of missing values in each column

We still have some missing values in **18** columns. For textual columns, we will impute them with '**unknown**', while numerical columns will be filled by **-1**. In this way, we ensure to impute data without losing any information, ensuring clear differentiation between imputed and actual values.

### Imputation for textual data:

```
columns_text_unknown = ['purchase_places', 'allergens', 'stores', 'packaging', 'labels',
df[columns_text_unknown]=df[columns_text_unknown].fillna('unknown')
```

**text_imputation.py** hosted with ❤️ by **GitHub** **view raw**

## Imputation for numerical data:

```
columns_num_unknown= ['ingredients_that_may_be_from_palm_oil_n','additives_n','ingredien
df[columns_num_unknown]=df[columns_num_unknown].fillna('-1')
```

**num_imputation.py** hosted with ❤️ by **GitHub** **view raw**

```
df.isnull().sum()
```

**sum_null.py** hosted with ❤️ by **GitHub** **view raw**

| | |
|---|---|
| code | 0 |
| creator | 0 |
| created_t | 0 |
| last_modified_t | 0 |
| product_name | 0 |
| quantity | 0 |
| packaging | 0 |
| brands | 0 |
| categories | 0 |
| compared_to_category | 0 |
| labels | 0 |
| purchase_places | 0 |
| stores | 0 |
| countries | 0 |
| ingredients_text | 0 |
| allergens | 0 |
| serving_size | 0 |
| nutriments | 0 |
| additives_n | 0 |
| additives_tags | 0 |
| ingredients_from_palm_oil_n | 0 |
| ingredients_from_palm_oil_tags | 0 |
| ingredients_that_may_be_from_palm_oil_n | 0 |
| ingredients_that_may_be_from_palm_oil_tags | 0 |
| nutrition_grades | 0 |
| pnns_groups_1 | 0 |
| pnns_groups_2 | 0 |
| states | 0 |
| ecoscore_score | 0 |
| ecoscore_grade | 0 |

dtype: int64

No missing values

Finally! No missing values. Now, we can dive into our analysis.

## Analysis of the Open Food Facts Dataset

To get a clear idea of the size of the data we are dealing with, we will count the total number of food products in our dataset.

```
1  total_products = df.shape[0]
2  print(f"Total number of food products in the dataset: {total_products}")
```

**total_foods.py** hosted with ❤️ by **GitHub** **view raw**

```
Total number of food products in the dataset: 1277375
```

Number of foods in our dataset

A significant amount of data for analysis is present in the dataset including

**1,227,375** food items in total.

For now, we will use this large dataset to gain insight such as the following:

- How eco-scores are presented and available for different categories and products.
- An in-depth look at eco-score factors, such as packaging information.
- A comparison of food products with and without eco-scores to find ways to make environmental impact scores better.

### Eco-score Availability Analysis

To figure out how food products affect the environment, we need to know how the eco-scores in our dataset are distributed.

Next, is to find out how many products have an assigned eco-score and how many do not.

```
# Calculate the total number of products with an assigned Eco-Score
products_with_eco_score = (df['ecoscore_score'] != -1).sum()

# Calculate the total number of products without an assigned Eco-Score
products_without_eco_score = (df['ecoscore_score'] == -1).sum()

# Calculate the percentage of products with an assigned Eco-Score
total_products = len(df)
eco_score_percentage = (products_with_eco_score / total_products) * 100

# Calculate the percentage of products without  an assigned Eco-Score
no_eco_score_percentage = (products_without_eco_score / total_products) * 100

print(f"Total number of products with an Eco-Score: {products_with_eco_score}")
print(f"Total number of products without an Eco-Score: {products_without_eco_score}")
print(f"Percentage of products with an Eco-Score: {eco_score_percentage:.2f}%")
print(f"Percentage of products without an Eco-Score: {no_eco_score_percentage:.2f}%")
```

**products_ecoscore.py** hosted with ❤️ by **GitHub** **view raw**

```
Total number of products with an Eco-Score: 462095
Total number of products without an Eco-Score: 815280
Percentage of products with an Eco-Score: 36.18%
Percentage of products without an Eco-Score: 63.82%
```

Total number and percentage of products with and without Eco-Score

The analysis of the dataset gave us very interesting information about how Eco-Scores are distributed among food products. A total of **462,095** products have an Eco-Score. A big part of the dataset (**815,280**), did not have an Eco-Score, showing that we need to do a better job of evaluating and sharing how food products affect the environment.

**Total Number of Products With and Without Packaging:**

We will take a close look at the `packaging` column to see how often our dataset contains packaging data.

```python
# Count products with and without packaging data
with_packaging = df[df['packaging'] != 'unknown'].shape[0]
without_packaging = df[df['packaging'] == 'unknown'].shape[0]
packaging_percentage= (with_packaging / total_products) * 100
no_packaging_percentage= (without_packaging / total_products) * 100

print(f"Products with packaging data: {with_packaging}")
print(f"Products without packaging data: {without_packaging}")
print(f"Percentage of products with Packaging: {packaging_percentage:.2f}%")
print(f"Percentage of products without Packaging: {no_packaging_percentage:.2f}%")
```

**Packaging.py** hosted with ❤️ by **GitHub** **view raw**

```
Products with packaging data: 257285
Products without packaging data: 1020090
Percentage of products with Packaging: 20.14%
Percentage of products without Packaging: 79.86%
```

Output from the code above about packaging data

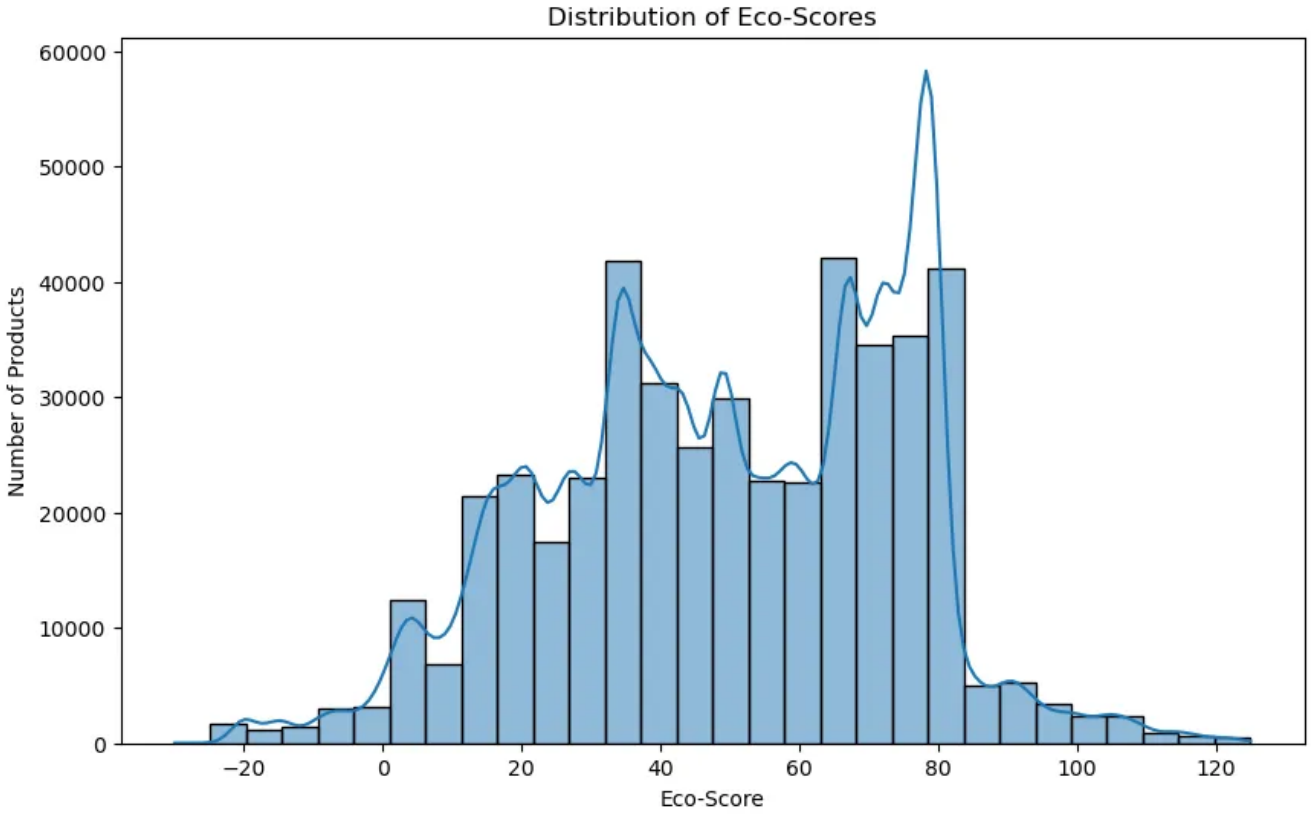
From all products, we found that **257,285** have information about their packaging. Surprisingly, **1,020,090** products don't have any information on the packaging, and

that's approximately **80%** of the missing data!

### Histogram of Eco-Scores:

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df[df['ecoscore_score'] != -1], x='ecoscore_score', bins=30, kde = Tru
plt.title('Distribution of Eco-Scores')
plt.xlabel('Eco-Score')
plt.ylabel('Number of Products')
plt.show()
```

**dist_eco.py** hosted with ❤️ by **GitHub** **view raw**



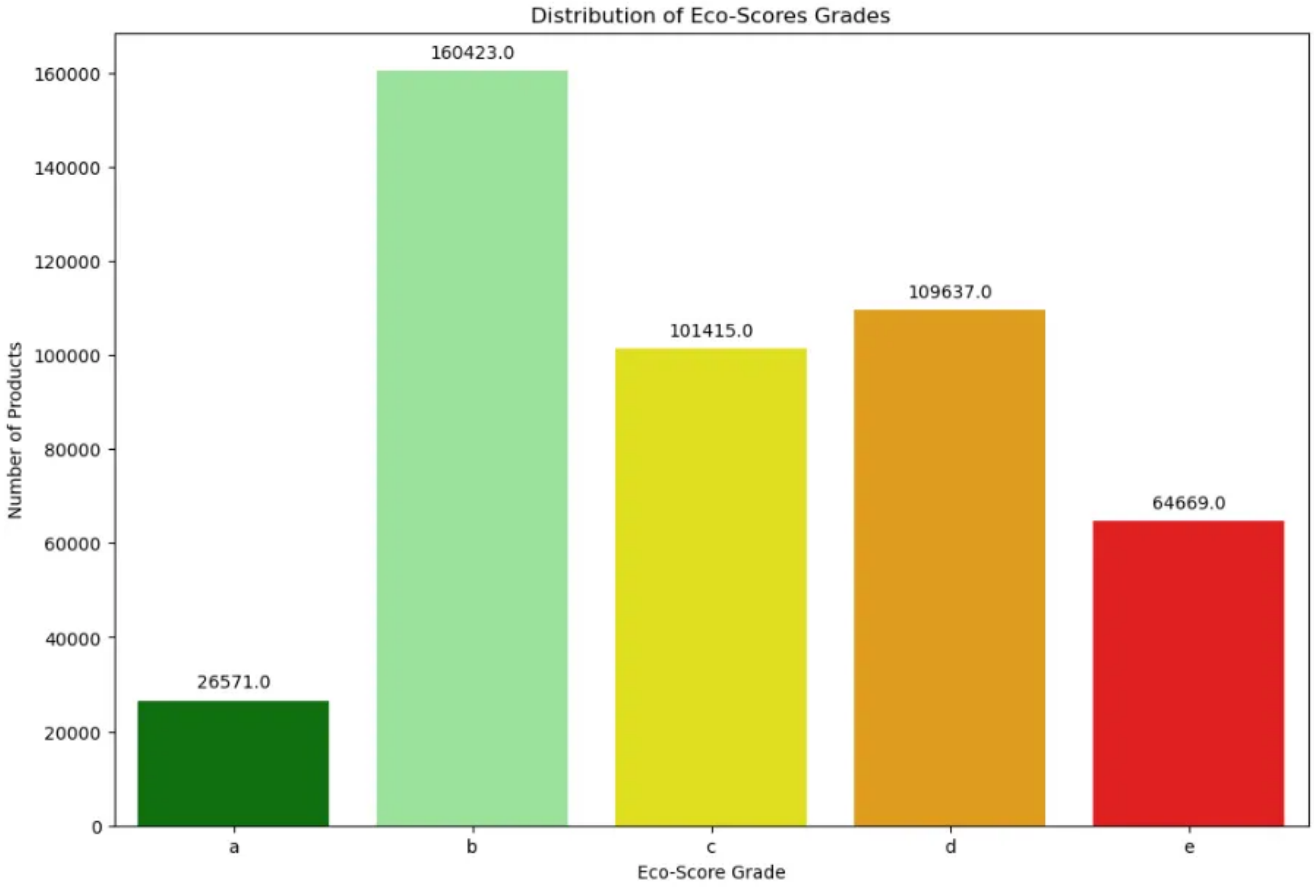

Distribution of Eco-Scores

We can see above a number of peaks in the lower score ranges and the middle range between **60** and **80.**

It's interesting that some products have negative scores, which have a bad effect on the environment or maybe there are problems with scoring algorithms that cause them to get that rating.

Scores above **100** are not very common, but they could indicate some special products or outliers.

**Distribution of Eco-Score Grades:**

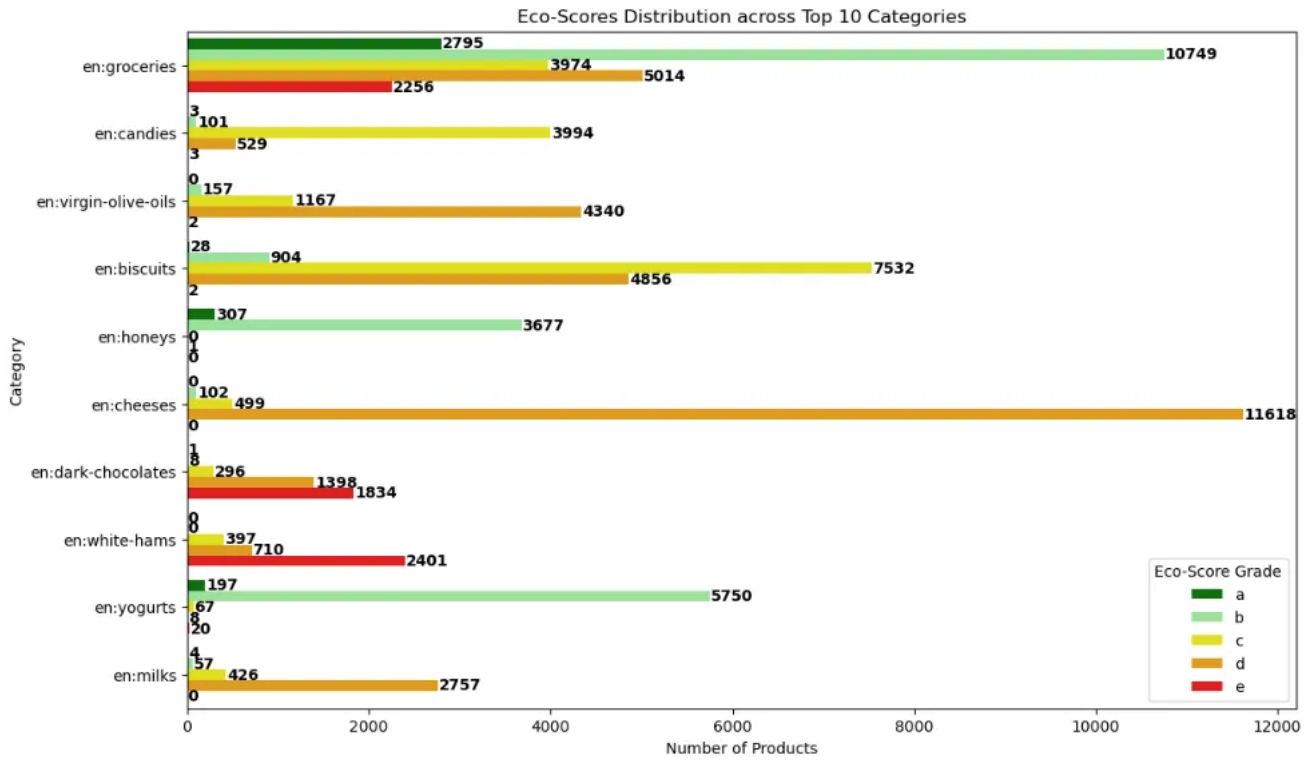



Distribution of Eco-Score Grades

From the bar chat:

- There are **160,423** products in category **'b'** which is the highest of any group. It seems a lot of products have a pretty low impact on the environment, but do not reach the top rating.
- The next biggest group is **'c'** with **101,415** products which has a modest impact on the environment.
- **'D'** and **'e'** come with **109,637 and 64,669** products respectively. It's possible that the grade **'e'** could harm the environment more than the other grades.
- In grade **'a'**, very few products(**26,571**) meet the best or the preferred choice. There is a clear opportunity to stand out in this category and encourage producers to strive for higher Eco-Scores.

## Eco-Score Distribution across Categories




Eco_Scores Distribution across Top 10 Categories

Looking at the Eco-Scores for different categories of food is one of the best ways to understand the environmental impact of our food choices.

**Groceries:** A lot of people eat groceries, and they have a lot of items with an **'a'** grade and a leading **10,749** products with a **'b'** grade. On the other hand, a substantial number fall into the less desired **'d'** and **'e'** ratings, encouraging this category to do better.

**Candies:** This category doesn't get many '**a**' or '**e**' grades, which suggests that environmental impact isn't a big deal in this category.

**Virgin Olive Oils:** Olive oils, especially virgin types, have a huge number of products with **'d'** grade. The lack of grade '**a**' shows that there may be problems with environmental sustainability in this category.

**Biscuits:** Have a large number leaning toward **'c'** grade. Even though some products have a moderate effect on the environment, the category could move toward more sustainable practices.

**Honeys:** Stand out with a high count of products receiving a grade **'b'.** We don't see products in grades '**c**', '**d**', or '**e**', as companies in this area are moving toward more eco-friendly production.

**Cheeses:** There's an absence of both grades '**a**' and '**e**', and a big lean toward grade '**d**'. Unfortunately, the damage to the environment is serious in this category.

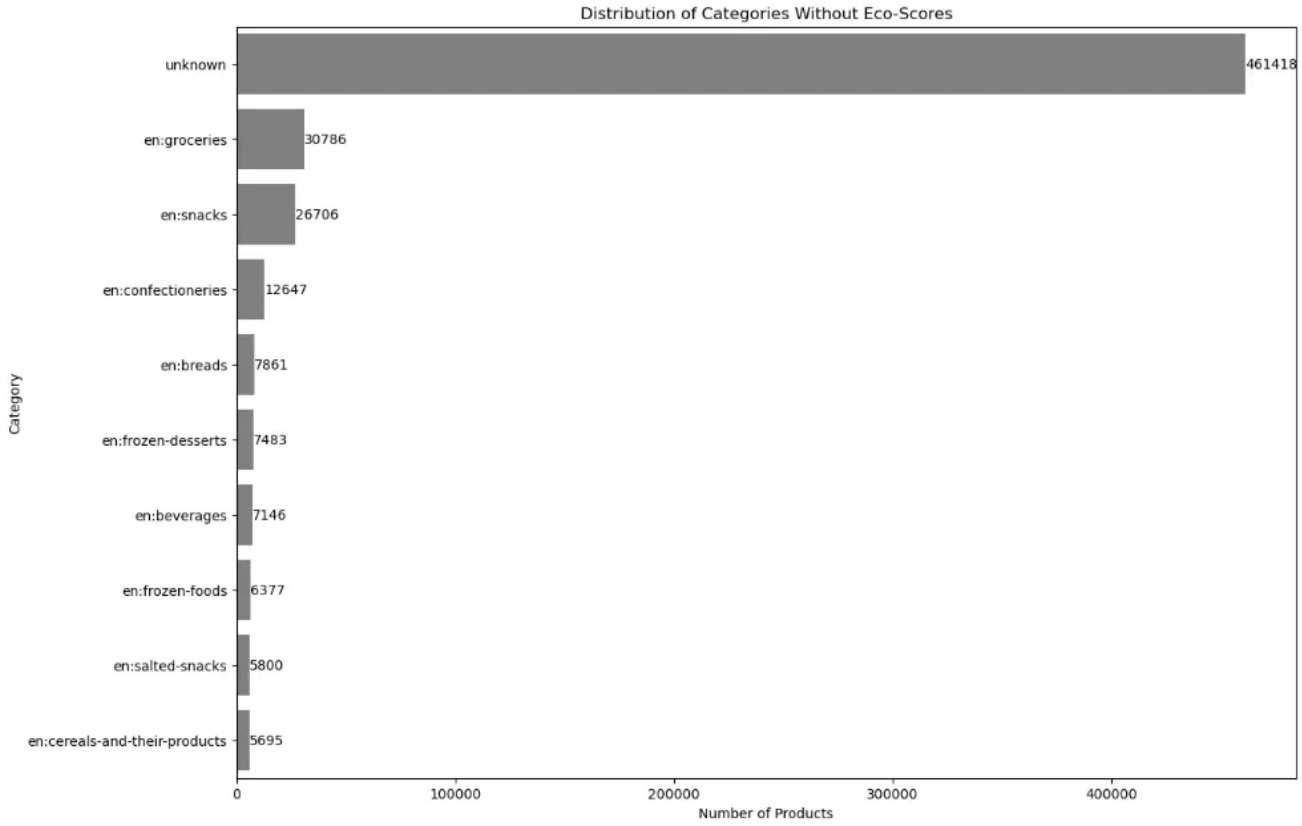
**Dark chocolates:** The distribution of dark chocolates is quite fascinating, with many products receiving an '**e**' grade. Companies in this category need an urgent requirement for more environmentally responsible production of dark chocolates.

**White Hams:** The meat industry needs to do more to protect the environment since most of the products fall into grade **'e'.**

**Yogurts:** Present a positive trend, with most of them getting a grade '**b**'; however, the presence of grade '**e**' shows that more work needs to be done.

**Milks:** Show a majority of products with a '**d**' grade and a very small fraction reaching the '**a**' grade. We can say that the Milks category has a moderate to high environmental impact and could use more eco-friendly practices.

**Categories without Eco-Scores**




Distribution of Categories Without Eco-Scores

In the visualisation above, we can see how many products in different groups don't have an eco-score. The '**unknown**' category is the largest with **461,418** products whose eco-score status is not assigned.

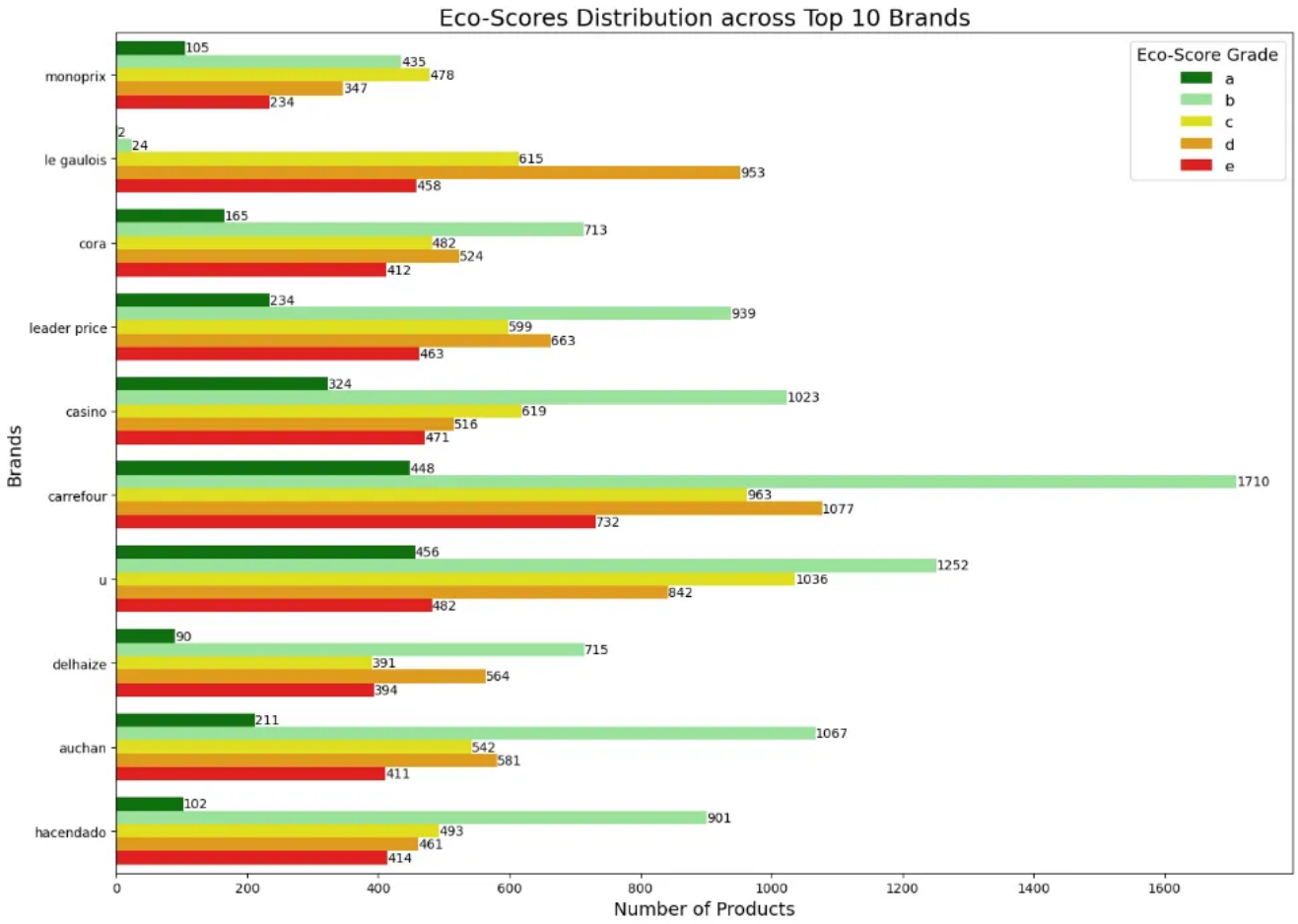
For specific categories, '**groceries**' has over **30,000** without an eco-score, followed closely by '**snacks**' and '**confectioneries**', each with over **25,000** and **12,000** products respectively. Many people use these products every day, and the fact that there is no eco-score data suggests a missed chance to make more informed decisions about sustainability.

'**Breads**', '**frozen desserts**', '**beverages**', '**frozen foods**', '**salted snacks**' and '**cereals and their products**' all present a considerable number of products without eco-scores.

It's clear from the bar plot that eco-scores haven't covered every part of the food business yet. Filling this gap could give consumers the knowledge they need to

choose products that are better for the environment and may also make companies think more about how their products will last.

### Eco-Score Distribution across Brands




Eco-Scores Distribution across top 10 Brands

After checking the eco-score distribution across the top 10 categories, now we'll examine the eco-score distribution across the top 10 brands. The bar chart reveals that some brands show a commitment to sustainability, but there's always a clear room for improvement.

'**Monoprix**' has a good range of grades, most of them are '**b**' and '**c**'. Le '**Gaulois**' and '**Cora**' show a stronger tendency toward the middle to lower eco-grades.

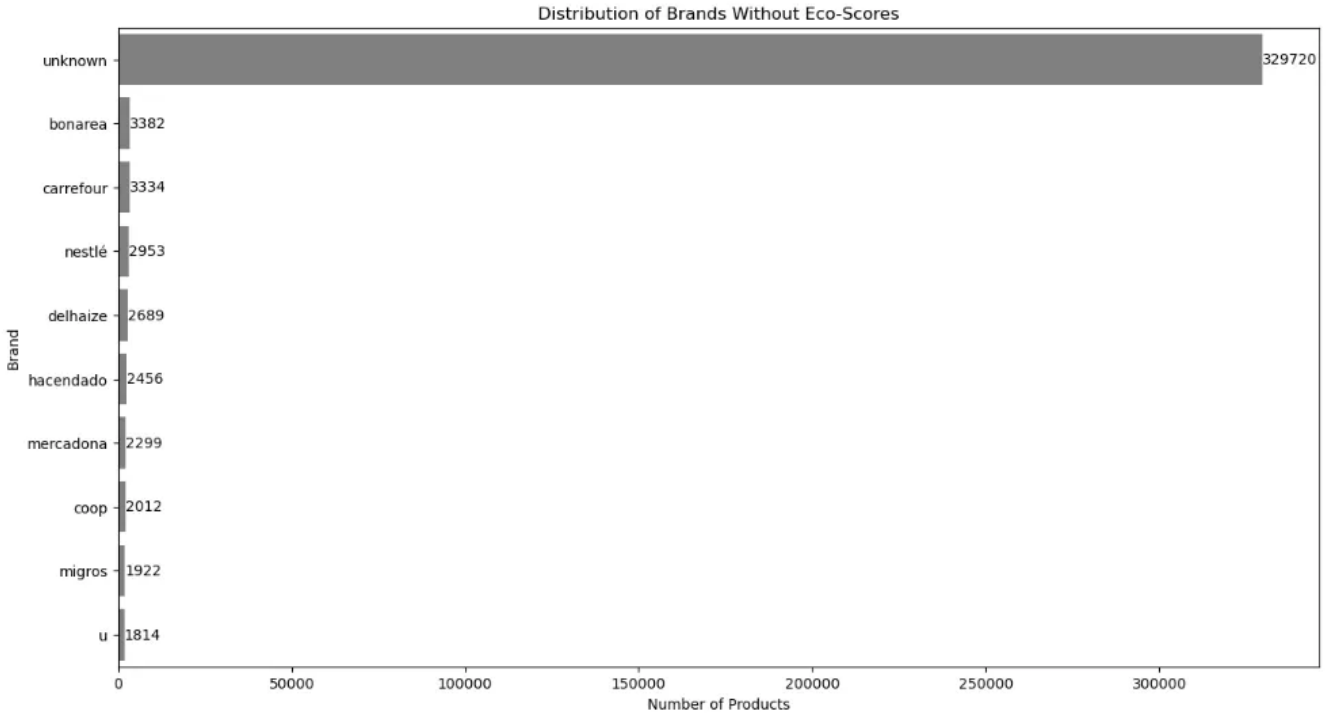
'**Leader Price**' and '**Casino**' are doing well with '**b**' grades and a significant presence in '**c**' and '**d**'. On the other hand, '**Carrefour**' stands out as it has the most products in the '**b**' grade. However, it also faces the challenge of a large number of products in the lower eco-grades.

The '**u**' brand shows a trend toward better sustainability, but the numbers indicate a

balanced distribution that includes lower grades. **'Delhaize'** and **'Auchan'** are mostly in the **'b'** and **'c'** grades, while the **'a'** grade is still an area of potential growth.

Finally, **'Hacendado'** has quite a similar score from **'c'** to **'e'.** The brand has a different effect on the environment and shows a potential for more eco-friendly products.

**Brands without Eco-Scores**




Distribution of Brands Without Eco-Scores

The bar chart shows the distribution of brands with products without eco-scores. A huge number of products marked as **'unknown'** are those without a known brand. **'Carrefour'**, **'Nestlé'**, and **'Delhaize'** are some of the most well-known names, with thousands of products lacking an eco-score.

## Exploring 'not-applicable' Eco-Score Grades

One thing that caught my attention while looking at the dataset, was the **'not-applicable'** field in the `eco-score` grade column.

```
Out[33]:
array(['unknown', 'unknown', 'unknown', ..., 'unknown', 'not-applicable',
       'unknown'], dtype=object)
```

List of values in the ecoscore_grade column

Let’s first find out the total number of products with this field and its proportion in the dataset.

```
Total number of products: 1277375
Number of 'not-applicable' products: 22013
Proportion of 'not-applicable' products: 1.72%
```

Total number of products with ‘not-applicable’ grade

**22,013** out of **1,277,375** products have a **‘not-applicable’** grade.

Next, we will see the top 10 categories with the most **'not-applicable'** products as well as their brands.

```
beverages, carbonated drinks, sodas                                                                                    4
069
beverages, waters                                                                                                      2
856
boissons, eaux, eaux de sources, eaux minérales, eaux minérales naturelles
313
bebidas, aguas, aguas de manantial, aguas minerales, aguas minerales naturales
264
boissons, eaux, eaux de sources, eaux minérales
257
boissons, boissons gazeuses, sodas, sodas au cola, boissons avec sucre ajouté
243
boissons, boissons énergisantes
230
boissons, eaux, eaux de sources
219
boissons, boissons gazeuses, sodas, boissons avec sucre ajouté
215
boissons, boissons gazeuses, boissons édulcorées, sodas, boissons light, sodas au cola, sodas light, sodas au cola light
212
Name: categories, dtype: int64
```

Top 10 categories with the most ‘not-applicable’ products

```
unknown                  4528
coca-cola                 537
schweppes                 241
pepsi                     214
carrefour                 173
fanta                     169
u                         139
coca cola                 125
red bull                  125
san pellegrino, nestlé    120
Name: brands, dtype: int64
```

Top brands with ‘not-applicable’ grade

It was interesting to see how products with **‘not-applicable’** Eco-Scores were grouped within certain categories and brands.

The most frequent categories associated with **‘not-applicable’** Eco-Scores are related to beverages. There are a lot of carbonated drinks and beers at the top, followed by different kinds of water (mineral, spring, etc…).

Based on these results, it seems that the Eco-Score grades might be less applicable or challenging to apply for beverage products, possibly due to the nature of their production or packaging.

Furthermore, the brand analysis supports the category findings with popular brands like **Coca-Cola**, **Schweppes**, **Pepsi**, **Fanta** and **Red Bull** being among the products with **‘not-applicable’** scores.

## Conclusion

Clarifying why beverages often get **‘no-applicable’** scores is really important for both stakeholders and consumers, as it could help improve environmental impact assessments in this area.

The best thing to do is to get in touch with organisations responsible for Eco-Score assessments, as they can tell about the specific limitations or exclusions that apply to beverage products. Also, more research into the environmental impact of beverage production and packaging could show ways to improve data and make assessment practices more inclusive.

## Thanks for reading!

That’s all for now! I hope you find this post useful.

Sustainability | Data Analysis | Food Technology | Nutrition

Follow

## Written by Achraf Elkhanjari

1 Follower · 4 Following

Graduate Data Scientist

## Responses (1)

What are your thoughts?

Respond

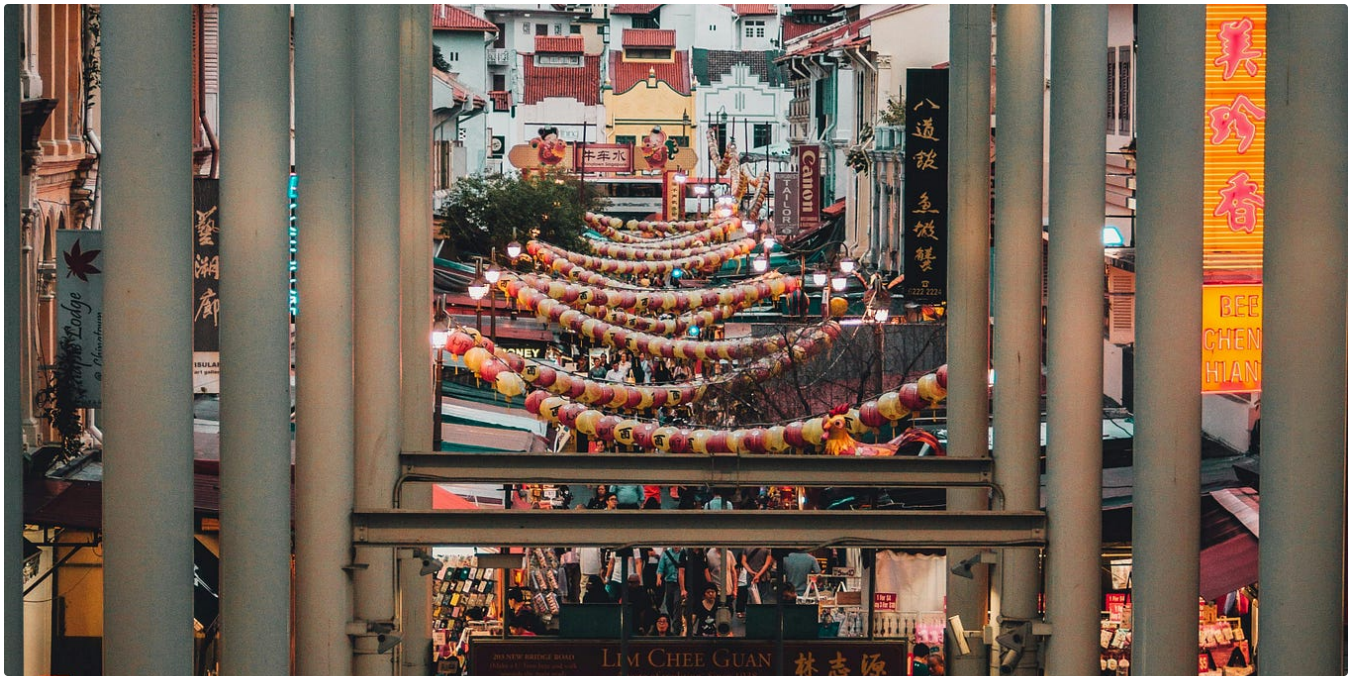
Dynamoreznik
18 days ago

man this was great, i have a question tho, what type of models can be used to create an ingredient scanner, which classifies products based on this dataset

1 1 reply

Reply

## Recommended from Medium




Mark Shrime, MD, PhD

### The dumbest decision I ever made (and the Nobel Prize that explains it)

Decision science, Family Guy, and why the “safe” choice is often the riskiest.

Nov 21 7.5K 172

Adaobi Adibe

## How to do things if you're not that smart and don't have any talent

This is a blog post aimed at people who want to do important work or make meaningful contributions to work, but feel they aren't that smart...

6d ago 4.9K 137

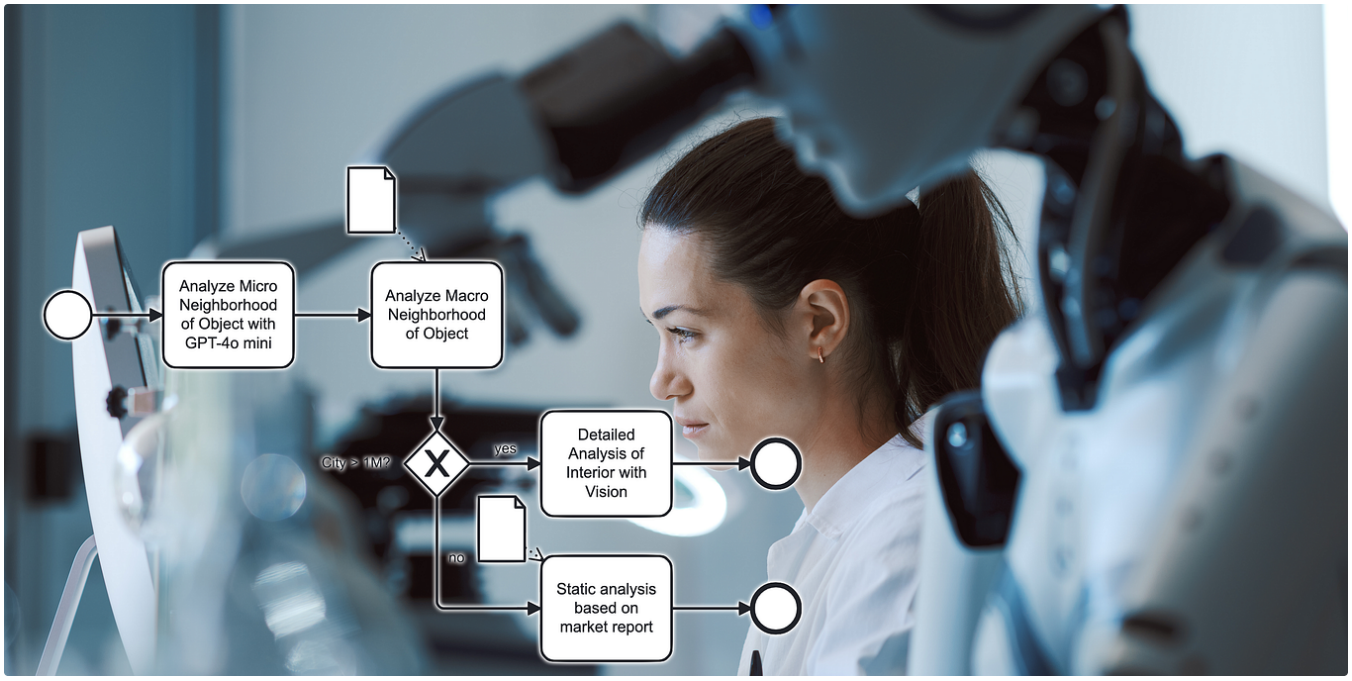
### Lists

**Business**

41 stories · 162 saves

**Medium's Huge List of Publications Accepting Submissions**

377 stories · 4050 saves




tds In Towards Data Science by Dr. Marcel Müller

## Why Internal Company Chatbots Fail and How to Use Generative AI in Enterprise with Impact

Start with the problem and not with the solution

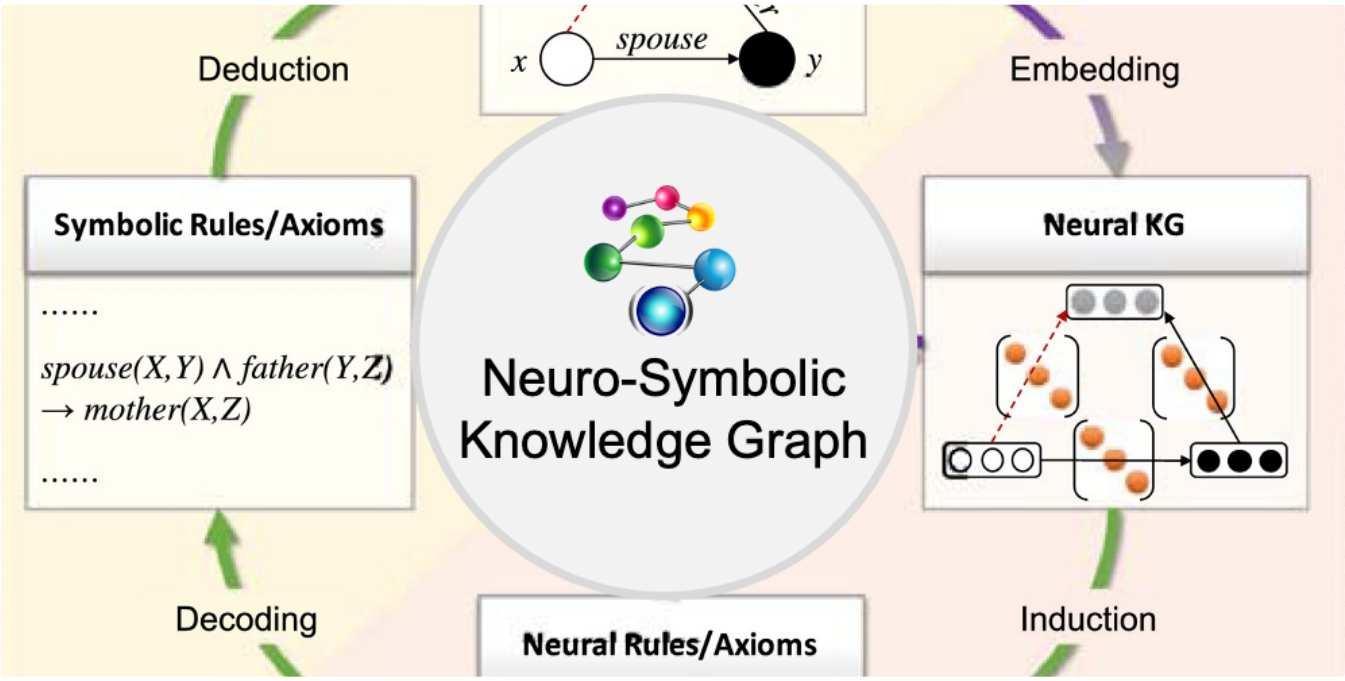
3d ago 614 11

In Geowox by Marco Giardina

## 2024 Q2 Irish Housing Market Analysis: Emerging Trends and Insights

Exploring Price Fluctuations, Sales Dynamics, and the Rising Influence of Energy Efficiency in Ireland's Property Market

Jul 19




In Towards AI by Shashwat (Shane) Gupta

## [AI] Neurosymbolic AI—a microthesis

Following is a microthesis on Neurosymbolic AI, highlighting investment areas, interesting companies and the invest opportunities.

Oct 26 48


In Magnetic Newsletter Pro by George Schneider, M.A.

## Turn $20,000 in Savings into $1000 a Month in Passive Income

It's easier than you might think to generate $1,000 a month from just $20,000.

Nov 18 1.1K 173

See more recommendations