PepsiCo Data science challenge 2022

By Zahra Adahman

Rutgers University, PhD student

October 21st, 2022.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
ls
```

```
 Volume in drive C is OS
 Volume Serial Number is 9E6A-A1B0

 Directory of C:\Users\Zahra\Documents\PepsiCo2022

10/21/2022  11:02 PM    <DIR>          .
10/21/2022  11:02 PM    <DIR>          ..
10/21/2022  10:27 AM    <DIR>          .ipynb_checkpoints
10/21/2022  10:27 AM        33,024,998 Covexper_classfiication.ipynb
10/21/2022  11:02 PM         1,032,670 PepsiCo_2022_Zahra.ipynb
10/12/2022  03:18 PM           744,598 PepsiCo_Zahra.ipynb
10/21/2022  10:44 PM            84,848 Pepsicotest.png
10/21/2022  03:48 PM            14,495 Pepsicotrain.png
10/21/2022  04:24 PM            73,817 Pepsicotrain2.png
10/21/2022  10:50 PM             1,911 TestDecisionTree
10/21/2022  10:50 PM            27,505 TestDecisionTree.pdf
10/12/2022  03:14 PM        14,959,459 TestingData.csv
10/12/2022  03:05 PM         8,249,868 TestingData.xlsx
10/12/2022  03:10 PM        40,787,076 TrainingData.csv
10/12/2022  03:09 PM        22,965,781 TrainingData.xlsx
              12 File(s)    121,967,026 bytes
               3 Dir(s)  71,818,252,288 bytes free
```

1. Dataexploration

Opening traininng data

Data Extraction and Cleaning.

In [3]:
```
df = pd.read_csv("TrainingData.csv")
```

```
C:\Users\Zahra\AppData\Local\Temp\ipykernel_17468\2285231466.py:1: DtypeWarning: Colu
mns (6) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("TrainingData.csv")
```

In [4]:
```
df.head(5)
```

Out[4]:

	Year	Quarter	Country	Local Material Description	Product ID	Local Brand Name	Local Sub-Brand Name	Product Type	Global Brand Name	Standard Brand Name	..
0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	BRAND1	BRAND1	FOOD	BRAND7	BRAND1	..
1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	BRAND2	BRAND2	FOOD	BRAND2	BRAND2	..
2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	BRAND2	BRAND2	FOOD	BRAND2	BRAND2	..
3	2020	Q1	Egypt	BRAND1 180G C AND O 6P	22200904	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1	..
4	2020	Q1	Egypt	BRAND1 180G C AND L 6P	22200905	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1	..

5 rows × 200 columns

In [5]:
```
df.dtypes
```

Out[5]:
```
Year                            int64
Quarter                        object
Country                        object
Local Material Description     object
Product ID                      int64
                                ...
Monosaccharides                float64
Neotame                        float64
Nitrogen                       float64
Phenylalanine + Tyrosine       float64
Polydextrose                   float64
Length: 200, dtype: object
```

In [6]:
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(df.dtypes)
```

Year	int64
Quarter	object
Country	object
Local Material Description	object
Product ID	int64
Local Brand Name	object
Local Sub-Brand Name	object
Product Type	object
Global Brand Name	object
Standard Brand Name	object
Standard Sub-Brand Name	object
Category Name	object
Sub-Category Name	object
Segment Name	object
Sub-Segment Name	object
Volume Units	object
Flavor	object
Standard Flavor Name	object
Kilojoules	float64
Total Calories	float64
Total Fat	float64
Calories from Fat	float64
Calories From Saturated Fat	float64
Calories - Canada	float64
Saturated Fat	float64
Saturated Fat per 100 kcals	float64
Trans Fatty Acids	float64
Monounsaturated Fats	float64
Polyunsaturated Fat	float64
Cholesterol	float64
Omega 3 Fatty Acids	float64
Omega 6 Fatty Acids	float64
Linolenic Acid	float64
DHA	float64
Carbohydrate Total	float64
Carbohydrates using Difference Method	float64
Roll-Up Of Component Carbohydrates	float64
Other Carbohydrates/Starch	float64
Sugars	float64
Added Sugars	float64
Calories from Added Sugars per 355ml	float64
% of Energy from Added Sugars	float64
Sucrose	float64
Fructose	float64
Glucose	float64
Sodium	float64
Sodium per kcal	float64
Sodium Excluding Contribution From Plant Water Ingredient	float64
Sodium (Historical Value) MG	float64
Sodium (Including Canada Water)	float64
Sodium (Including US Water)	float64
Sodium From Components	float64
Sodium In Water (Canada)	float64
Sodium In Water (US)	float64
Salt	float64
Protein	float64
Dietary Fiber	float64
Dietary Fiber Not Allowable By US FDA	float64
Total Dietary Fiber - US FDA	float64
Insoluble Fiber	float64

Insoluble Fiber - US FDA	float64
Insoluble Fiber Not Allowable By US FDA	float64
Soluble Fiber	float64
Soluble Fiber - US FDA	float64
Soluble Fiber Not Allowable By US FDA	float64
Wheat Bran Fiber	float64
Oat Content	float64
Gluten	float64
Beta Glucan	float64
Whole Grains	float64
Total Grains	float64
Fruits - Solids	float64
Fruits - Liquids	float64
Vegetables - Solids	float64
Vegetables - Liquids	float64
Total Dairy Products	float64
Low Fat Dairy	float64
Nuts & Seeds	float64
Legumes	float64
Beta Carotene	float64
Beta Carotene, Prior To Loss Factor	float64
Historical Value: Beta Carotene	float64
Historical Value: Beta Carotene, Prior To Loss Factor	float64
Biotin	float64
Biotin, Prior To Loss Factor	float64
Calcium	float64
Calcium, Prior To Loss Factor	float64
Chloride	float64
Chloride, Prior To Loss Factor	float64
Total Choline	float64
Chromium	float64
Chromium, Prior To Loss Factor	float64
Copper	float64
Copper, Prior To Loss Factor	float64
Folate	float64
Folate, Prior To Loss Factor	float64
Folic Acid	float64
Folic Acid, Prior To Loss Factor	float64
Folic Acid (Synthetic)	float64
Folic Acid (Synthetic), Prior To Loss Factor	float64
Iodine	float64
Iodine, Prior To Loss Factor	float64
Iron	float64
Iron, Prior To Loss Factor	float64
Manganese	float64
Manganese, Prior To Loss Factor	float64
Magnesium	float64
Magnesium, Prior To Loss Factor	float64
Molybdenum	float64
Molybdenum, Prior To Process Loss	float64
Niacin	float64
Niacin Equivalents	float64
Niacin, Prior To Loss Factor	float64
Pantothenic Acid	float64
Pantothenic Acid, Prior To Loss Factor	float64
Phosphorus	float64
Phosphorus, Prior To Loss Factor	float64
Potassium	float64
Riboflavin	float64
Vitamin B2-Riboflavin, Prior To Loss Factor	float64

Selenium	float64
Selenium, Prior To Loss Factor	float64
Thiamin	float64
Vitamin B1-Thiamin, Prior To Loss Factor	float64
Vitamin A	float64
Vitamin A, Prior To Loss Factor	float64
Historical Value: Vitamin A	float64
Historical Value: Vitamin A, Prior To Loss Factor	float64
Vitamin A (RAE)	float64
Vitamin A, Prior To Loss Factor (RAE)	float64
Vitamin B12	float64
Vitamin B12, Prior To Loss Factor	float64
Vitamin B6	float64
Vitamin B6, Prior To Loss Factor	float64
Vitamin C	float64
Vitamin C, Prior To Loss Factor	float64
Vitamin D	float64
Vitamin D, Prior To Loss Factor	float64
Historical Value: Vitamin D	float64
Historical Value: Vitamin D, Prior To Loss Factor	float64
Vitamin E	float64
Vitamin E, Prior To Loss Facto	float64
Historical Value: Vitamin E	float64
Historical Value: Vitamin E (IU), Prior To Loss Factor	float64
Vitamin K	float64
Vitamin K, Prior To Loss Factor	float64
Zinc	float64
Zinc, Prior To Loss Factor	float64
Acesulfame Potassium	float64
Aspartame	float64
Erythritol	float64
Isomalt	float64
Lactitol	float64
Maltitol	float64
Maltitol Syrup	float64
Mannitol	float64
Rebaudioside A	float64
Saccharin	float64
Sorbitol	float64
Sorbitol Syrup	float64
Sucralose	float64
Sorbic Acid	float64
Stearic Acid	float64
Steviol Glycosides	float64
Tagatose	float64
Xylitol	float64
Flavonoids	float64
Hydrogenated Starch Hydrolysales	float64
Inositol	float64
Moisture	float64
Moisture, Pre-Adjusted	float64
Caffeine	float64
Alcohol	float64
Sugar Alcohol	float64
Sugar Alcohol - Canada	float64
Formula Alcohol	float64
Benzoic Acid	float64
Ash	float64
Histidine	float64
Isoleucine	float64

```
Leucine                                 float64
Lysine                                  float64
Methionine + Cystine                    float64
Proline                                 float64
Taurine                                 float64
Threonine                               float64
Tryptophan                              float64
Valine                                  float64
Carotenoid                              float64
Disaccharides                           float64
Flouride                                float64
Glycerol                                float64
Inulin                                  float64
L-Arginine                              float64
L-Carnitine                             float64
Monosaccharides                         float64
Neotame                                 float64
Nitrogen                                float64
Phenylalanine + Tyrosine                float64
Polydextrose                            float64
dtype: object
```

In [7]:
```
df.shape
```

Out[7]:
```
(90347, 200)
```

In [8]:
```
#lets see which features are missing data and how many
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(df.isnull().sum(axis = 0))
```

Year	0
Quarter	0
Country	0
Local Material Description	8
Product ID	0
Local Brand Name	1550
Local Sub-Brand Name	24612
Product Type	0
Global Brand Name	0
Standard Brand Name	0
Standard Sub-Brand Name	0
Category Name	18400
Sub-Category Name	22311
Segment Name	22996
Sub-Segment Name	28442
Volume Units	11510
Flavor	9420
Standard Flavor Name	37291
Kilojoules	36213
Total Calories	9420
Total Fat	10343
Calories from Fat	80588
Calories From Saturated Fat	88419
Calories - Canada	82704
Saturated Fat	9451
Saturated Fat per 100 kcals	9502
Trans Fatty Acids	15706
Monounsaturated Fats	76943
Polyunsaturated Fat	76977
Cholesterol	39077
Omega 3 Fatty Acids	81250
Omega 6 Fatty Acids	81138
Linolenic Acid	84299
DHA	82119
Carbohydrate Total	27503
Carbohydrates using Difference Method	82687
Roll-Up Of Component Carbohydrates	85268
Other Carbohydrates/Starch	81522
Sugars	26334
Added Sugars	12149
Calories from Added Sugars per 355ml	12149
% of Energy from Added Sugars	12200
Sucrose	81718
Fructose	81523
Glucose	81718
Sodium	9460
Sodium per kcal	9511
Sodium Excluding Contribution From Plant Water Ingredient	89674
Sodium (Historical Value) MG	86558
Sodium (Including Canada Water)	85268
Sodium (Including US Water)	84704
Sodium From Components	86558
Sodium In Water (Canada)	82704
Sodium In Water (US)	82687
Salt	63226
Protein	10615
Dietary Fiber	15911
Dietary Fiber Not Allowable By US FDA	82738
Total Dietary Fiber - US FDA	82706
Insoluble Fiber	81705

Insoluble Fiber - US FDA	82706
Insoluble Fiber Not Allowable By US FDA	82751
Soluble Fiber	80367
Soluble Fiber - US FDA	82706
Soluble Fiber Not Allowable By US FDA	82751
Wheat Bran Fiber	84099
Oat Content	82871
Gluten	90347
Beta Glucan	81710
Whole Grains	35554
Total Grains	52281
Fruits - Solids	40863
Fruits - Liquids	35401
Vegetables - Solids	41377
Vegetables - Liquids	41431
Total Dairy Products	44109
Low Fat Dairy	52250
Nuts & Seeds	49878
Legumes	52465
Beta Carotene	87697
Beta Carotene, Prior To Loss Factor	86431
Historical Value: Beta Carotene	84676
Historical Value: Beta Carotene, Prior To Loss Factor	89483
Biotin	80838
Biotin, Prior To Loss Factor	85211
Calcium	35789
Calcium, Prior To Loss Factor	85211
Chloride	81718
Chloride, Prior To Loss Factor	85211
Total Choline	81583
Chromium	81246
Chromium, Prior To Loss Factor	85211
Copper	80821
Copper, Prior To Loss Factor	85211
Folate	57796
Folate, Prior To Loss Factor	85235
Folic Acid	83994
Folic Acid, Prior To Loss Factor	85263
Folic Acid (Synthetic)	81855
Folic Acid (Synthetic), Prior To Loss Factor	86399
Iodine	82609
Iodine, Prior To Loss Factor	85211
Iron	37823
Iron, Prior To Loss Factor	85211
Manganese	80841
Manganese, Prior To Loss Factor	85207
Magnesium	51507
Magnesium, Prior To Loss Factor	85211
Molybdenum	83097
Molybdenum, Prior To Process Loss	85211
Niacin	79739
Niacin Equivalents	81643
Niacin, Prior To Loss Factor	86501
Pantothenic Acid	80723
Pantothenic Acid, Prior To Loss Factor	85211
Phosphorus	79525
Phosphorus, Prior To Loss Factor	85211
Potassium	41347
Riboflavin	79850
Vitamin B2-Riboflavin, Prior To Loss Factor	86372

Selenium	80848
Selenium, Prior To Loss Factor	85211
Thiamin	79518
Vitamin B1-Thiamin, Prior To Loss Factor	85208
Vitamin A	50257
Vitamin A, Prior To Loss Factor	85270
Historical Value: Vitamin A	83123
Historical Value: Vitamin A, Prior To Loss Factor	89479
Vitamin A (RAE)	83971
Vitamin A, Prior To Loss Factor (RAE)	86392
Vitamin B12	80285
Vitamin B12, Prior To Loss Factor	85211
Vitamin B6	80164
Vitamin B6, Prior To Loss Factor	85211
Vitamin C	46392
Vitamin C, Prior To Loss Factor	85211
Vitamin D	43545
Vitamin D, Prior To Loss Factor	85211
Historical Value: Vitamin D	84440
Historical Value: Vitamin D, Prior To Loss Factor	89524
Vitamin E	52086
Vitamin E, Prior To Loss Facto	85211
Historical Value: Vitamin E	83132
Historical Value: Vitamin E (IU), Prior To Loss Factor	90347
Vitamin K	81551
Vitamin K, Prior To Loss Factor	85211
Zinc	51926
Zinc, Prior To Loss Factor	85211
Acesulfame Potassium	81134
Aspartame	81138
Erythritol	81718
Isomalt	81855
Lactitol	81855
Maltitol	81855
Maltitol Syrup	88473
Mannitol	81855
Rebaudioside A	81138
Saccharin	81138
Sorbitol	81855
Sorbitol Syrup	88473
Sucralose	81138
Sorbic Acid	82228
Stearic Acid	90347
Steviol Glycosides	85268
Tagatose	81718
Xylitol	81855
Flavonoids	81718
Hydrogenated Starch Hydrolysales	82360
Inositol	90347
Moisture	84299
Moisture, Pre-Adjusted	85268
Caffeine	80887
Alcohol	84565
Sugar Alcohol	81529
Sugar Alcohol - Canada	82703
Formula Alcohol	85268
Benzoic Acid	81718
Ash	81718
Histidine	81718
Isoleucine	81704

```
Leucine                                81581
Lysine                                 81718
Methionine + Cystine                   81718
Proline                                81718
Taurine                                81581
Threonine                              81718
Tryptophan                             81718
Valine                                 81688
Carotenoid                             88581
Disaccharides                          88683
Flouride                               89273
Glycerol                               84099
Inulin                                 84099
L-Arginine                             90347
L-Carnitine                            90347
Monosaccharides                        88688
Neotame                                88473
Nitrogen                               87766
Phenylalanine + Tyrosine               82687
Polydextrose                           84099
dtype: int64
```

In [9]:
```
#since I am working with values tthat at not missing focused on added sugar component
#Hence, all missing added sugar data will be dropped.
df1=df.dropna(subset=['Added Sugars', 'Calories from Added Sugars per 355ml','% of Ene
df1.head()
```

Out[9]:

	Year	Quarter	Country	Local Material Description	Product ID	Local Brand Name	Local Sub-Brand Name	Product Type	Global Brand Name	Standard Brand Name
0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	BRAND1	BRAND1	FOOD	BRAND7	BRAND1
1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	BRAND2	BRAND2	FOOD	BRAND2	BRAND2
2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	BRAND2	BRAND2	FOOD	BRAND2	BRAND2
5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1
7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	BRAND3	BRAND3 POPCORN	FOOD	BRAND3	BRAND3

5 rows × 200 columns

In [10]:
```
df1.shape
```

Out[10]:
```
(78147, 200)
```

In [11]:
```
#lets see which features are missing data and how many in the new df1 data frame
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(df1.isnull().sum(axis = 0))
```

Year	0
Quarter	0
Country	0
Local Material Description	4
Product ID	0
Local Brand Name	1403
Local Sub-Brand Name	19988
Product Type	0
Global Brand Name	0
Standard Brand Name	0
Standard Sub-Brand Name	0
Category Name	13252
Sub-Category Name	16878
Segment Name	17365
Sub-Segment Name	22093
Volume Units	2068
Flavor	0
Standard Flavor Name	26610
Kilojoules	24122
Total Calories	0
Total Fat	385
Calories from Fat	68586
Calories From Saturated Fat	76227
Calories - Canada	70531
Saturated Fat	31
Saturated Fat per 100 kcals	31
Trans Fatty Acids	5566
Monounsaturated Fats	64930
Polyunsaturated Fat	64946
Cholesterol	28946
Omega 3 Fatty Acids	69124
Omega 6 Fatty Acids	69020
Linolenic Acid	72160
DHA	69988
Carbohydrate Total	17115
Carbohydrates using Difference Method	70522
Roll-Up Of Component Carbohydrates	73091
Other Carbohydrates/Starch	69357
Sugars	16372
Added Sugars	0
Calories from Added Sugars per 355ml	0
% of Energy from Added Sugars	0
Sucrose	69591
Fructose	69396
Glucose	69591
Sodium	31
Sodium per kcal	31
Sodium Excluding Contribution From Plant Water Ingredient	77493
Sodium (Historical Value) MG	74362
Sodium (Including Canada Water)	73091
Sodium (Including US Water)	72535
Sodium From Components	74362
Sodium In Water (Canada)	70531
Sodium In Water (US)	70522
Salt	51035
Protein	657
Dietary Fiber	5945
Dietary Fiber Not Allowable By US FDA	70542
Total Dietary Fiber - US FDA	70522
Insoluble Fiber	69587

Insoluble Fiber - US FDA	70522
Insoluble Fiber Not Allowable By US FDA	70555
Soluble Fiber	68251
Soluble Fiber - US FDA	70522
Soluble Fiber Not Allowable By US FDA	70555
Wheat Bran Fiber	71903
Oat Content	70706
Gluten	78147
Beta Glucan	69583
Whole Grains	23443
Total Grains	40128
Fruits - Solids	28710
Fruits - Liquids	23545
Vegetables - Solids	29224
Vegetables - Liquids	29278
Total Dairy Products	31956
Low Fat Dairy	40097
Nuts & Seeds	37725
Legumes	40312
Beta Carotene	75531
Beta Carotene, Prior To Loss Factor	74234
Historical Value: Beta Carotene	72507
Historical Value: Beta Carotene, Prior To Loss Factor	77302
Biotin	68711
Biotin, Prior To Loss Factor	73034
Calcium	25658
Calcium, Prior To Loss Factor	73034
Chloride	69591
Chloride, Prior To Loss Factor	73034
Total Choline	69391
Chromium	69111
Chromium, Prior To Loss Factor	73034
Copper	68686
Copper, Prior To Loss Factor	73034
Folate	45684
Folate, Prior To Loss Factor	73039
Folic Acid	71925
Folic Acid, Prior To Loss Factor	73086
Folic Acid (Synthetic)	69663
Folic Acid (Synthetic), Prior To Loss Factor	74202
Iodine	70474
Iodine, Prior To Loss Factor	73034
Iron	27692
Iron, Prior To Loss Factor	73034
Manganese	68706
Manganese, Prior To Loss Factor	73030
Magnesium	39431
Magnesium, Prior To Loss Factor	73034
Molybdenum	70962
Molybdenum, Prior To Process Loss	73034
Niacin	67719
Niacin Equivalents	69478
Niacin, Prior To Loss Factor	74305
Pantothenic Acid	68604
Pantothenic Acid, Prior To Loss Factor	73034
Phosphorus	67401
Phosphorus, Prior To Loss Factor	73034
Potassium	29429
Riboflavin	67822
Vitamin B2-Riboflavin, Prior To Loss Factor	74194

Selenium	68713
Selenium, Prior To Loss Factor	73034
Thiamin	67492
Vitamin B1-Thiamin, Prior To Loss Factor	73031
Vitamin A	40056
Vitamin A, Prior To Loss Factor	73074
Historical Value: Vitamin A	70984
Historical Value: Vitamin A, Prior To Loss Factor	77298
Vitamin A (RAE)	71827
Vitamin A, Prior To Loss Factor (RAE)	74214
Vitamin B12	68158
Vitamin B12, Prior To Loss Factor	73034
Vitamin B6	68050
Vitamin B6, Prior To Loss Factor	73034
Vitamin C	36249
Vitamin C, Prior To Loss Factor	73034
Vitamin D	31514
Vitamin D, Prior To Loss Factor	73034
Historical Value: Vitamin D	72297
Historical Value: Vitamin D, Prior To Loss Factor	77343
Vitamin E	40006
Vitamin E, Prior To Loss Facto	73034
Historical Value: Vitamin E	70993
Historical Value: Vitamin E (IU), Prior To Loss Factor	78147
Vitamin K	69424
Vitamin K, Prior To Loss Factor	73034
Zinc	39881
Zinc, Prior To Loss Factor	73034
Acesulfame Potassium	69009
Aspartame	69013
Erythritol	69591
Isomalt	69663
Lactitol	69663
Maltitol	69663
Maltitol Syrup	76273
Mannitol	69663
Rebaudioside A	69013
Saccharin	69013
Sorbitol	69663
Sorbitol Syrup	76273
Sucralose	69013
Sorbic Acid	70089
Stearic Acid	78147
Steviol Glycosides	73091
Tagatose	69591
Xylitol	69663
Flavonoids	69591
Hydrogenated Starch Hydrolysales	70164
Inositol	78147
Moisture	72160
Moisture, Pre-Adjusted	73091
Caffeine	68760
Alcohol	72411
Sugar Alcohol	69450
Sugar Alcohol - Canada	70530
Formula Alcohol	73091
Benzoic Acid	69591
Ash	69591
Histidine	69591
Isoleucine	69586

```
Leucine                                 69463
Lysine                                  69591
Methionine + Cystine                    69591
Proline                                 69591
Taurine                                 69463
Threonine                               69591
Tryptophan                              69591
Valine                                  69570
Carotenoid                              76389
Disaccharides                           76491
Flouride                                77081
Glycerol                                71903
Inulin                                  71903
L-Arginine                              78147
L-Carnitine                             78147
Monosaccharides                         76496
Neotame                                 76273
Nitrogen                                75578
Phenylalanine + Tyrosine                70522
Polydextrose                            71903
dtype: int64
```

In [12]:
```
#Attempting to minimize features with the least amount of missing valuses,
#by selecting only features with less than 1000 missing values
df1_1000=pd.isnull(df1).sum() < 1000
df1_1000
```

Out[12]:
```
Year                           True
Quarter                        True
Country                        True
Local Material Description     True
Product ID                     True
                              ...
Monosaccharides               False
Neotame                       False
Nitrogen                      False
Phenylalanine + Tyrosine      False
Polydextrose                  False
Length: 200, dtype: bool
```

Selecting only columns with less than 1000 missing values NAN

Hence, we don't want to skew the training data by replacing the misssing data it with 0.

This will help us comb through the data to extrat the most complete data revelant and use the most complete data relevant to the 'added sugar' (Target variable).

In [13]:
```
columns=df1.columns[df1.isnull().sum() < 1000].tolist()
columns
```

Out[13]:
```
['Year',
 'Quarter',
 'Country',
 'Local Material Description',
 'Product ID',
 'Product Type',
 'Global Brand Name',
 'Standard Brand Name',
 'Standard Sub-Brand Name',
 'Flavor',
 'Total Calories',
 'Total Fat',
 'Saturated Fat',
 'Saturated Fat per 100 kcals',
 'Added Sugars',
 'Calories from Added Sugars per 355ml',
 '% of Energy from Added Sugars',
 'Sodium',
 'Sodium per kcal',
 'Protein']
```

In [14]:
```
#df2 is the extracted dataframe with all the complete values for the variable added su
#calories from Added Sugars per 355ml and % of Energy from Added Sugars.
df2=df1[columns]
df2.head()
```

Out[14]:

	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name	Flavor
0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	FOOD	BRAND7	BRAND1	BRAND1	Cheese & Onion
1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	FOOD	BRAND2	BRAND2	BRAND2	Cheese & Spice
2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	FOOD	BRAND2	BRAND2	BRAND2	Sweet Chilli
5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	FOOD	BRAND7	BRAND1	BRAND1 FAMILY	SALT
7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN	Spicy Tomato

In [15]:
```
#reset index
df_addedsugar=df2.reset_index()
df_addedsugar.head()
```

Out[15]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	FOOD	BRAND7	BRAND1	BRAND1
1	1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	FOOD	BRAND2	BRAND2	BRAND2
2	2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	FOOD	BRAND2	BRAND2	BRAND2
3	5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
4	7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

5 rows × 21 columns

In [16]:
```
df_addedsugar.shape
```

Out[16]:
```
(78147, 21)
```

In [17]:
```
df_addedsugar.isnull().sum(axis = 0)
```

Out[17]:
```
index                                  0
Year                                   0
Quarter                                0
Country                                0
Local Material Description             4
Product ID                             0
Product Type                           0
Global Brand Name                      0
Standard Brand Name                    0
Standard Sub-Brand Name                0
Flavor                                 0
Total Calories                         0
Total Fat                            385
Saturated Fat                         31
Saturated Fat per 100 kcals           31
Added Sugars                           0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                31
Sodium per kcal                       31
Protein                              657
dtype: int64
```

Fill the NAN values with 0 in features with less than a 1000 missing values

In [18]:
```
df_addedsugar.fillna(0, inplace=True)
df_addedsugar.isnull().sum(axis = 0)
```

Out[18]:
```
index                                  0
Year                                   0
Quarter                                0
Country                                0
Local Material Description             0
Product ID                             0
Product Type                           0
Global Brand Name                      0
Standard Brand Name                    0
Standard Sub-Brand Name                0
Flavor                                 0
Total Calories                         0
Total Fat                              0
Saturated Fat                          0
Saturated Fat per 100 kcals            0
Added Sugars                           0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 0
Sodium per kcal                        0
Protein                                0
dtype: int64
```

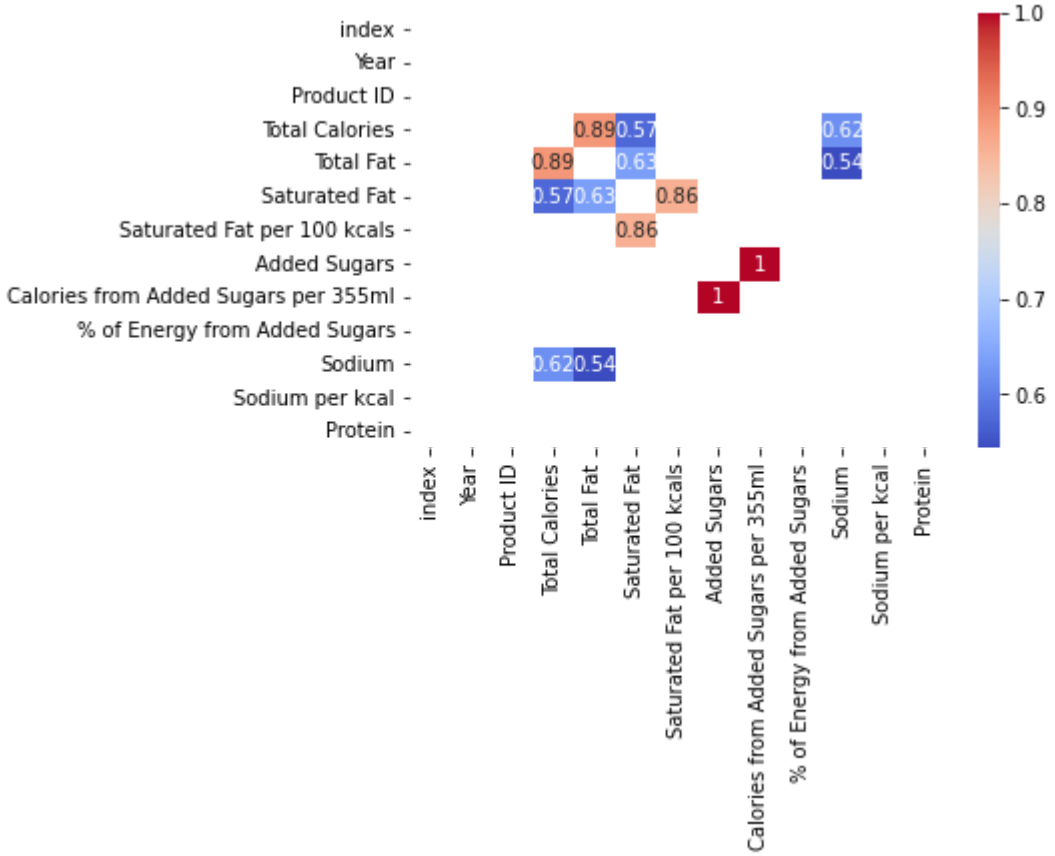
Exploratory Data Analysis

In [19]:
```
df_addedsugar.corr()
```

Out[19]:

	index	Year	Product ID	Total Calories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	Added Sugars	Calories from Added Sugars pe 355m
index	1.000000	NaN	0.019904	0.012934	-0.008886	-0.010452	-0.031009	0.036936	0.03693
Year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Product ID	0.019904	NaN	1.000000	0.241526	0.252295	0.231470	0.215350	-0.094049	-0.09404
Total Calories	0.012934	NaN	0.241526	1.000000	0.888201	0.572918	0.349786	-0.053243	-0.05324
Total Fat	-0.008886	NaN	0.252295	0.888201	1.000000	0.634968	0.433029	-0.213895	-0.21389
Saturated Fat	-0.010452	NaN	0.231470	0.572918	0.634968	1.000000	0.858726	-0.002301	-0.00230
Saturated Fat per 100 kcals	-0.031009	NaN	0.215350	0.349786	0.433029	0.858726	1.000000	0.008716	0.00871
Added Sugars	0.036936	NaN	-0.094049	-0.053243	-0.213895	-0.002301	0.008716	1.000000	1.00000
Calories from Added Sugars per 355ml	0.036936	NaN	-0.094049	-0.053243	-0.213895	-0.002301	0.008716	1.000000	1.00000
% of Energy from Added Sugars	0.008133	NaN	-0.154843	-0.408161	-0.370502	-0.223176	-0.180035	0.323912	0.32391
Sodium	0.022365	NaN	0.097655	0.616103	0.543122	0.318557	0.179962	-0.115473	-0.11547
Sodium per kcal	0.025706	NaN	0.023545	-0.130979	-0.093770	-0.070798	-0.069809	-0.045440	-0.04544
Protein	0.002309	NaN	0.042580	0.399256	0.385427	0.199476	0.153221	-0.049049	-0.04904

In [20]:
```
import seaborn as sns
dfCorr =df_addedsugar.corr()
filteredDf = dfCorr[((dfCorr >= .5) | (dfCorr <= -.5)) & (dfCorr !=1.000)]
sns.heatmap(filteredDf, annot=True, cmap="coolwarm")
plt.figure(figsize=(40,20))
plt.show()
```




```
<Figure size 2880x1440 with 0 Axes>
```

Correlation results:

1. 100 % of Added sugar corealted with added sugars per 355ml,

2. 89% of Total Calories corelates with total fat in the products,

3. 86% of Saturated fat corelates with saturated fat per 100 kcals in the products,

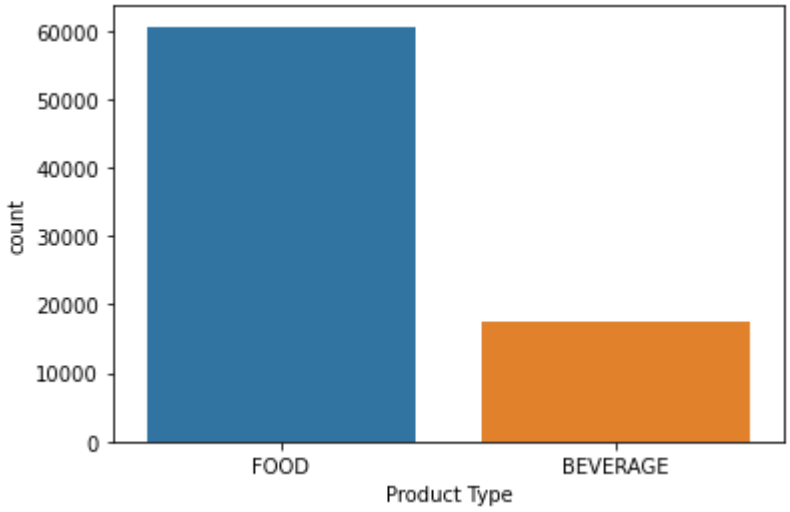
4. 64% of Saturated fat corelates with total fat in the products,

5. 62% of Total Calories corelates with sodium in the products,

6. 57% of Total Calories corelates with saturated fat in the products,

7. 57% of Total fat corelates with sodium in the products,

In [21]:
```
p = sns.countplot(data=df_addedsugar, x="Product Type")
plt.show()
```

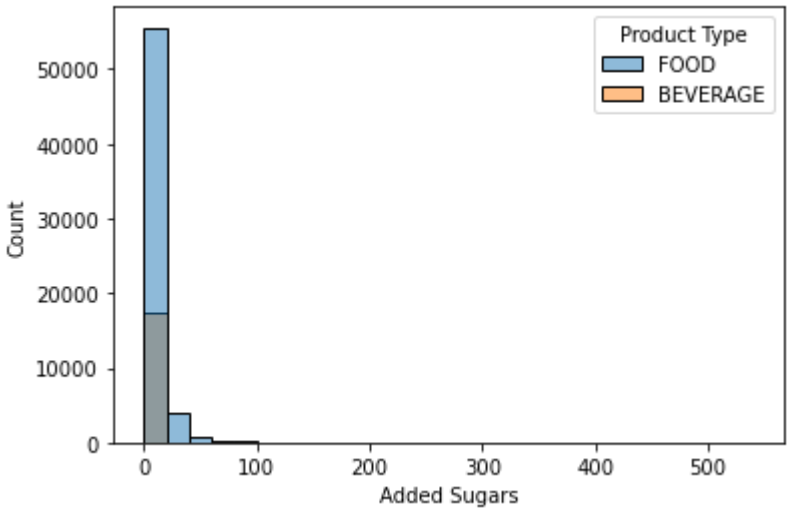


In [22]:
```
df_addedsugar[['Added Sugars']].describe()
```

Out[22]:

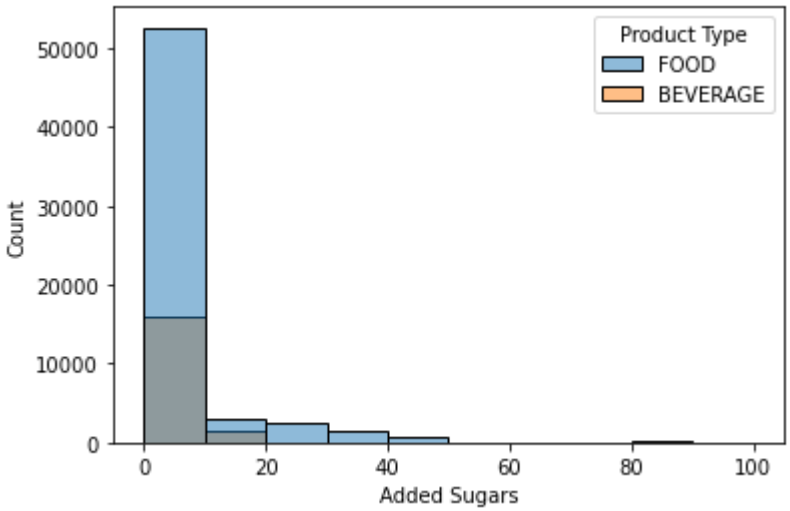
	Added Sugars
count	78147.000000
mean	4.596881
std	10.290242
min	0.000000
25%	0.000000
50%	0.700000
75%	5.236000
max	532.800000

In [23]:
```
p1 = sns.histplot(df_addedsugar, x="Added Sugars", hue="Product Type", binwidth=20, bi

plt.show()
```

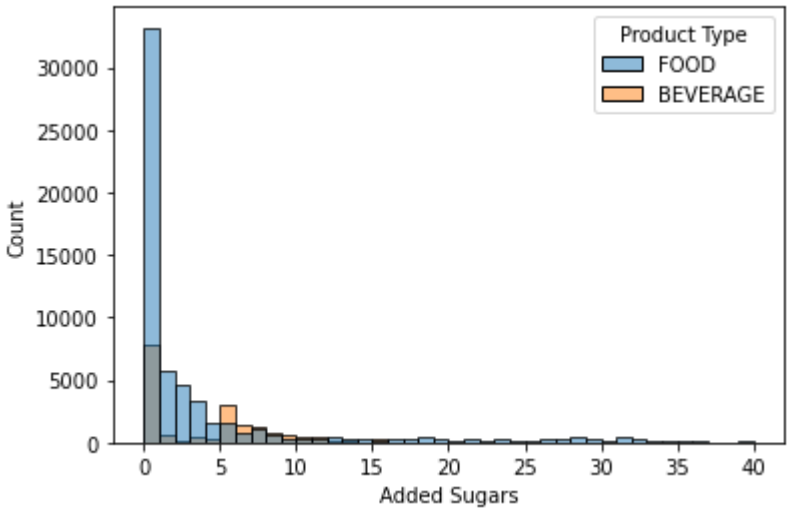


In [24]:
```
p1 = sns.histplot(df_addedsugar, x="Added Sugars", hue="Product Type", binwidth=10, bi
```

```
plt.show()
```



In [25]:
```
p1= sns.histplot(df_addedsugar, x="Added Sugars", hue="Product Type", binwidth=1, binr

plt.show()
```



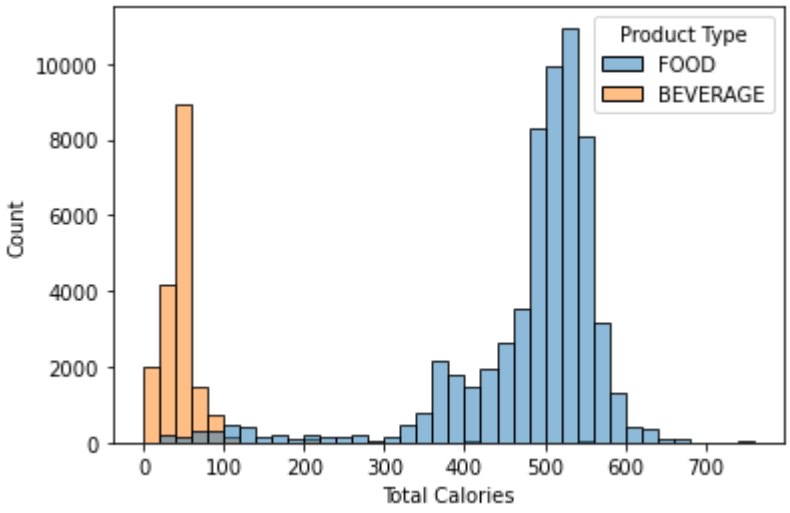
In [26]:
```
df_addedsugar[['Total Calories']].describe()
```

Out[26]:

	Total Calories
count	78147.000000
mean	383.154421
std	203.256977
min	0.199000
25%	161.000000
50%	489.120000
75%	528.000000
max	748.000000

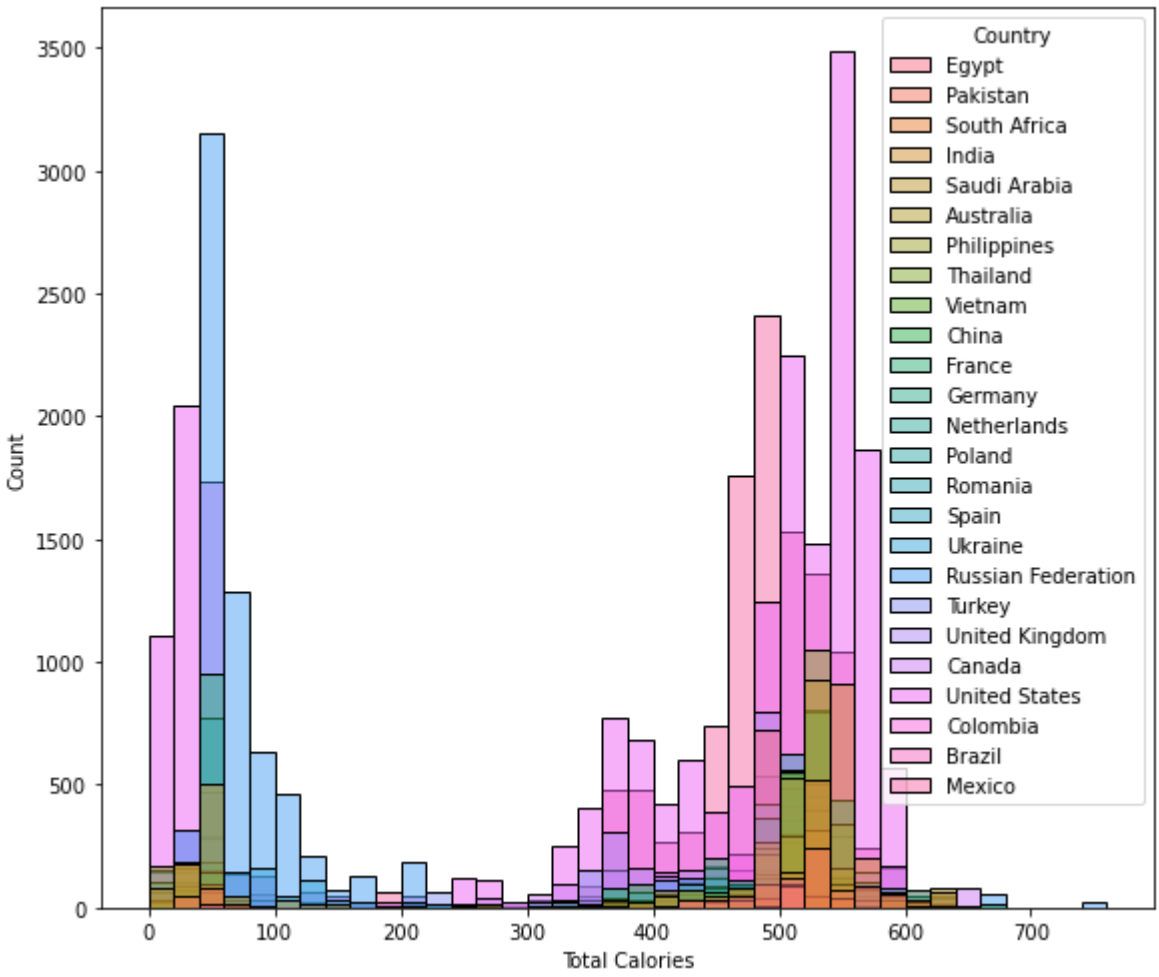
In [27]:
```
p2 = sns.histplot(df_addedsugar, x="Total Calories", hue="Product Type", binwidth=20,
```

```
plt.show()
```



In [ ]:

In [28]:
```
a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
sns.histplot(ax=ax,data=df_addedsugar, x="Total Calories", hue="Country", binwidth=20,
plt.subplots_adjust(right=0.75)
plt.show()
```



There's a high a wide range of values for the added sugars variable. Since this is the case, a new added sugar feature will be created and used as the main target feature according to the FDA we now the DV of added sugar is 50g and 5% DV or 2.5g less is a LOW source of added sugars, and 5.1-19.9% is 2.51-9.99g of sugar is MID source of added sugar and 20% DV or 10.0g or more is a HIGH source of added suga. The new target variable will be added sugars class, LOW, MID and HIGH.

In [29]:
```
df_addedsugar=df_addedsugar.rename(columns={"Added Sugars": "AddedSugars"})
```

In [30]:
```
df_addedsugar.head()
```

Out[30]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	FOOD	BRAND7	BRAND1	BRAND1
1	1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	FOOD	BRAND2	BRAND2	BRAND2
2	2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	FOOD	BRAND2	BRAND2	BRAND2
3	5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
4	7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

5 rows × 21 columns

In [31]:
```
filter_method = lambda AddedSugars: 'low' if AddedSugars <= 2.5 else 'mid' if (AddedSu
```

In [32]:
```
df_addedsugar['AddedSugarClass'] = df_addedsugar['AddedSugars'].apply(filter_method)
```

In [33]:
```
df_addedsugar.head()
```

Out[33]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	FOOD	BRAND7	BRAND1	BRAND1
1	1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	FOOD	BRAND2	BRAND2	BRAND2
2	2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	FOOD	BRAND2	BRAND2	BRAND2
3	5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
4	7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

5 rows × 22 columns

In [34]:
```
#making sure no added sugar class cell/value is empty/NAN
df_addedsugar.isnull().sum(axis = 0)
```

Out[34]:
```
index                                  0
Year                                   0
Quarter                                0
Country                                0
Local Material Description             0
Product ID                             0
Product Type                           0
Global Brand Name                      0
Standard Brand Name                    0
Standard Sub-Brand Name                0
Flavor                                 0
Total Calories                         0
Total Fat                              0
Saturated Fat                          0
Saturated Fat per 100 kcals            0
AddedSugars                            0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 0
Sodium per kcal                        0
Protein                                0
AddedSugarClass                        0
dtype: int64
```

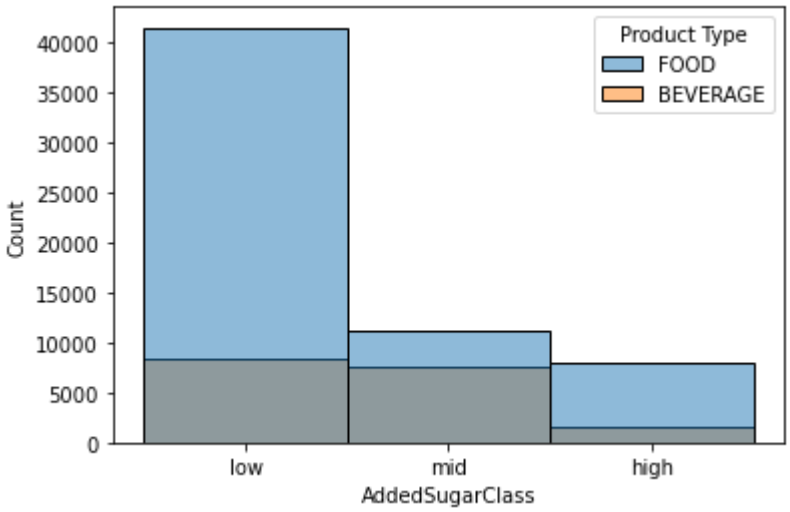
In [35]:
```
#good
```

In [36]:
```
df_addedsugar['AddedSugarClass']
```

Out[36]:
```
0         low
1         mid
2         mid
3         low
4         mid
         ...
78142    high
78143    high
78144    high
78145    high
78146    high
Name: AddedSugarClass, Length: 78147, dtype: object
```

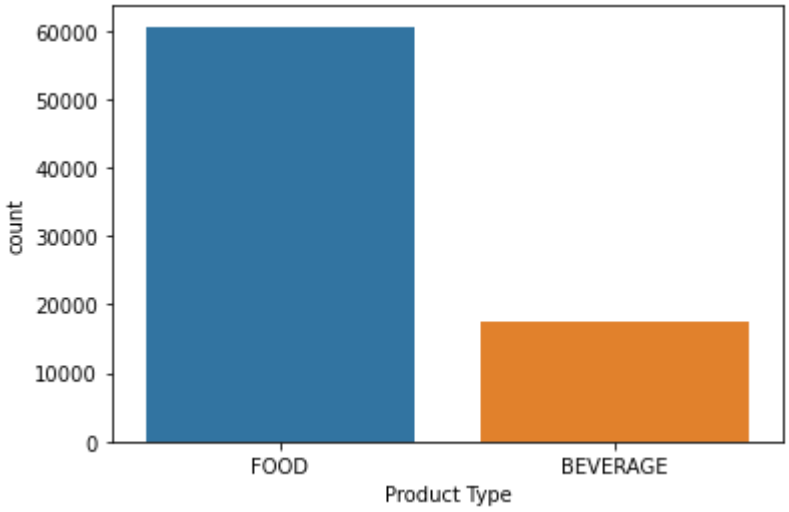
Visualizations

In [37]:
```
p6 = sns.histplot(df_addedsugar, x="AddedSugarClass", hue="Product Type")

plt.show()
```



In [ ]:
```

```

In [38]:
```
p3 = sns.countplot(data=df_addedsugar, x="Product Type")
plt.show()
```



Feature selection:

In [39]:
```
df_addedsugar["Local Material Description"].value_counts()
```

Out[39]:
```
МКК Сметана "Кунгурский" Нет значения вк    44
BRAND102 Curd traditional "Classic"          43
МКК Кефир "Кунгурский" Нет значения вкус     43
ВБД Десерт творожный взбитый пастеризова     36
18CT VP FLAVOR                               36
                                             ..
TP TRPCN TRNC K MYV 1L 12X AKDENIZ            1
TP TRPCN SFT ELM 200ML 4X6 100 FJ             1
RUF CBX KETC SUPER 30 BONUSPACK               1
RUF MAKS PEYSOG AILE 76 EXP                   1
BRAND467 7.04Z 8CS CARAMEL                    1
Name: Local Material Description, Length: 24175, dtype: int64
```

In [40]:
```
df_addedsugar["Global Brand Name"].value_counts()
```

Out[40]:
```
BRAND589    23377
BRAND7      17433
BRAND591     9005
BRAND2       5520
BRAND3       4973
BRAND11      4533
BRAND6       4144
BRAND9       2893
BRAND77      2720
BRAND40      1556
BRAND590     1118
BRAND13       875
Name: Global Brand Name, dtype: int64
```

In [41]:
```
df_addedsugar["Standard Brand Name"].value_counts()
```

Out[41]:
```
BRAND7      13300
BRAND2       5476
BRAND3       4973
BRAND593     4389
BRAND6       4144
             ...
BRAND751        2
BRAND715        2
BRAND736        1
BRAND557        1
BRAND755        1
Name: Standard Brand Name, Length: 286, dtype: int64
```

In [42]:
```
df_addedsugar["Product ID"].value_counts()
```

Out[42]:
```
953792      16
953787      16
953797      15
24877137    14
985311      14
            ..
13320529     1
13747860     1
24162264     1
24302574     1
975202       1
Name: Product ID, Length: 25614, dtype: int64
```

In [43]:
```
df_addedsugar["Country"].value_counts()
```

Out[43]:
```
United States         20655
Mexico                11279
Russian Federation     8200
United Kingdom         3344
Canada                 3194
France                 2833
Turkey                 2794
Thailand               2442
India                  2344
China                  2208
Spain                  2100
Saudi Arabia           2058
Colombia               2008
Ukraine                1924
Australia              1791
Netherlands            1707
Brazil                 1491
Poland                 1286
Romania                1175
Egypt                  1068
South Africa            895
Pakistan                617
Germany                 438
Philippines             220
Vietnam                  76
Name: Country, dtype: int64
```

In [44]:
```
df_addedsugar["Flavor"].value_counts()
```

Out[44]:
```
Salted                              1745
Original                            1132
Flamin' Hot                         1085
BBQ                                 1054
Orange                               898
                                    ...
Cheese & Nacho Cheese-Mixups           1
LAYS CLAS SWEET CHILI FLAVOR           1
RAISIN & ALMOND                        1
Cinnamon & Honey                       1
CARAMEL (GLUTEN FREE)                  1
Name: Flavor, Length: 3814, dtype: int64
```

In [45]:
```
df_addedsugar.head()
```

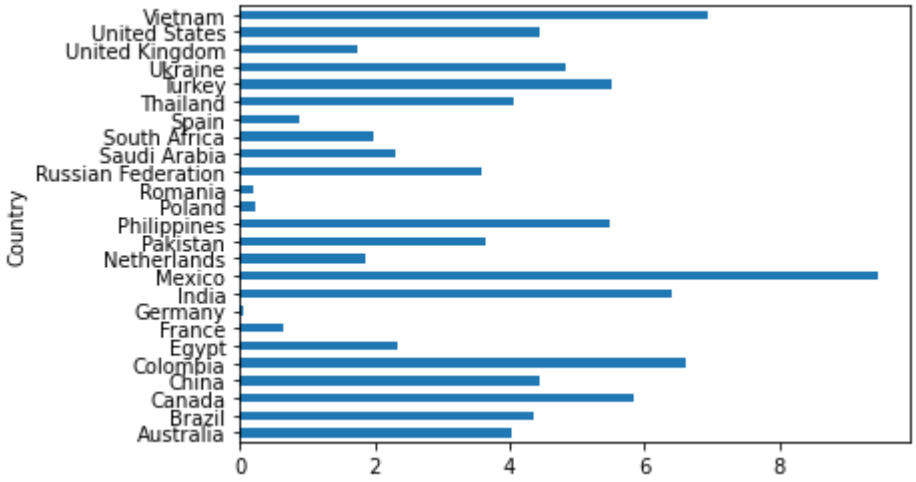
Out[45]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2020	Q1	Egypt	BRAND1 7 LE C AND O 120G 6P	23673483	FOOD	BRAND7	BRAND1	BRAND1
1	1	2020	Q1	Egypt	BRAND2 CHEESE 84G 9P 5LE	24784584	FOOD	BRAND2	BRAND2	BRAND2
2	2	2020	Q1	Egypt	BRAND2 SWEET CHILLI 84G 9P 5LE	24784585	FOOD	BRAND2	BRAND2	BRAND2
3	5	2020	Q1	Egypt	BRAND1 180G SALT 6P	22200907	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
4	7	2020	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

5 rows × 22 columns

In [46]:
```
df_addedsugar.groupby('Country').AddedSugars.mean().plot(kind='barh')
```

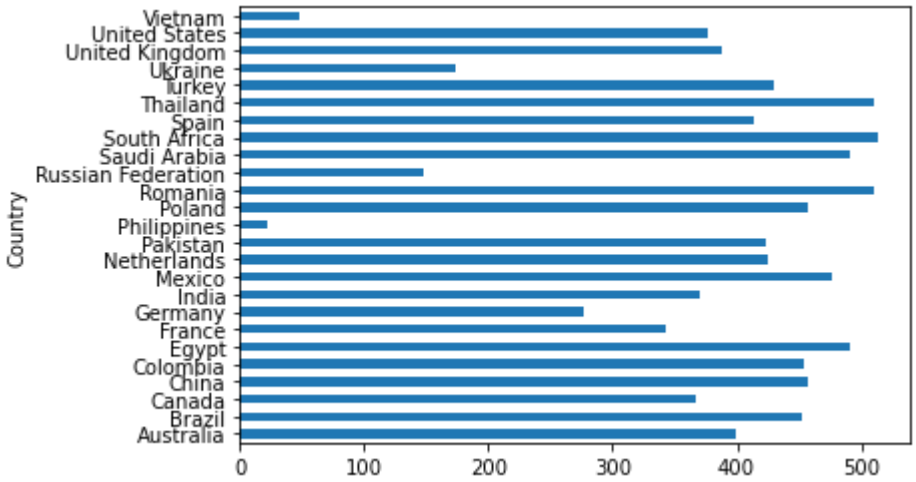
Out[46]:
```
<AxesSubplot:ylabel='Country'>
```



In [47]:
```
df_addedsugar=df_addedsugar.rename(columns={"Total Calories": "TotalCalories"})
```

In [48]:
```
df_addedsugar.groupby('Country').TotalCalories.mean().plot(kind='barh')
```

Out[48]:
```
<AxesSubplot:ylabel='Country'>
```



In [49]:
```
df_addedsugar.dtypes
```

Out[49]:
```
index                                  int64
Year                                   int64
Quarter                               object
Country                               object
Local Material Description            object
Product ID                             int64
Product Type                          object
Global Brand Name                     object
Standard Brand Name                   object
Standard Sub-Brand Name               object
Flavor                                object
TotalCalories                        float64
Total Fat                            float64
Saturated Fat                        float64
Saturated Fat per 100 kcals          float64
AddedSugars                          float64
Calories from Added Sugars per 355ml float64
% of Energy from Added Sugars        float64
Sodium                               float64
Sodium per kcal                      float64
Protein                              float64
AddedSugarClass                       object
dtype: object
```

In [ ]:

Classification would be the best type of model for this kind of data.

Some preprocessing with one-hot encoding for country, product type, and Global brand name, because they are less than 200 categories. Convert added sugars class to 1, 2 and 3 for low, mid and high.

The features: index, year, quarter, local material description, product ID, Standard Brand, Standard Sub-Brand Name, and Flavor will be dropped as they have over 200 catergories each.

This will be best approach for the classification modeling to be performed.

In [50]:
```
df_addedsugar=df_addedsugar.drop(['index', 'Year', 'Quarter', 'Local Material Descript
```

In [51]:
```
df_addedsugar.dtypes
```

Out[51]:
```
Country                                  object
Product Type                             object
Global Brand Name                        object
TotalCalories                           float64
Total Fat                               float64
Saturated Fat                           float64
Saturated Fat per 100 kcals             float64
AddedSugars                             float64
Calories from Added Sugars per 355ml    float64
% of Energy from Added Sugars           float64
Sodium                                  float64
Sodium per kcal                         float64
Protein                                 float64
AddedSugarClass                          object
dtype: object
```

In [52]:
```
# These are the columns to one-hot encode to simplify machine learning classification
df_addedsugar_ohe=df_addedsugar
categorical_cols=['Country', 'Product Type', 'Global Brand Name']
for col in categorical_cols:
    col_ohe = pd.get_dummies(df_addedsugar[col], prefix=col)
    df_addedsugar_ohe = pd.concat((df_addedsugar_ohe , col_ohe), axis=1).drop(col, axi
```

In [53]:
```
print(df_addedsugar_ohe.columns)
```

```
Index(['TotalCalories', 'Total Fat', 'Saturated Fat',
       'Saturated Fat per 100 kcals', 'AddedSugars',
       'Calories from Added Sugars per 355ml', '% of Energy from Added Sugars',
       'Sodium', 'Sodium per kcal', 'Protein', 'AddedSugarClass',
       'Country_Australia', 'Country_Brazil', 'Country_Canada',
       'Country_China', 'Country_Colombia', 'Country_Egypt', 'Country_France',
       'Country_Germany', 'Country_India', 'Country_Mexico',
       'Country_Netherlands', 'Country_Pakistan', 'Country_Philippines',
       'Country_Poland', 'Country_Romania', 'Country_Russian Federation',
       'Country_Saudi Arabia', 'Country_South Africa', 'Country_Spain',
       'Country_Thailand', 'Country_Turkey', 'Country_Ukraine',
       'Country_United Kingdom', 'Country_United States', 'Country_Vietnam',
       'Product Type_BEVERAGE', 'Product Type_FOOD',
       'Global Brand Name_BRAND11', 'Global Brand Name_BRAND13',
       'Global Brand Name_BRAND2', 'Global Brand Name_BRAND3',
       'Global Brand Name_BRAND40', 'Global Brand Name_BRAND589',
       'Global Brand Name_BRAND590', 'Global Brand Name_BRAND591',
       'Global Brand Name_BRAND6', 'Global Brand Name_BRAND7',
       'Global Brand Name_BRAND77', 'Global Brand Name_BRAND9'],
      dtype='object')
```

In [54]:
```
df_addedsugar_ohe.dtypes
```

Out[54]:
```
TotalCalories                          float64
Total Fat                              float64
Saturated Fat                          float64
Saturated Fat per 100 kcals            float64
AddedSugars                            float64
Calories from Added Sugars per 355ml   float64
% of Energy from Added Sugars          float64
Sodium                                 float64
Sodium per kcal                        float64
Protein                                float64
AddedSugarClass                         object
Country_Australia                        uint8
Country_Brazil                           uint8
Country_Canada                           uint8
Country_China                            uint8
Country_Colombia                         uint8
Country_Egypt                            uint8
Country_France                           uint8
Country_Germany                          uint8
Country_India                            uint8
Country_Mexico                           uint8
Country_Netherlands                      uint8
Country_Pakistan                         uint8
Country_Philippines                      uint8
Country_Poland                           uint8
Country_Romania                          uint8
Country_Russian Federation               uint8
Country_Saudi Arabia                     uint8
Country_South Africa                     uint8
Country_Spain                            uint8
Country_Thailand                         uint8
Country_Turkey                           uint8
Country_Ukraine                          uint8
Country_United Kingdom                   uint8
Country_United States                    uint8
Country_Vietnam                          uint8
Product Type_BEVERAGE                    uint8
Product Type_FOOD                        uint8
Global Brand Name_BRAND11                uint8
Global Brand Name_BRAND13                uint8
Global Brand Name_BRAND2                 uint8
Global Brand Name_BRAND3                 uint8
Global Brand Name_BRAND40                uint8
Global Brand Name_BRAND589               uint8
Global Brand Name_BRAND590               uint8
Global Brand Name_BRAND591               uint8
Global Brand Name_BRAND6                 uint8
Global Brand Name_BRAND7                 uint8
Global Brand Name_BRAND77                uint8
Global Brand Name_BRAND9                 uint8
dtype: object
```

In [55]:
```
#Convert added sugars class to 1, 2 and 3 for low, mid and high.
df_addedsugar_ohe["AddedSugarClass"].replace('low', '1',inplace=True)
df_addedsugar_ohe["AddedSugarClass"].replace('mid', '2',inplace=True)
df_addedsugar_ohe["AddedSugarClass"].replace('high', '3',inplace=True)
```

In [56]:
```
df_addedsugar_ohe["AddedSugarClass"].value_counts()
```

Out[56]:
```
1    49850
2    18804
3     9493
Name: AddedSugarClass, dtype: int64
```

In [57]:
```
df_addedsugar_ohe.head()
```

Out[57]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal	Prot
0	488.0	28.0	13.0	2.66	1.8	25.56	1.48	554.0	1.1352	
1	499.0	26.0	12.0	2.40	2.9	41.18	2.32	612.0	1.2265	
2	491.0	25.0	12.0	2.44	4.4	62.48	3.58	808.5	1.6466	
3	502.0	30.0	13.3	2.65	0.6	8.52	0.48	600.0	1.1952	
4	460.0	25.0	11.0	2.39	7.0	99.40	6.09	540.0	1.1739	

5 rows × 50 columns

In [58]:
```
df_addedsugar_ohe.dtypes
```

Out[58]:
```
TotalCalories                          float64
Total Fat                              float64
Saturated Fat                          float64
Saturated Fat per 100 kcals            float64
AddedSugars                            float64
Calories from Added Sugars per 355ml   float64
% of Energy from Added Sugars          float64
Sodium                                 float64
Sodium per kcal                        float64
Protein                                float64
AddedSugarClass                         object
Country_Australia                        uint8
Country_Brazil                           uint8
Country_Canada                           uint8
Country_China                            uint8
Country_Colombia                         uint8
Country_Egypt                            uint8
Country_France                           uint8
Country_Germany                          uint8
Country_India                            uint8
Country_Mexico                           uint8
Country_Netherlands                      uint8
Country_Pakistan                         uint8
Country_Philippines                      uint8
Country_Poland                           uint8
Country_Romania                          uint8
Country_Russian Federation               uint8
Country_Saudi Arabia                     uint8
Country_South Africa                     uint8
Country_Spain                            uint8
Country_Thailand                         uint8
Country_Turkey                           uint8
Country_Ukraine                          uint8
Country_United Kingdom                   uint8
Country_United States                    uint8
Country_Vietnam                          uint8
Product Type_BEVERAGE                    uint8
Product Type_FOOD                        uint8
Global Brand Name_BRAND11                uint8
Global Brand Name_BRAND13                uint8
Global Brand Name_BRAND2                 uint8
Global Brand Name_BRAND3                 uint8
Global Brand Name_BRAND40                uint8
Global Brand Name_BRAND589               uint8
Global Brand Name_BRAND590               uint8
Global Brand Name_BRAND591               uint8
Global Brand Name_BRAND6                 uint8
Global Brand Name_BRAND7                 uint8
Global Brand Name_BRAND77                uint8
Global Brand Name_BRAND9                 uint8
dtype: object
```

In [59]:
```
df_addedsugar_ohe.shape
```

Out[59]:
```
(78147, 50)
```

Scaling the features using standard scaler

In [60]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [61]:
```
columns_to_scale=df_addedsugar_ohe.iloc[:,[0,1,2,3,4,5,6,7,8,9]]
columns_to_scale
```

Out[61]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal
0	488.00	28.00	13.00	2.66	1.80	25.56	1.48	554.00	1.1352
1	499.00	26.00	12.00	2.40	2.90	41.18	2.32	612.00	1.2265
2	491.00	25.00	12.00	2.44	4.40	62.48	3.58	808.50	1.6466
3	502.00	30.00	13.30	2.65	0.60	8.52	0.48	600.00	1.1952
4	460.00	25.00	11.00	2.39	7.00	99.40	6.09	540.00	1.1739
...	...	...	...	...	...	...	...	...	...
78142	400.00	24.00	4.00	1.00	12.00	170.40	12.00	240.00	0.6000
78143	400.00	24.00	4.00	1.00	12.00	170.40	12.00	240.00	0.6000
78144	400.00	24.00	4.00	1.00	12.00	170.40	12.00	240.00	0.6000
78145	400.00	24.00	8.00	2.00	12.00	170.40	12.00	180.00	0.4500
78146	382.96	2.72	0.54	0.14	30.35	430.97	31.70	1008.43	2.6333

78147 rows × 10 columns

In [62]:
```
scaled_values=scaler.fit_transform(columns_to_scale)
scaled_values
```

Out[62]:
```
array([[ 5.15831001e-01,  5.87451569e-01,  1.52447329e+00, ...,
         2.11878809e-01, -8.30454938e-02,  3.43951335e-01],
       [ 5.69950031e-01,  4.47916936e-01,  1.33365049e+00, ...,
         3.53221817e-01, -6.82637462e-02,  9.91064427e-02],
       [ 5.30590736e-01,  3.78149620e-01,  1.33365049e+00, ...,
         8.32082179e-01, -2.48278883e-04,  1.56717006e-01],
       ...,
       [ 8.28787625e-02,  3.08382304e-01, -1.92931942e-01, ...,
        -5.53322993e-01, -1.69696001e-01,  8.76849041e-01],
       [ 8.28787625e-02,  3.08382304e-01,  5.70359273e-01, ...,
        -6.99539897e-01, -1.93981457e-01,  8.76849041e-01],
       [-9.56534626e-04, -1.17626619e+00, -8.53178843e-01, ...,
         1.31930128e+00,  1.59501451e-01, -1.55820298e-01]])
```

In [63]:
```
scaled_values = pd.DataFrame(scaled_values, columns=columns_to_scale.columns)
scaled_values
```

Out[63]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium
0	0.515831	0.587452	1.524473	1.412038	-0.271801	-0.271801	-0.299931	0.211879
1	0.569950	0.447917	1.333650	1.176199	-0.164903	-0.164903	-0.278821	0.353222
2	0.530591	0.378150	1.333650	1.212482	-0.019133	-0.019133	-0.247156	0.832082
3	0.584710	0.726986	1.581720	1.402967	-0.388417	-0.388417	-0.325062	0.323978
4	0.378073	0.378150	1.142828	1.167129	0.233535	0.233535	-0.184077	0.177762
...	...	...	...	...	...	...	...	...
78142	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78143	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78144	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78145	0.082879	0.308382	0.570359	0.813371	0.719436	0.719436	-0.035552	-0.699540
78146	-0.000957	-1.176266	-0.853179	-0.873783	2.502690	2.502690	0.459529	1.319301

78147 rows × 10 columns

In [64]:
```
scaled_dfaddedsugar_ohe = pd.concat([scaled_values,df_addedsugar_ohe.iloc[:,[10,11,12,
scaled_dfaddedsugar_ohe
```

Out[64]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium
0	0.515831	0.587452	1.524473	1.412038	-0.271801	-0.271801	-0.299931	0.211879
1	0.569950	0.447917	1.333650	1.176199	-0.164903	-0.164903	-0.278821	0.353222
2	0.530591	0.378150	1.333650	1.212482	-0.019133	-0.019133	-0.247156	0.832082
3	0.584710	0.726986	1.581720	1.402967	-0.388417	-0.388417	-0.325062	0.323978
4	0.378073	0.378150	1.142828	1.167129	0.233535	0.233535	-0.184077	0.177762
...	...	...	...	...	...	...	...	...
78142	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78143	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78144	0.082879	0.308382	-0.192932	-0.093701	0.719436	0.719436	-0.035552	-0.553323
78145	0.082879	0.308382	0.570359	0.813371	0.719436	0.719436	-0.035552	-0.699540
78146	-0.000957	-1.176266	-0.853179	-0.873783	2.502690	2.502690	0.459529	1.319301

78147 rows × 50 columns

In [65]:
```
scaled_dfaddedsugar_ohe.dtypes
```

Out[65]:
```
TotalCalories                          float64
Total Fat                              float64
Saturated Fat                          float64
Saturated Fat per 100 kcals            float64
AddedSugars                            float64
Calories from Added Sugars per 355ml   float64
% of Energy from Added Sugars          float64
Sodium                                 float64
Sodium per kcal                        float64
Protein                                float64
AddedSugarClass                         object
Country_Australia                        uint8
Country_Brazil                           uint8
Country_Canada                           uint8
Country_China                            uint8
Country_Colombia                         uint8
Country_Egypt                            uint8
Country_France                           uint8
Country_Germany                          uint8
Country_India                            uint8
Country_Mexico                           uint8
Country_Netherlands                      uint8
Country_Pakistan                         uint8
Country_Philippines                      uint8
Country_Poland                           uint8
Country_Romania                          uint8
Country_Russian Federation               uint8
Country_Saudi Arabia                     uint8
Country_South Africa                     uint8
Country_Spain                            uint8
Country_Thailand                         uint8
Country_Turkey                           uint8
Country_Ukraine                          uint8
Country_United Kingdom                   uint8
Country_United States                    uint8
Country_Vietnam                          uint8
Product Type_BEVERAGE                    uint8
Product Type_FOOD                        uint8
Global Brand Name_BRAND11                uint8
Global Brand Name_BRAND13                uint8
Global Brand Name_BRAND2                 uint8
Global Brand Name_BRAND3                 uint8
Global Brand Name_BRAND40                uint8
Global Brand Name_BRAND589               uint8
Global Brand Name_BRAND590               uint8
Global Brand Name_BRAND591               uint8
Global Brand Name_BRAND6                 uint8
Global Brand Name_BRAND7                 uint8
Global Brand Name_BRAND77                uint8
Global Brand Name_BRAND9                 uint8
dtype: object
```

In [66]:
```
cols2num = ['TotalCalories','Total Fat','Saturated Fat','Saturated Fat per 100 kcals',
scaled_dfaddedsugar_ohe[cols2num] = scaled_dfaddedsugar_ohe[cols2num].applymap(pd.to_r
```

In [67]:
```
scaled_dfaddedsugar_ohe.dtypes
```

Out[67]:
```
TotalCalories                          float64
Total Fat                              float64
Saturated Fat                          float64
Saturated Fat per 100 kcals            float64
AddedSugars                            float64
Calories from Added Sugars per 355ml   float64
% of Energy from Added Sugars          float64
Sodium                                 float64
Sodium per kcal                        float64
Protein                                float64
AddedSugarClass                         object
Country_Australia                        uint8
Country_Brazil                           uint8
Country_Canada                           uint8
Country_China                            uint8
Country_Colombia                         uint8
Country_Egypt                            uint8
Country_France                           uint8
Country_Germany                          uint8
Country_India                            uint8
Country_Mexico                           uint8
Country_Netherlands                      uint8
Country_Pakistan                         uint8
Country_Philippines                      uint8
Country_Poland                           uint8
Country_Romania                          uint8
Country_Russian Federation               uint8
Country_Saudi Arabia                     uint8
Country_South Africa                     uint8
Country_Spain                            uint8
Country_Thailand                         uint8
Country_Turkey                           uint8
Country_Ukraine                          uint8
Country_United Kingdom                   uint8
Country_United States                    uint8
Country_Vietnam                          uint8
Product Type_BEVERAGE                    uint8
Product Type_FOOD                        uint8
Global Brand Name_BRAND11                uint8
Global Brand Name_BRAND13                uint8
Global Brand Name_BRAND2                 uint8
Global Brand Name_BRAND3                 uint8
Global Brand Name_BRAND40                uint8
Global Brand Name_BRAND589               uint8
Global Brand Name_BRAND590               uint8
Global Brand Name_BRAND591               uint8
Global Brand Name_BRAND6                 uint8
Global Brand Name_BRAND7                 uint8
Global Brand Name_BRAND77                uint8
Global Brand Name_BRAND9                 uint8
dtype: object
```

In [68]:
```
scaled_dfaddedsugar_ohe.isnull().sum(axis = 0)
```

Out[68]:
```
TotalCalories                          0
Total Fat                              0
Saturated Fat                          0
Saturated Fat per 100 kcals            0
AddedSugars                            0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 0
Sodium per kcal                        0
Protein                                0
AddedSugarClass                        0
Country_Australia                      0
Country_Brazil                         0
Country_Canada                         0
Country_China                          0
Country_Colombia                       0
Country_Egypt                          0
Country_France                         0
Country_Germany                        0
Country_India                          0
Country_Mexico                         0
Country_Netherlands                    0
Country_Pakistan                       0
Country_Philippines                    0
Country_Poland                         0
Country_Romania                        0
Country_Russian Federation             0
Country_Saudi Arabia                   0
Country_South Africa                   0
Country_Spain                          0
Country_Thailand                       0
Country_Turkey                         0
Country_Ukraine                        0
Country_United Kingdom                 0
Country_United States                  0
Country_Vietnam                        0
Product Type_BEVERAGE                  0
Product Type_FOOD                      0
Global Brand Name_BRAND11              0
Global Brand Name_BRAND13              0
Global Brand Name_BRAND2               0
Global Brand Name_BRAND3               0
Global Brand Name_BRAND40              0
Global Brand Name_BRAND589             0
Global Brand Name_BRAND590             0
Global Brand Name_BRAND591             0
Global Brand Name_BRAND6               0
Global Brand Name_BRAND7               0
Global Brand Name_BRAND77              0
Global Brand Name_BRAND9               0
dtype: int64
```

In [69]:
```
scaled_dfaddedsugar_ohe.shape
```

Out[69]:
```
(78147, 50)
```

In [ ]:
```

```

Modeling-Classification modeling

Training

In [70]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
```

In [71]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\zahra\anaconda3\lib\site-packages
(1.6.2)
Requirement already satisfied: scipy in c:\users\zahra\anaconda3\lib\site-packages (f
rom xgboost) (1.7.3)
Requirement already satisfied: numpy in c:\users\zahra\anaconda3\lib\site-packages (f
rom xgboost) (1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

In [72]:
```
import xgboost as xgb
```

In [73]:
```
X = scaled_dfaddedsugar_ohe.drop('AddedSugarClass', axis = 1)
y = scaled_dfaddedsugar_ohe['AddedSugarClass']
```

In [74]:
```
key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','Ran
value = [LogisticRegression(random_state=9), KNeighborsClassifier(), SVC(), DecisionTr
models = dict(zip(key,value))
```

In [75]:
```
#importing train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=42,sh
```

In [76]:
```
predicted =[]
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state =
```

In [77]:
```
for name,algo in models.items():
    model=algo
    model.fit(X_train,y_train)
    predict = model.predict(X_test)
    acc = accuracy_score(y_test, predict)
    predicted.append(acc)
    print(name,acc)
```

```
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression 0.9942418426103646
KNeighborsClassifier 0.9957773512476008
SVC 0.990722968650032
DecisionTreeClassifier 1.0
RandomForestClassifier 1.0
GradientBoostingClassifier 1.0
AdaBoostClassifier 1.0
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [77], in <cell line: 1>()
      1 for name,algo in models.items():
      2     model=algo
----> 3     model.fit(X_train,y_train)
      4     predict = model.predict(X_test)
      5     acc = accuracy_score(y_test, predict)

File ~\anaconda3\lib\site-packages\xgboost\core.py:575, in _deprecate_positional_arg
s.<locals>.inner_f(*args, **kwargs)
    573 for k, arg in zip(sig.parameters, args):
    574     kwargs[k] = arg
--> 575 return f(**kwargs)

File ~\anaconda3\lib\site-packages\xgboost\sklearn.py:1357, in XGBClassifier.fit(sel
f, X, y, sample_weight, base_margin, eval_set, eval_metric, early_stopping_rounds, ve
rbose, xgb_model, sample_weight_eval_set, base_margin_eval_set, feature_weights, call
backs)
   1352     expected_classes = np.arange(self.n_classes_)
   1353 if (
   1354     self.classes_.shape != expected_classes.shape
   1355     or not (self.classes_ == expected_classes).all()
   1356 ):
-> 1357     raise ValueError(
   1358         f"Invalid classes inferred from unique values of `y`.  "
   1359         f"Expected: {expected_classes}, got {self.classes_}"
   1360     )
   1362 params = self.get_xgb_params()
   1364 if callable(self.objective):

ValueError: Invalid classes inferred from unique values of `y`.  Expected: [0 1 2], g
ot ['1' '2' '3']
```

In [78]:

```
algo_tests = list(zip(predicted,key))
algo_tests=pd.DataFrame(algo_tests, columns=['predicted','key'])
algo_tests.head(5)
```

Out[78]:

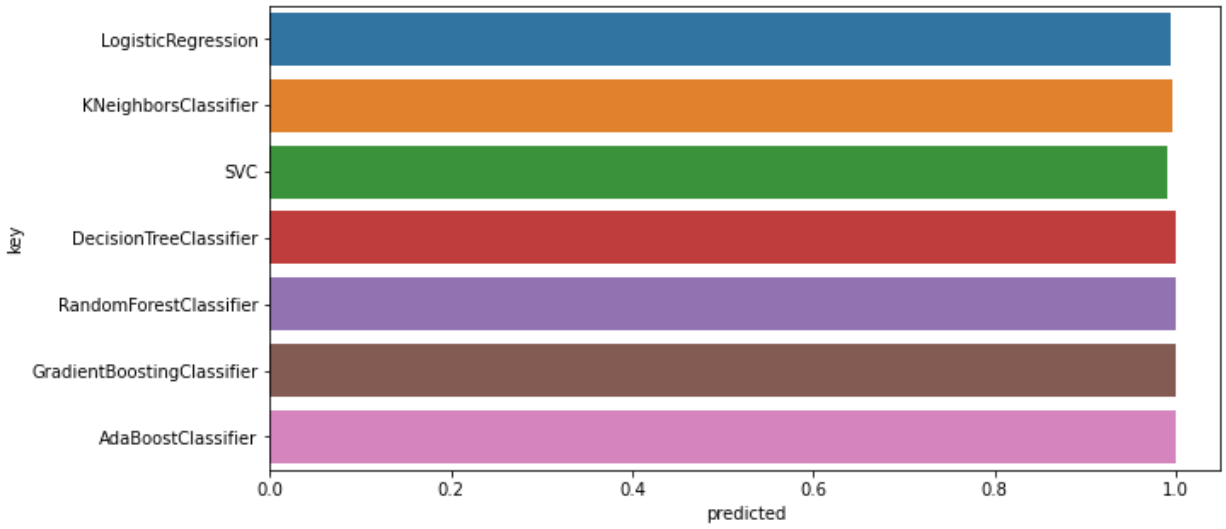
	predicted	key
0	0.994242	LogisticRegression
1	0.995777	KNeighborsClassifier
2	0.990723	SVC
3	1.000000	DecisionTreeClassifier
4	1.000000	RandomForestClassifier

In [79]:
```
algo_tests
```

Out[79]:

	predicted	key
0	0.994242	LogisticRegression
1	0.995777	KNeighborsClassifier
2	0.990723	SVC
3	1.000000	DecisionTreeClassifier
4	1.000000	RandomForestClassifier
5	1.000000	GradientBoostingClassifier
6	1.000000	AdaBoostClassifier

In [80]:
```
plt.figure(figsize = (10,5))
ax=sns.barplot(x = 'predicted', y = 'key', data=algo_tests)
```



Any classification model will work great based on the training dataset.

In [81]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

error_list = list()
```

```
# Iterate through various possibilities for number of trees
tree_list = [1, 2, 3]
for n_trees in tree_list:

    # Initialize the gradient boost classifier
    GBC = GradientBoostingClassifier(n_estimators=n_trees, random_state=42)

    # Fit the model
    print(f'Fitting model with {n_trees} trees')
    GBC.fit(X_train.values, y_train.values)
    y_pred = GBC.predict(X_test)

    # Get the error
    error = 1.0 - accuracy_score(y_test, y_pred)

    # Store it
    error_list.append(pd.Series({'n_trees': n_trees, 'error': error}))

error_df = pd.concat(error_list, axis=1).T.set_index('n_trees')

error_df
```

```
Fitting model with 1 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 2 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 3 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
```

Out[81]:

n_trees	error
1.0	0.362188
2.0	0.362188
3.0	0.122713

In [82]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
```

In [83]:
```
dt.tree_.node_count, dt.tree_.max_depth
```

Out[83]:
```
(5, 2)
```

In [84]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
```

```
                        'recall': recall_score(y_true, y_pred),
                        'f1': f1_score(y_true, y_pred)},
                       name=label)
```

In [85]:
```
# The error on the training and test data sets
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
### END SOLUTION
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [85], in <cell line: 5>()
      2 y_train_pred = dt.predict(X_train)
      3 y_test_pred = dt.predict(X_test)
----> 5 train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'trai
n'),
      6                                   measure_error(y_test, y_test_pred, 'test')],
      7                                   axis=1)
      9 train_test_full_error

Input In [84], in measure_error(y_true, y_pred, label)
      3 def measure_error(y_true, y_pred, label):
      4     return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
----> 5                       'precision': precision_score(y_true, y_pred),
      6                       'recall': recall_score(y_true, y_pred),
      7                       'f1': f1_score(y_true, y_pred)},
      8                      name=label)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1757, in precis
ion_score(y_true, y_pred, labels, pos_label, average, sample_weight, zero_division)
   1628 def precision_score(
   1629     y_true,
   1630     y_pred,
   (...)
   1636     zero_division="warn",
   1637 ):
   1638     """Compute the precision.
   1639
   1640     The precision is the ratio ``tp / (tp + fp)`` where ``tp`` is the number
 of
   (...)
   1755     array([0.5, 1. , 1. ])
   1756     """
-> 1757     p, _, _, _ = precision_recall_fscore_support(
   1758         y_true,
   1759         y_pred,
   1760         labels=labels,
   1761         pos_label=pos_label,
   1762         average=average,
   1763         warn_for=("precision",),
   1764         sample_weight=sample_weight,
   1765         zero_division=zero_division,
   1766     )
   1767     return p

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1544, in precis
ion_recall_fscore_support(y_true, y_pred, beta, labels, pos_label, average, warn_for,
sample_weight, zero_division)
   1542 if beta < 0:
   1543     raise ValueError("beta should be >=0 in the F-beta score")
-> 1544 labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1546 # Calculate tp_sum, pred_sum, true_sum ###
   1547 samplewise = average == "samples"

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1365, in _check
_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1363         if y_type == "multiclass":
   1364             average_options.remove("samples")
-> 1365         raise ValueError(
```

```
   1366             "Target is %s but average='binary'. Please "
   1367             "choose another average setting, one of %r." % (y_type, average_o
ptions)
   1368         )
   1369 elif pos_label not in (None, 1):
   1370     warnings.warn(
   1371         "Note that pos_label (set to %r) is ignored when "
   1372         "average != 'binary' (got %r). You may use "
   (...)
   1375         UserWarning,
   1376     )

ValueError: Target is multiclass but average='binary'. Please choose another average
 setting, one of [None, 'micro', 'macro', 'weighted'].
```

In [86]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

In [87]:
```
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

Out[87]:
```
(3, 1)
```

In [88]:
```
y_train_pred_gr = GR.predict(X_train)
y_test_pred_gr = GR.predict(X_test)

train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr, 'train'),
                                 measure_error(y_test, y_test_pred_gr, 'test')],
                                axis=1)
train_test_gr_error
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [88], in <cell line: 4>()
      1 y_train_pred_gr = GR.predict(X_train)
      2 y_test_pred_gr = GR.predict(X_test)
----> 4 train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr, 'tra
in'),
      5                                 measure_error(y_test, y_test_pred_gr, 'test'
)],
      6                                axis=1)
      7 train_test_gr_error

Input In [84], in measure_error(y_true, y_pred, label)
      3 def measure_error(y_true, y_pred, label):
      4     return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
----> 5                       'precision': precision_score(y_true, y_pred),
      6                       'recall': recall_score(y_true, y_pred),
      7                       'f1': f1_score(y_true, y_pred)},
      8                      name=label)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1757, in precis
ion_score(y_true, y_pred, labels, pos_label, average, sample_weight, zero_division)
   1628 def precision_score(
   1629     y_true,
   1630     y_pred,
   (...)
   1636     zero_division="warn",
   1637 ):
   1638     """Compute the precision.
   1639
   1640     The precision is the ratio ``tp / (tp + fp)`` where ``tp`` is the number
 of
   (...)
   1755     array([0.5, 1. , 1. ])
   1756     """
-> 1757     p, _, _, _ = precision_recall_fscore_support(
   1758         y_true,
   1759         y_pred,
   1760         labels=labels,
   1761         pos_label=pos_label,
   1762         average=average,
   1763         warn_for=("precision",),
   1764         sample_weight=sample_weight,
   1765         zero_division=zero_division,
   1766     )
   1767     return p

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1544, in precis
ion_recall_fscore_support(y_true, y_pred, beta, labels, pos_label, average, warn_for,
sample_weight, zero_division)
   1542 if beta < 0:
   1543     raise ValueError("beta should be >=0 in the F-beta score")
-> 1544 labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1546 # Calculate tp_sum, pred_sum, true_sum ###
   1547 samplewise = average == "samples"

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1365, in _check
_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1363         if y_type == "multiclass":
   1364             average_options.remove("samples")
```

```
-> 1365         raise ValueError(
   1366             "Target is %s but average='binary'. Please "
   1367             "choose another average setting, one of %r." % (y_type, average_o
ptions)
   1368         )
   1369 elif pos_label not in (None, 1):
   1370     warnings.warn(
   1371         "Note that pos_label (set to %r) is ignored when "
   1372         "average != 'binary' (got %r). You may use "
   (...)
   1375         UserWarning,
   1376     )

ValueError: Target is multiclass but average='binary'. Please choose another average
 setting, one of [None, 'micro', 'macro', 'weighted'].
```

In [89]:
```
from io import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
from pydotplus import graph_from_dot_data
```

In [90]:
```
pip install pydotplus
```

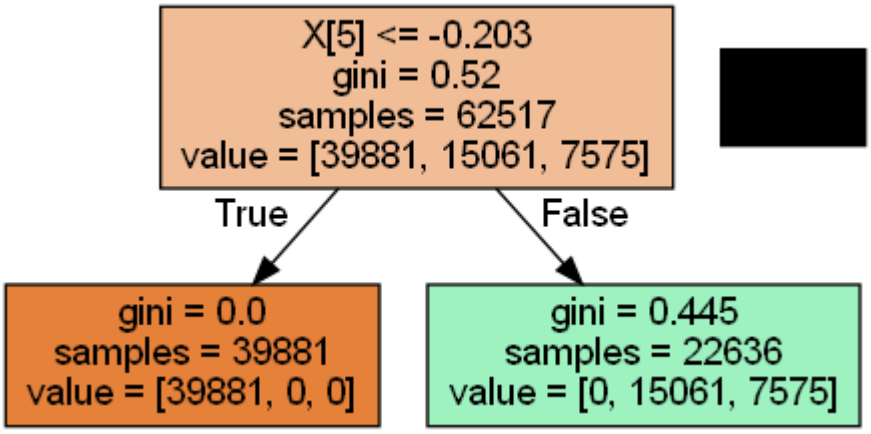
```
Requirement already satisfied: pydotplus in c:\users\zahra\anaconda3\lib\site-package
s (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\zahra\anaconda3\lib\site-
packages (from pydotplus) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [91]:
```
conda install graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.

Retrieving notices: ...working... done

Note: you may need to restart the kernel to use updated packages.
```

In [92]:
```
# Create an output destination for the file
dot_data = StringIO()

export_graphviz(GR.best_estimator_, out_file=dot_data, filled=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

# View the tree image
filename = 'Pepsicotrain.png'
graph.write_png(filename)
Image(filename=filename)
### END SOLUTION
```

Out[92]:



Let's model without scaling just rounding the features.

Scaling makes the data modeling overfitting in the decision tree with depth of 1 and node of 3

In [93]:
```
df_addedsugar_ohe.head()
```

Out[93]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal	Prot
0	488.0	28.0	13.0	2.66	1.8	25.56	1.48	554.0	1.1352	
1	499.0	26.0	12.0	2.40	2.9	41.18	2.32	612.0	1.2265	
2	491.0	25.0	12.0	2.44	4.4	62.48	3.58	808.5	1.6466	
3	502.0	30.0	13.3	2.65	0.6	8.52	0.48	600.0	1.1952	
4	460.0	25.0	11.0	2.39	7.0	99.40	6.09	540.0	1.1739	

5 rows × 50 columns

In [94]:
```
cols2num = ['TotalCalories','Total Fat','Saturated Fat','Saturated Fat per 100 kcals',
df_addedsugar_ohe[cols2num] = df_addedsugar_ohe[cols2num].round(0).astype(int)
```

In [95]:
```
df_addedsugar_ohe.dtypes
```

Out[95]:

```
TotalCalories                          int32
Total Fat                              int32
Saturated Fat                          int32
Saturated Fat per 100 kcals            int32
AddedSugars                            int32
Calories from Added Sugars per 355ml   int32
% of Energy from Added Sugars          int32
Sodium                                 int32
Sodium per kcal                        int32
Protein                                int32
AddedSugarClass                       object
Country_Australia                      uint8
Country_Brazil                         uint8
Country_Canada                         uint8
Country_China                          uint8
Country_Colombia                       uint8
Country_Egypt                          uint8
Country_France                         uint8
Country_Germany                        uint8
Country_India                          uint8
Country_Mexico                         uint8
Country_Netherlands                    uint8
Country_Pakistan                       uint8
Country_Philippines                    uint8
Country_Poland                         uint8
Country_Romania                        uint8
Country_Russian Federation             uint8
Country_Saudi Arabia                   uint8
Country_South Africa                   uint8
Country_Spain                          uint8
Country_Thailand                       uint8
Country_Turkey                         uint8
Country_Ukraine                        uint8
Country_United Kingdom                 uint8
Country_United States                  uint8
Country_Vietnam                        uint8
Product Type_BEVERAGE                  uint8
Product Type_FOOD                      uint8
Global Brand Name_BRAND11              uint8
Global Brand Name_BRAND13              uint8
Global Brand Name_BRAND2               uint8
Global Brand Name_BRAND3               uint8
Global Brand Name_BRAND40              uint8
Global Brand Name_BRAND589             uint8
Global Brand Name_BRAND590             uint8
Global Brand Name_BRAND591             uint8
Global Brand Name_BRAND6               uint8
Global Brand Name_BRAND7               uint8
Global Brand Name_BRAND77              uint8
Global Brand Name_BRAND9               uint8
dtype: object
```

In [96]:

```
df_addedsugar_ohe.head()
```

Out[96]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal	Prot
0	488	28	13	3	2	26	1	554	1	
1	499	26	12	2	3	41	2	612	1	
2	491	25	12	2	4	62	4	808	2	
3	502	30	13	3	1	9	0	600	1	
4	460	25	11	2	7	99	6	540	1	

5 rows × 50 columns

In [97]:
```
X = df_addedsugar_ohe.drop('AddedSugarClass', axis = 1)
y = df_addedsugar_ohe['AddedSugarClass']
```

In [98]:
```
key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','Ran
value = [LogisticRegression(random_state=9), KNeighborsClassifier(), SVC(), DecisionTr
models = dict(zip(key,value))
```

In [99]:
```
#importing train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=42,sh
```

In [100…
```
predicted =[]
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state =
```

In [101…
```
for name,algo in models.items():
    model=algo
    model.fit(X_train,y_train)
    predict = model.predict(X_test)
    acc = accuracy_score(y_test, predict)
    predicted.append(acc)
    print(name,acc)
```

```
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression 0.9526551503518874
KNeighborsClassifier 0.9982725527831094
SVC 0.9886116442738324
DecisionTreeClassifier 0.999872040946897
RandomForestClassifier 0.999872040946897
GradientBoostingClassifier 0.999872040946897
AdaBoostClassifier 0.999872040946897
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [101], in <cell line: 1>()
      1 for name,algo in models.items():
      2     model=algo
----> 3     model.fit(X_train,y_train)
      4     predict = model.predict(X_test)
      5     acc = accuracy_score(y_test, predict)

File ~\anaconda3\lib\site-packages\xgboost\core.py:575, in _deprecate_positional_args.<locals>.inner_f(*args, **kwargs)
    573 for k, arg in zip(sig.parameters, args):
    574     kwargs[k] = arg
--> 575 return f(**kwargs)

File ~\anaconda3\lib\site-packages\xgboost\sklearn.py:1357, in XGBClassifier.fit(self, X, y, sample_weight, base_margin, eval_set, eval_metric, early_stopping_rounds, verbose, xgb_model, sample_weight_eval_set, base_margin_eval_set, feature_weights, callbacks)
   1352     expected_classes = np.arange(self.n_classes_)
   1353 if (
   1354     self.classes_.shape != expected_classes.shape
   1355     or not (self.classes_ == expected_classes).all()
   1356 ):
-> 1357     raise ValueError(
   1358         f"Invalid classes inferred from unique values of `y`.  "
   1359         f"Expected: {expected_classes}, got {self.classes_}"
   1360     )
   1362 params = self.get_xgb_params()
   1364 if callable(self.objective):

ValueError: Invalid classes inferred from unique values of `y`.  Expected: [0 1 2], got ['1' '2' '3']
```

In [102…]
```
algo_tests = list(zip(predicted,key))
algo_tests=pd.DataFrame(algo_tests, columns=['predicted','key'])
algo_tests.head(5)
```

Out[102]:

	predicted	key
0	0.952655	LogisticRegression
1	0.998273	KNeighborsClassifier
2	0.988612	SVC
3	0.999872	DecisionTreeClassifier
4	0.999872	RandomForestClassifier

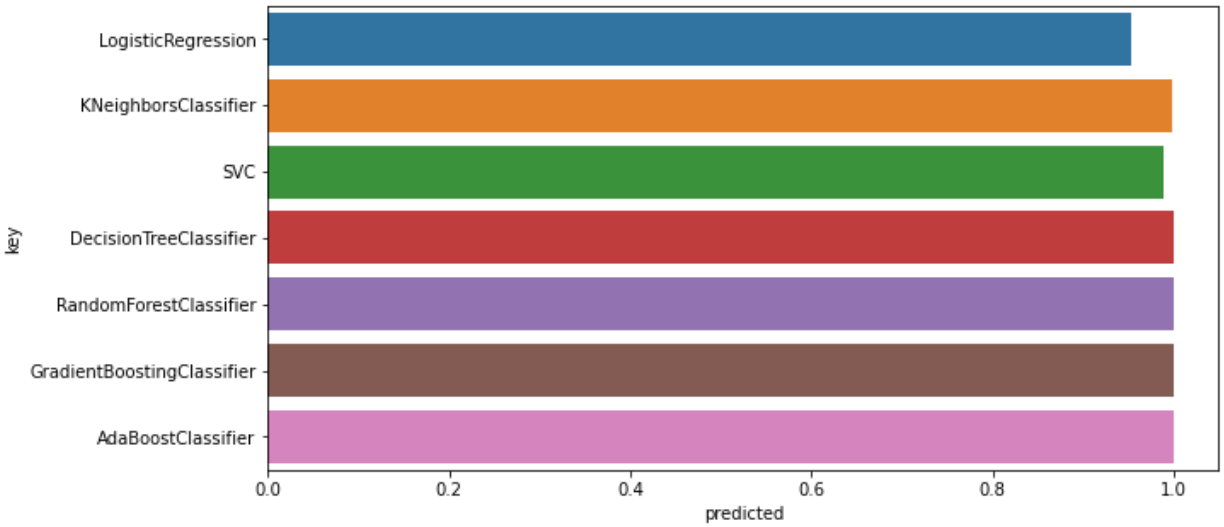
In [103…]
```
algo_tests
```

Out[103]:

	predicted	key
0	0.952655	LogisticRegression
1	0.998273	KNeighborsClassifier
2	0.988612	SVC
3	0.999872	DecisionTreeClassifier
4	0.999872	RandomForestClassifier
5	0.999872	GradientBoostingClassifier
6	0.999872	AdaBoostClassifier

In [104…

```
plt.figure(figsize = (10,5))
ax=sns.barplot(x = 'predicted', y = 'key', data=algo_tests)
```



In [105…

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

error_list = list()

# Iterate through various possibilities for number of trees
tree_list = [1, 2, 3, 4, 5]
for n_trees in tree_list:

    # Initialize the gradient boost classifier
    GBC = GradientBoostingClassifier(n_estimators=n_trees, random_state=42)

    # Fit the model
    print(f'Fitting model with {n_trees} trees')
    GBC.fit(X_train.values, y_train.values)
    y_pred = GBC.predict(X_test)

    # Get the error
    error = 1.0 - accuracy_score(y_test, y_pred)

    # Store it
    error_list.append(pd.Series({'n_trees': n_trees, 'error': error}))

error_df = pd.concat(error_list, axis=1).T.set_index('n_trees')
```

```
error_df
```

```
Fitting model with 1 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 2 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 3 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 4 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 5 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
```

Out[105]:

n_trees	error
1.0	0.362188
2.0	0.362188
3.0	0.122841
4.0	0.000128
5.0	0.000128

In [106…

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
```

In [107…

```
dt.tree_.node_count, dt.tree_.max_depth
```

Out[107]:

```
(13, 6)
```

In [108…

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
                      'recall': recall_score(y_true, y_pred),
                      'f1': f1_score(y_true, y_pred)},
                     name=label)
```

In [109…

```
# The error on the training and test data sets
y_train_pred = dt.predict(X_train)
```

```
y_test_pred = dt.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
### END SOLUTION
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [109], in <cell line: 5>()
      2 y_train_pred = dt.predict(X_train)
      3 y_test_pred = dt.predict(X_test)
----> 5 train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'trai
n'),
      6                                   measure_error(y_test, y_test_pred, 'test')],
      7                                   axis=1)
      9 train_test_full_error

Input In [108], in measure_error(y_true, y_pred, label)
      3 def measure_error(y_true, y_pred, label):
      4     return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
----> 5                       'precision': precision_score(y_true, y_pred),
      6                       'recall': recall_score(y_true, y_pred),
      7                       'f1': f1_score(y_true, y_pred)},
      8                      name=label)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1757, in precis
ion_score(y_true, y_pred, labels, pos_label, average, sample_weight, zero_division)
   1628 def precision_score(
   1629     y_true,
   1630     y_pred,
   (...)
   1636     zero_division="warn",
   1637 ):
   1638     """Compute the precision.
   1639
   1640     The precision is the ratio ``tp / (tp + fp)`` where ``tp`` is the number
 of
   (...)
   1755     array([0.5, 1. , 1. ])
   1756     """
-> 1757     p, _, _, _ = precision_recall_fscore_support(
   1758         y_true,
   1759         y_pred,
   1760         labels=labels,
   1761         pos_label=pos_label,
   1762         average=average,
   1763         warn_for=("precision",),
   1764         sample_weight=sample_weight,
   1765         zero_division=zero_division,
   1766     )
   1767     return p

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1544, in precis
ion_recall_fscore_support(y_true, y_pred, beta, labels, pos_label, average, warn_for,
sample_weight, zero_division)
   1542 if beta < 0:
   1543     raise ValueError("beta should be >=0 in the F-beta score")
-> 1544 labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1546 # Calculate tp_sum, pred_sum, true_sum ###
   1547 samplewise = average == "samples"

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1365, in _check
_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1363         if y_type == "multiclass":
   1364             average_options.remove("samples")
-> 1365         raise ValueError(
```

```
   1366             "Target is %s but average='binary'. Please "
   1367             "choose another average setting, one of %r." % (y_type, average_o
ptions)
   1368         )
   1369 elif pos_label not in (None, 1):
   1370     warnings.warn(
   1371         "Note that pos_label (set to %r) is ignored when "
   1372         "average != 'binary' (got %r). You may use "
   (...)
   1375         UserWarning,
   1376     )

ValueError: Target is multiclass but average='binary'. Please choose another average
 setting, one of [None, 'micro', 'macro', 'weighted'].
```

In [110…
```
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

In [111…
```
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

Out[111]:
```
(13, 5)
```

In [112…
```
y_train_pred_gr = GR.predict(X_train)
y_test_pred_gr = GR.predict(X_test)

train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr, 'train'),
                                 measure_error(y_test, y_test_pred_gr, 'test')],
                                axis=1)
train_test_gr_error
```

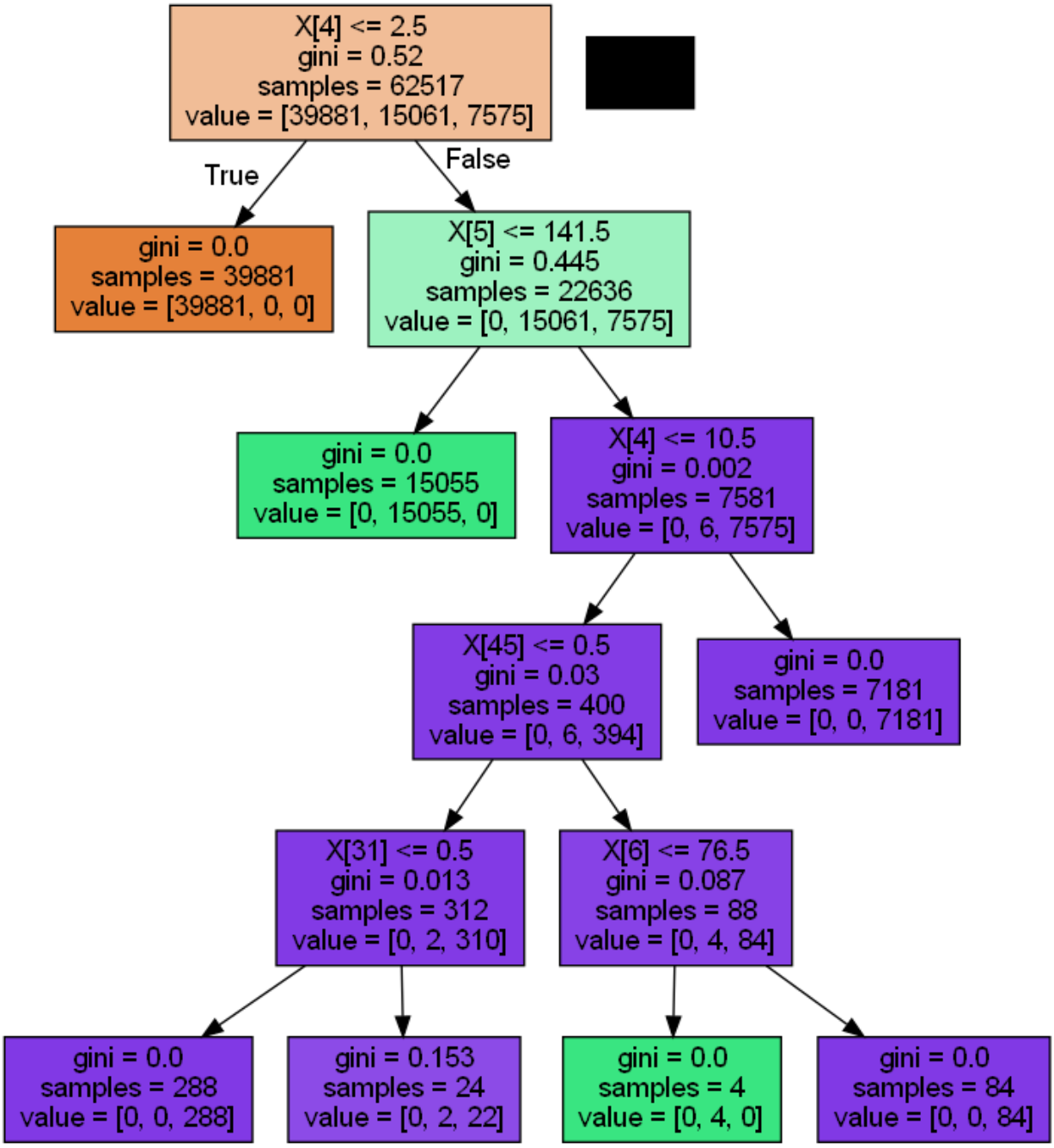
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [112], in <cell line: 4>()
      1 y_train_pred_gr = GR.predict(X_train)
      2 y_test_pred_gr = GR.predict(X_test)
----> 4 train_test_gr_error = pd.concat([measure_error(y_train, y_train_pred_gr, 'tra
in'),
      5                                 measure_error(y_test, y_test_pred_gr, 'test'
)],
      6                                axis=1)
      7 train_test_gr_error

Input In [108], in measure_error(y_true, y_pred, label)
      3 def measure_error(y_true, y_pred, label):
      4     return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
----> 5                       'precision': precision_score(y_true, y_pred),
      6                       'recall': recall_score(y_true, y_pred),
      7                       'f1': f1_score(y_true, y_pred)},
      8                      name=label)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1757, in precis
ion_score(y_true, y_pred, labels, pos_label, average, sample_weight, zero_division)
   1628 def precision_score(
   1629     y_true,
   1630     y_pred,
   (...)
   1636     zero_division="warn",
   1637 ):
   1638     """Compute the precision.
   1639
   1640     The precision is the ratio ``tp / (tp + fp)`` where ``tp`` is the number
 of
   (...)
   1755     array([0.5, 1. , 1. ])
   1756     """
-> 1757     p, _, _, _ = precision_recall_fscore_support(
   1758         y_true,
   1759         y_pred,
   1760         labels=labels,
   1761         pos_label=pos_label,
   1762         average=average,
   1763         warn_for=("precision",),
   1764         sample_weight=sample_weight,
   1765         zero_division=zero_division,
   1766     )
   1767     return p

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1544, in precis
ion_recall_fscore_support(y_true, y_pred, beta, labels, pos_label, average, warn_for,
sample_weight, zero_division)
   1542 if beta < 0:
   1543     raise ValueError("beta should be >=0 in the F-beta score")
-> 1544 labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1546 # Calculate tp_sum, pred_sum, true_sum ###
   1547 samplewise = average == "samples"

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1365, in _check
_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1363         if y_type == "multiclass":
   1364             average_options.remove("samples")
```

```
-> 1365         raise ValueError(
   1366             "Target is %s but average='binary'. Please "
   1367             "choose another average setting, one of %r." % (y_type, average_o
ptions)
   1368         )
   1369 elif pos_label not in (None, 1):
   1370     warnings.warn(
   1371         "Note that pos_label (set to %r) is ignored when "
   1372         "average != 'binary' (got %r). You may use "
   (...)
   1375         UserWarning,
   1376     )

ValueError: Target is multiclass but average='binary'. Please choose another average
 setting, one of [None, 'micro', 'macro', 'weighted'].
```

In [113…

```
# Create an output destination for the file
dot_data = StringIO()

export_graphviz(GR.best_estimator_, out_file=dot_data, filled=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

# View the tree image
filename = 'Pepsicotrain2.png'
graph.write_png(filename)
Image(filename=filename)
### END SOLUTION
```

Out[113]:



In [115…
```
X = df_addedsugar_ohe.drop('AddedSugarClass', axis = 1)
y = df_addedsugar_ohe['AddedSugarClass']
#from GR.best_estimator
dt = DecisionTreeClassifier(max_depth=5, max_leaf_nodes=13)

model = dt.fit(X,y)
```

In [116…
```
pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\zahra\anaconda3\lib\site-packages
(0.20.1)
Note: you may need to restart the kernel to use updated packages.
```

In [117…
```
import graphviz
from graphviz import Source
dot_data = export_graphviz(model, out_file=None, feature_names=X.columns)
```

```
graph = graphviz.Source(dot_data)
graph.render("TrainDecisionTree",view = True)
```

Out[117]:
```
'TrainDecisionTree.pdf'
```

Validating model

Working with Test data set

In [118…
```
df2= pd.read_csv("TestingData.csv")
df2.head(5)
```

```
C:\Users\Zahra\AppData\Local\Temp\ipykernel_17468\173048071.py:1: DtypeWarning: Columns (40) have mixed types. Specify dtype option on import or set low_memory=False.
  df2= pd.read_csv("TestingData.csv")
```

Out[118]:

	Year	Quarter	Country	Local Material Description	Product ID	Local Brand Name	Local Sub-Brand Name	Product Type	Global Brand Name	Standard Brand Name
0	2021	Q1	Egypt	BRAND1 SALT 75G 11P 5LE	26552831	BRAND1	BRAND1	FOOD	BRAND7	BRAND1
1	2021	Q1	Egypt	BRAND1 BRAND613TO 10LE 6P 187G	26167968	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1
2	2021	Q1	Egypt	BRAND1 C AND O 10LE 6P 168G LOG	26612862	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1
3	2021	Q1	Egypt	BRAND1 SALT 10LE 6P 187G	26167969	BRAND1	BRAND1 Family	FOOD	BRAND7	BRAND1
4	2021	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	BRAND3	BRAND3 POPCORN	FOOD	BRAND3	BRAND3

5 rows × 200 columns

In [119…
```
df2.shape
```

Out[119]:
```
(35202, 200)
```

In [120…
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(df2.isnull().sum(axis = 0))
```

Year	0
Quarter	0
Country	0
Local Material Description	2
Product ID	0
Local Brand Name	39
Local Sub-Brand Name	9471
Product Type	0
Global Brand Name	0
Standard Brand Name	0
Standard Sub-Brand Name	0
Category Name	17798
Sub-Category Name	18596
Segment Name	18877
Sub-Segment Name	20120
Volume Units	4806
Flavor	4374
Standard Flavor Name	19181
Kilojoules	20313
Total Calories	4376
Total Fat	5091
Calories from Fat	30049
Calories From Saturated Fat	32236
Calories - Canada	33147
Saturated Fat	4376
Saturated Fat per 100 kcals	4376
Trans Fatty Acids	5786
Monounsaturated Fats	29202
Polyunsaturated Fat	29216
Cholesterol	17629
Omega 3 Fatty Acids	32163
Omega 6 Fatty Acids	32163
Linolenic Acid	34374
DHA	33329
Carbohydrate Total	12182
Carbohydrates using Difference Method	33346
Roll-Up Of Component Carbohydrates	34391
Other Carbohydrates/Starch	30941
Sugars	9263
Added Sugars	6215
Calories from Added Sugars per 355ml	6215
% of Energy from Added Sugars	6215
Sucrose	33329
Fructose	33260
Glucose	33329
Sodium	4377
Sodium per kcal	4377
Sodium Excluding Contribution From Plant Water Ingredient	34686
Sodium (Historical Value) MG	35202
Sodium (Including Canada Water)	34391
Sodium (Including US Water)	34161
Sodium From Components	35202
Sodium In Water (Canada)	33350
Sodium In Water (US)	33346
Salt	25566
Protein	5159
Dietary Fiber	7590
Dietary Fiber Not Allowable By US FDA	33355
Total Dietary Fiber - US FDA	33346
Insoluble Fiber	33288

Insoluble Fiber - US FDA	33346
Insoluble Fiber Not Allowable By US FDA	33360
Soluble Fiber	30617
Soluble Fiber - US FDA	33346
Soluble Fiber Not Allowable By US FDA	33360
Wheat Bran Fiber	33660
Oat Content	33473
Gluten	35202
Beta Glucan	33325
Whole Grains	18746
Total Grains	23709
Fruits - Solids	20900
Fruits - Liquids	20326
Vegetables - Solids	21022
Vegetables - Liquids	21081
Total Dairy Products	21054
Low Fat Dairy	24369
Nuts & Seeds	22742
Legumes	24424
Beta Carotene	32731
Beta Carotene, Prior To Loss Factor	34386
Historical Value: Beta Carotene	33772
Historical Value: Beta Carotene, Prior To Loss Factor	34484
Biotin	31623
Biotin, Prior To Loss Factor	34384
Calcium	17815
Calcium, Prior To Loss Factor	34384
Chloride	33329
Chloride, Prior To Loss Factor	34384
Total Choline	32855
Chromium	31726
Chromium, Prior To Loss Factor	34384
Copper	31525
Copper, Prior To Loss Factor	34384
Folate	25327
Folate, Prior To Loss Factor	34386
Folic Acid	34323
Folic Acid, Prior To Loss Factor	34389
Folic Acid (Synthetic)	33350
Folic Acid (Synthetic), Prior To Loss Factor	34386
Iodine	32263
Iodine, Prior To Loss Factor	34384
Iron	18899
Iron, Prior To Loss Factor	34384
Manganese	31529
Manganese, Prior To Loss Factor	34384
Magnesium	23476
Magnesium, Prior To Loss Factor	34384
Molybdenum	32405
Molybdenum, Prior To Process Loss	34384
Niacin	30604
Niacin Equivalents	30903
Niacin, Prior To Loss Factor	35195
Pantothenic Acid	31538
Pantothenic Acid, Prior To Loss Factor	34384
Phosphorus	30380
Phosphorus, Prior To Loss Factor	34384
Potassium	19529
Riboflavin	30682
Vitamin B2-Riboflavin, Prior To Loss Factor	34384

Selenium	31531
Selenium, Prior To Loss Factor	34384
Thiamin	30566
Vitamin B1-Thiamin, Prior To Loss Factor	34384
Vitamin A	22622
Vitamin A, Prior To Loss Factor	34384
Historical Value: Vitamin A	33504
Historical Value: Vitamin A, Prior To Loss Factor	34475
Vitamin A (RAE)	30957
Vitamin A, Prior To Loss Factor (RAE)	34391
Vitamin B12	30674
Vitamin B12, Prior To Loss Factor	34384
Vitamin B6	31152
Vitamin B6, Prior To Loss Factor	34384
Vitamin C	21238
Vitamin C, Prior To Loss Factor	34384
Vitamin D	19986
Vitamin D, Prior To Loss Factor	34384
Historical Value: Vitamin D	33576
Historical Value: Vitamin D, Prior To Loss Factor	34547
Vitamin E	23874
Vitamin E, Prior To Loss Facto	34384
Historical Value: Vitamin E	33513
Historical Value: Vitamin E (IU), Prior To Loss Factor	35202
Vitamin K	31802
Vitamin K, Prior To Loss Factor	34384
Zinc	23917
Zinc, Prior To Loss Factor	34384
Acesulfame Potassium	33329
Aspartame	33329
Erythritol	33329
Isomalt	33350
Lactitol	33350
Maltitol	33350
Maltitol Syrup	34364
Mannitol	33350
Rebaudioside A	33329
Saccharin	33329
Sorbitol	33350
Sorbitol Syrup	34364
Sucralose	33329
Sorbic Acid	33329
Stearic Acid	35202
Steviol Glycosides	34391
Tagatose	33329
Xylitol	33350
Flavonoids	33329
Hydrogenated Starch Hydrolysales	33350
Inositol	35202
Moisture	34374
Moisture, Pre-Adjusted	34391
Caffeine	31596
Alcohol	32199
Sugar Alcohol	33324
Sugar Alcohol - Canada	33350
Formula Alcohol	34391
Benzoic Acid	33329
Ash	33329
Histidine	33329
Isoleucine	33327

```
Leucine                                33322
Lysine                                 33329
Methionine + Cystine                   33329
Proline                                33329
Taurine                                33322
Threonine                              33329
Tryptophan                             33329
Valine                                 33322
Carotenoid                             32495
Disaccharides                          32536
Flouride                               33419
Glycerol                               33660
Inulin                                 33660
L-Arginine                             35202
L-Carnitine                            35202
Monosaccharides                        32544
Neotame                                34364
Nitrogen                               34157
Phenylalanine + Tyrosine               33346
Polydextrose                           33660
dtype: int64
```

In [121…
```
df3=df2.dropna(subset=['Added Sugars', 'Calories from Added Sugars per 355ml','% of Er
df3.shape
```

Out[121]:
```
(28987, 200)
```

In [122…
```
df3_1000=pd.isnull(df1).sum() < 1000
columns=df3.columns[df1.isnull().sum() < 1000].tolist()
df4=df3[columns]
df_addedsugar2=df4.reset_index()
df_addedsugar2.head()
```

Out[122]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2021	Q1	Egypt	BRAND1 SALT 75G 11P 5LE	26552831	FOOD	BRAND7	BRAND1	BRAND1
1	1	2021	Q1	Egypt	BRAND1 BRAND613TO 10LE 6P 187G	26167968	FOOD	BRAND7	BRAND1	BRAND1
2	3	2021	Q1	Egypt	BRAND1 SALT 10LE 6P 187G	26167969	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
3	4	2021	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN
4	5	2021	Q1	Egypt	BRAND3 BRAND70 CHEDAR CHZ 64G 9P 5LE	25416200	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

5 rows × 21 columns

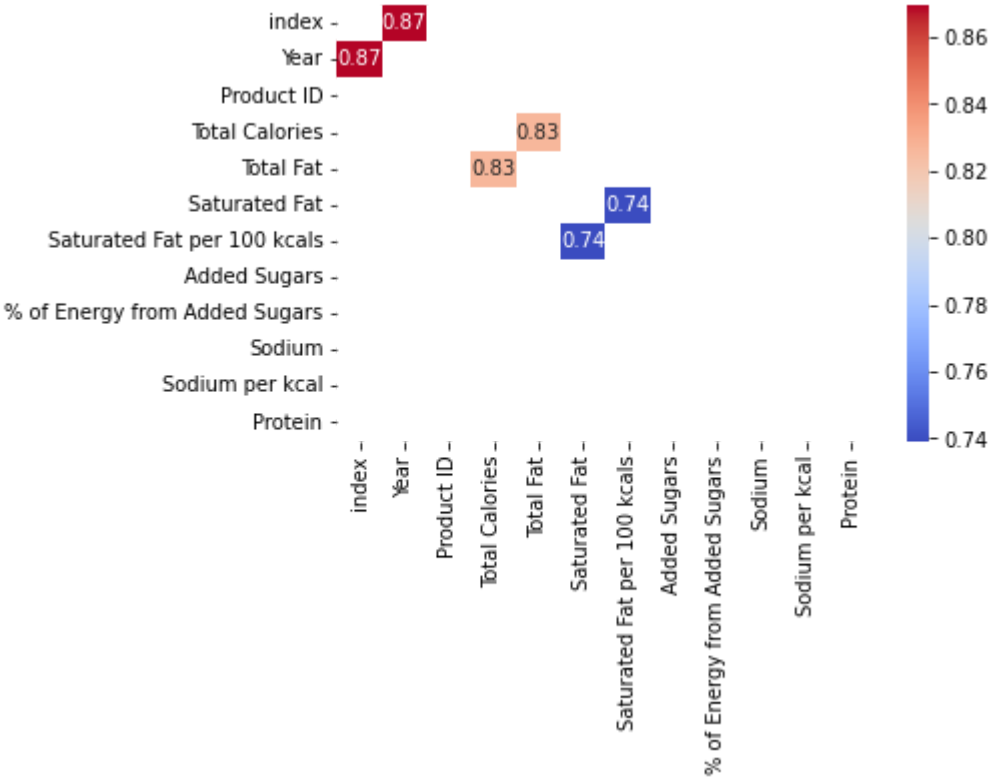
In [123…
```
df_addedsugar2.isnull().sum(axis = 0)
```

Out[123]:
```
index                                  0
Year                                   0
Quarter                                0
Country                                0
Local Material Description             2
Product ID                             0
Product Type                           0
Global Brand Name                      0
Standard Brand Name                    0
Standard Sub-Brand Name                0
Flavor                                 0
Total Calories                         0
Total Fat                            406
Saturated Fat                          0
Saturated Fat per 100 kcals            0
Added Sugars                           0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 1
Sodium per kcal                        1
Protein                              474
dtype: int64
```

In [124…
```
df_addedsugar.fillna(0, inplace=True)
df_addedsugar.isnull().sum(axis = 0)
```

Out[124]:
```
Country                                  0
Product Type                             0
Global Brand Name                        0
TotalCalories                            0
Total Fat                                0
Saturated Fat                            0
Saturated Fat per 100 kcals              0
AddedSugars                              0
Calories from Added Sugars per 355ml     0
% of Energy from Added Sugars            0
Sodium                                   0
Sodium per kcal                          0
Protein                                  0
AddedSugarClass                          0
dtype: int64
```

In [125…
```
dfCorr1 =df_addedsugar2.corr()
filteredDf1 = dfCorr1[((dfCorr1 >= .5) | (dfCorr1 <= -.5)) & (dfCorr1 !=1.000)]
sns.heatmap(filteredDf1, annot=True, cmap="coolwarm")
plt.figure(figsize=(40,20))
plt.show()
```




```
<Figure size 2880x1440 with 0 Axes>
```

Correlation results:

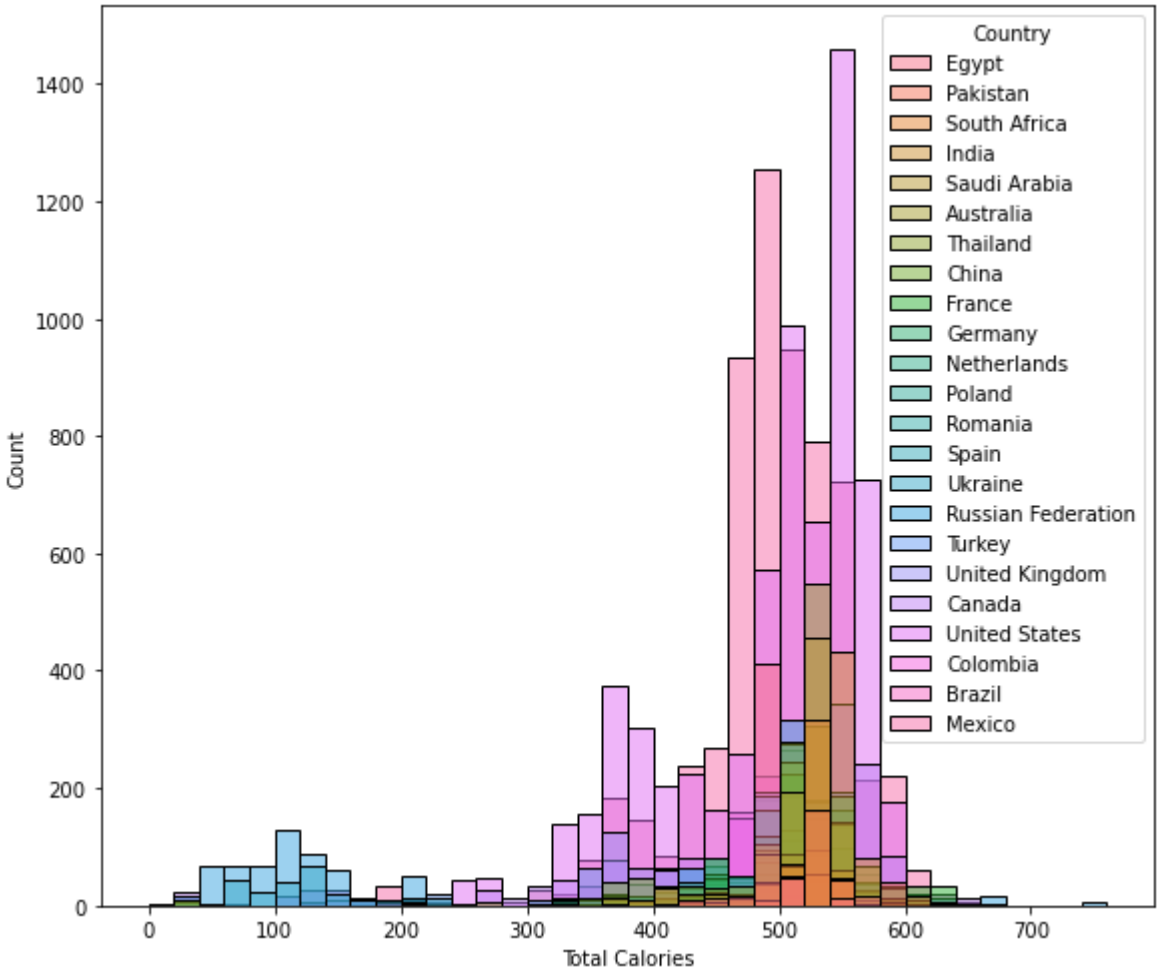
1. 83% of Total Calories corelates with total fat in the products,

2. 74% of Saturated fat corelates with saturated fat per 100 kcals in the products,

Different from the training data.

In [126…
```
a4_dims = (11.7, 8.27)
```

```
fig, ax = plt.subplots(figsize=a4_dims)
sns.histplot(ax=ax,data=df_addedsugar2, x="Total Calories", hue="Country", binwidth=20
plt.subplots_adjust(right=0.75)
plt.show()
```



In [127…

```
df_addedsugar2=df_addedsugar2.rename(columns={"Added Sugars": "AddedSugars"})
filter_method = lambda AddedSugars: 'low' if AddedSugars <= 2.5 else 'mid' if (AddedSu
df_addedsugar2['AddedSugarClass'] = df_addedsugar2['AddedSugars'].apply(filter_method)
df_addedsugar2.head()
```

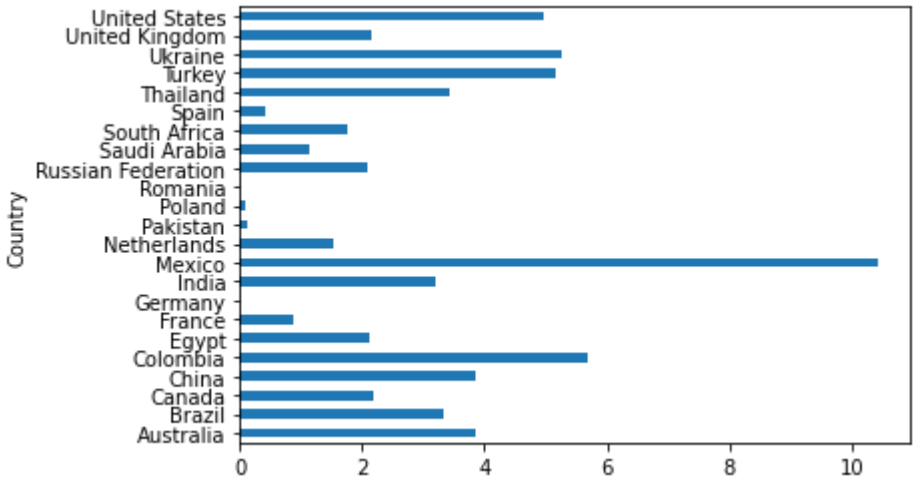
Out[127]:

	index	Year	Quarter	Country	Local Material Description	Product ID	Product Type	Global Brand Name	Standard Brand Name	Standard Sub-Brand Name
0	0	2021	Q1	Egypt	BRAND1 SALT 75G 11P 5LE	26552831	FOOD	BRAND7	BRAND1	BRAND1
1	1	2021	Q1	Egypt	BRAND1 BRAND613TO 10LE 6P 187G	26167968	FOOD	BRAND7	BRAND1	BRAND1
2	3	2021	Q1	Egypt	BRAND1 SALT 10LE 6P 187G	26167969	FOOD	BRAND7	BRAND1	BRAND1 FAMILY
3	4	2021	Q1	Egypt	BRAND3 BRAND70 SPICY BRAND613TO 64G 9P 5LE	25416202	FOOD	BRAND3	BRAND3	BRAND3 POPCORN
4	5	2021	Q1	Egypt	BRAND3 BRAND70 CHEDAR CHZ 64G 9P 5LE	25416200	FOOD	BRAND3	BRAND3	BRAND3 POPCORN

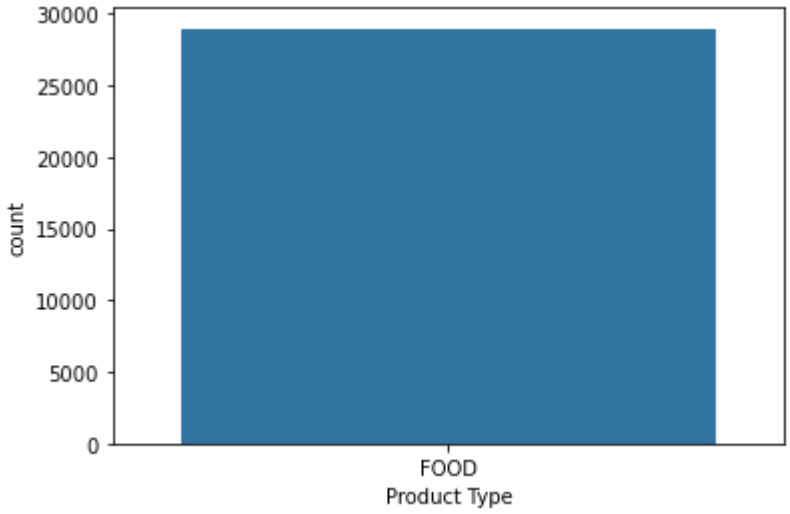
5 rows × 22 columns

In [128…
```
df_addedsugar2.groupby('Country').AddedSugars.mean().plot(kind='barh')
```

Out[128]:
```
<AxesSubplot:ylabel='Country'>
```



In [129…
```
p8 = sns.countplot(data=df_addedsugar2, x="Product Type")
plt.show()
```



No product type of beverage in the testing data

In [130…
```
df_addedsugar2=df_addedsugar2.drop(['index', 'Year', 'Quarter', 'Local Material Descri
df_addedsugar2.head()
```

Out[130]:

	Country	Product Type	Global Brand Name	Total Calories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars
0	Egypt	FOOD	BRAND7	502.0	30.0	13.300	2.65	0.6	8.52	0.48
1	Egypt	FOOD	BRAND7	495.0	28.2	12.566	2.54	5.8	82.36	4.69
2	Egypt	FOOD	BRAND7	502.0	30.0	13.300	2.65	0.6	8.52	0.48
3	Egypt	FOOD	BRAND3	460.0	25.0	11.000	2.39	7.0	99.4	6.09
4	Egypt	FOOD	BRAND3	460.0	26.0	11.000	2.39	1.0	14.2	0.87

In [131…
```
df_addedsugar2_ohe=df_addedsugar
categorical_cols=['Country', 'Product Type', 'Global Brand Name']
for col in categorical_cols:
    col_ohe = pd.get_dummies(df_addedsugar2[col], prefix=col)
    df_addedsugar2_ohe = pd.concat((df_addedsugar2_ohe , col_ohe), axis=1).drop(col, a
```

In [132…
```
print(df_addedsugar2_ohe.columns)
```

```
Index(['TotalCalories', 'Total Fat', 'Saturated Fat',
       'Saturated Fat per 100 kcals', 'AddedSugars',
       'Calories from Added Sugars per 355ml', '% of Energy from Added Sugars',
       'Sodium', 'Sodium per kcal', 'Protein', 'AddedSugarClass',
       'Country_Australia', 'Country_Brazil', 'Country_Canada',
       'Country_China', 'Country_Colombia', 'Country_Egypt', 'Country_France',
       'Country_Germany', 'Country_India', 'Country_Mexico',
       'Country_Netherlands', 'Country_Pakistan', 'Country_Poland',
       'Country_Romania', 'Country_Russian Federation', 'Country_Saudi Arabia',
       'Country_South Africa', 'Country_Spain', 'Country_Thailand',
       'Country_Turkey', 'Country_Ukraine', 'Country_United Kingdom',
       'Country_United States', 'Product Type_FOOD',
       'Global Brand Name_BRAND11', 'Global Brand Name_BRAND13',
       'Global Brand Name_BRAND2', 'Global Brand Name_BRAND3',
       'Global Brand Name_BRAND40', 'Global Brand Name_BRAND589',
       'Global Brand Name_BRAND590', 'Global Brand Name_BRAND591',
       'Global Brand Name_BRAND7', 'Global Brand Name_BRAND77',
       'Global Brand Name_BRAND9'],
      dtype='object')
```

In [133…
```
#Convert added sugars class to 1, 2 and 3 for low, mid and high.
df_addedsugar2_ohe["AddedSugarClass"].replace('low', '1',inplace=True)
df_addedsugar2_ohe["AddedSugarClass"].replace('mid', '2',inplace=True)
df_addedsugar2_ohe["AddedSugarClass"].replace('high', '3',inplace=True)
```

In [134…
```
df_addedsugar2_ohe.head()
```

Out[134]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal	Prot
0	488.0	28.0	13.0	2.66	1.8	25.56	1.48	554.0	1.1352	
1	499.0	26.0	12.0	2.40	2.9	41.18	2.32	612.0	1.2265	
2	491.0	25.0	12.0	2.44	4.4	62.48	3.58	808.5	1.6466	
3	502.0	30.0	13.3	2.65	0.6	8.52	0.48	600.0	1.1952	
4	460.0	25.0	11.0	2.39	7.0	99.40	6.09	540.0	1.1739	

5 rows × 46 columns

Round and turn dtype to integers for the testing data

In [135…
```
cols2num = ['TotalCalories','Total Fat','Saturated Fat','Saturated Fat per 100 kcals',
df_addedsugar2_ohe[cols2num] = df_addedsugar2_ohe[cols2num].round(0).astype(int)

#df_addedsugar2_ohe['Age'].mean().round(0).astype(int)
```

In [136…
```
df_addedsugar2_ohe.dtypes
```

Out[136]:

```
TotalCalories                            int32
Total Fat                                int32
Saturated Fat                            int32
Saturated Fat per 100 kcals              int32
AddedSugars                              int32
Calories from Added Sugars per 355ml     int32
% of Energy from Added Sugars            int32
Sodium                                   int32
Sodium per kcal                          int32
Protein                                  int32
AddedSugarClass                         object
Country_Australia                      float64
Country_Brazil                         float64
Country_Canada                         float64
Country_China                          float64
Country_Colombia                       float64
Country_Egypt                          float64
Country_France                         float64
Country_Germany                        float64
Country_India                          float64
Country_Mexico                         float64
Country_Netherlands                    float64
Country_Pakistan                       float64
Country_Poland                         float64
Country_Romania                        float64
Country_Russian Federation             float64
Country_Saudi Arabia                   float64
Country_South Africa                   float64
Country_Spain                          float64
Country_Thailand                       float64
Country_Turkey                         float64
Country_Ukraine                        float64
Country_United Kingdom                 float64
Country_United States                  float64
Product Type_FOOD                      float64
Global Brand Name_BRAND11              float64
Global Brand Name_BRAND13              float64
Global Brand Name_BRAND2               float64
Global Brand Name_BRAND3               float64
Global Brand Name_BRAND40              float64
Global Brand Name_BRAND589             float64
Global Brand Name_BRAND590             float64
Global Brand Name_BRAND591             float64
Global Brand Name_BRAND7               float64
Global Brand Name_BRAND77              float64
Global Brand Name_BRAND9               float64
dtype: object
```

In [137…

```
df_addedsugar2_ohe.head()
```

Out[137]:

	TotalCalories	Total Fat	Saturated Fat	Saturated Fat per 100 kcals	AddedSugars	Calories from Added Sugars per 355ml	% of Energy from Added Sugars	Sodium	Sodium per kcal	Prot
0	488	28	13	3	2	26	1	554	1	
1	499	26	12	2	3	41	2	612	1	
2	491	25	12	2	4	62	4	808	2	
3	502	30	13	3	1	9	0	600	1	
4	460	25	11	2	7	99	6	540	1	

5 rows × 46 columns

In [138…
```
df_addedsugar2_ohe['Product Type_FOOD'].value_counts()
```

Out[138]:
```
1.0    28987
Name: Product Type_FOOD, dtype: int64
```

In [139…
```
df_addedsugar2_ohe.shape
```

Out[139]:
```
(78147, 46)
```

In [140…
```
df_addedsugar2_ohe.isnull().sum(axis = 0)
```

Out[140]:

```
TotalCalories                          0
Total Fat                              0
Saturated Fat                          0
Saturated Fat per 100 kcals            0
AddedSugars                            0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 0
Sodium per kcal                        0
Protein                                0
AddedSugarClass                        0
Country_Australia                  49160
Country_Brazil                     49160
Country_Canada                     49160
Country_China                      49160
Country_Colombia                   49160
Country_Egypt                      49160
Country_France                     49160
Country_Germany                    49160
Country_India                      49160
Country_Mexico                     49160
Country_Netherlands                49160
Country_Pakistan                   49160
Country_Poland                     49160
Country_Romania                    49160
Country_Russian Federation         49160
Country_Saudi Arabia               49160
Country_South Africa               49160
Country_Spain                      49160
Country_Thailand                   49160
Country_Turkey                     49160
Country_Ukraine                    49160
Country_United Kingdom             49160
Country_United States              49160
Product Type_FOOD                  49160
Global Brand Name_BRAND11          49160
Global Brand Name_BRAND13          49160
Global Brand Name_BRAND2           49160
Global Brand Name_BRAND3           49160
Global Brand Name_BRAND40          49160
Global Brand Name_BRAND589         49160
Global Brand Name_BRAND590         49160
Global Brand Name_BRAND591         49160
Global Brand Name_BRAND7           49160
Global Brand Name_BRAND77          49160
Global Brand Name_BRAND9           49160
dtype: int64
```

In [141…

```
df_addedsugar2_ohe.fillna(0, inplace=True)
df_addedsugar2_ohe.isnull().sum(axis = 0)
```

Out[141]:
```
TotalCalories                          0
Total Fat                              0
Saturated Fat                          0
Saturated Fat per 100 kcals            0
AddedSugars                            0
Calories from Added Sugars per 355ml   0
% of Energy from Added Sugars          0
Sodium                                 0
Sodium per kcal                        0
Protein                                0
AddedSugarClass                        0
Country_Australia                      0
Country_Brazil                         0
Country_Canada                         0
Country_China                          0
Country_Colombia                       0
Country_Egypt                          0
Country_France                         0
Country_Germany                        0
Country_India                          0
Country_Mexico                         0
Country_Netherlands                    0
Country_Pakistan                       0
Country_Poland                         0
Country_Romania                        0
Country_Russian Federation             0
Country_Saudi Arabia                   0
Country_South Africa                   0
Country_Spain                          0
Country_Thailand                       0
Country_Turkey                         0
Country_Ukraine                        0
Country_United Kingdom                 0
Country_United States                  0
Product Type_FOOD                      0
Global Brand Name_BRAND11              0
Global Brand Name_BRAND13              0
Global Brand Name_BRAND2               0
Global Brand Name_BRAND3               0
Global Brand Name_BRAND40              0
Global Brand Name_BRAND589             0
Global Brand Name_BRAND590             0
Global Brand Name_BRAND591             0
Global Brand Name_BRAND7               0
Global Brand Name_BRAND77              0
Global Brand Name_BRAND9               0
dtype: int64
```

In [142…
```
X = df_addedsugar2_ohe.drop('AddedSugarClass', axis = 1)
y = df_addedsugar2_ohe['AddedSugarClass']
```

In [143…
```
key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','Ran
value = [LogisticRegression(random_state=9), KNeighborsClassifier(), SVC(), DecisionTr
models = dict(zip(key,value))
```

In [144…
```
#importing train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=42,sh
```

In [145…
```
predicted =[]
```

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state =
```

In [146…

```
for name,algo in models.items():
    model=algo
    model.fit(X_train,y_train)
    predict = model.predict(X_test)
    acc = accuracy_score(y_test, predict)
    predicted.append(acc)
    print(name,acc)
```

```
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression 0.9513755598208573
KNeighborsClassifier 0.9982085732565579
SVC 0.9886116442738324
DecisionTreeClassifier 0.9996801023672425
RandomForestClassifier 0.9997440818937939
GradientBoostingClassifier 0.999872040946897
AdaBoostClassifier 0.999872040946897
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [146], in <cell line: 1>()
      1 for name,algo in models.items():
      2     model=algo
----> 3     model.fit(X_train,y_train)
      4     predict = model.predict(X_test)
      5     acc = accuracy_score(y_test, predict)

File ~\anaconda3\lib\site-packages\xgboost\core.py:575, in _deprecate_positional_arg
s.<locals>.inner_f(*args, **kwargs)
    573 for k, arg in zip(sig.parameters, args):
    574     kwargs[k] = arg
--> 575 return f(**kwargs)

File ~\anaconda3\lib\site-packages\xgboost\sklearn.py:1357, in XGBClassifier.fit(sel
f, X, y, sample_weight, base_margin, eval_set, eval_metric, early_stopping_rounds, ve
rbose, xgb_model, sample_weight_eval_set, base_margin_eval_set, feature_weights, call
backs)
   1352     expected_classes = np.arange(self.n_classes_)
   1353 if (
   1354     self.classes_.shape != expected_classes.shape
   1355     or not (self.classes_ == expected_classes).all()
   1356 ):
-> 1357     raise ValueError(
   1358         f"Invalid classes inferred from unique values of `y`.  "
   1359         f"Expected: {expected_classes}, got {self.classes_}"
   1360     )
   1362 params = self.get_xgb_params()
   1364 if callable(self.objective):

ValueError: Invalid classes inferred from unique values of `y`.  Expected: [0 1 2], g
ot ['1' '2' '3']
```

In [147…
```
algo_tests = list(zip(predicted,key))
algo_tests=pd.DataFrame(algo_tests, columns=['predicted','key'])
algo_tests.head(5)
```

Out[147]:

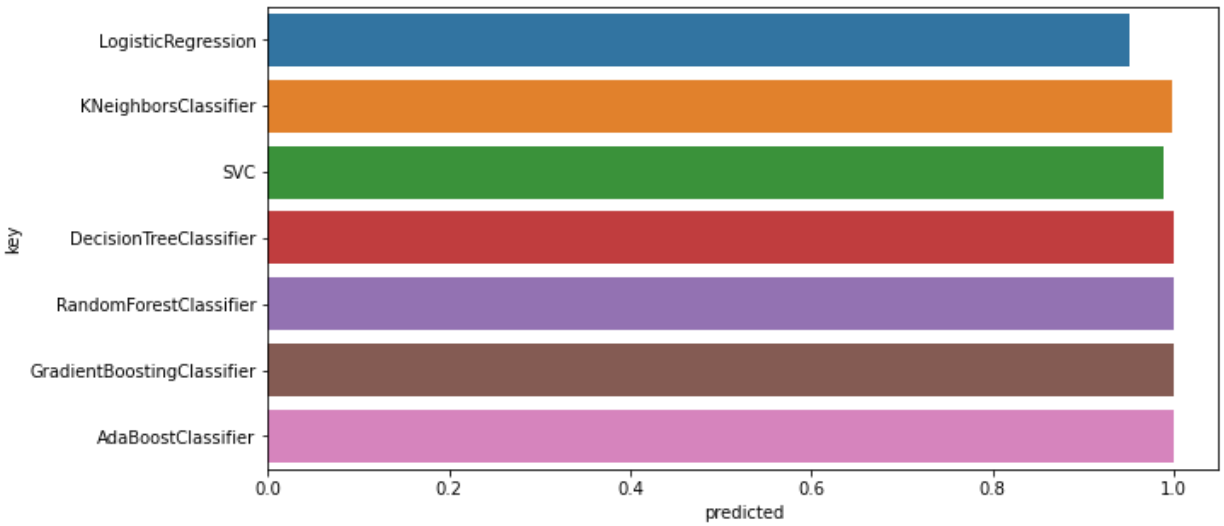
	predicted	key
0	0.951376	LogisticRegression
1	0.998209	KNeighborsClassifier
2	0.988612	SVC
3	0.999680	DecisionTreeClassifier
4	0.999744	RandomForestClassifier

In [148…
```
algo_tests
```

Out[148]:

	predicted	key
0	0.951376	LogisticRegression
1	0.998209	KNeighborsClassifier
2	0.988612	SVC
3	0.999680	DecisionTreeClassifier
4	0.999744	RandomForestClassifier
5	0.999872	GradientBoostingClassifier
6	0.999872	AdaBoostClassifier

In [149…
```
plt.figure(figsize = (10,5))
ax=sns.barplot(x = 'predicted', y = 'key', data=algo_tests)
```



In [150…
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

error_list = list()

# Iterate through various possibilities for number of trees
```

```
tree_list = [1, 2, 3, 4, 5, 6]
for n_trees in tree_list:

    # Initialize the gradient boost classifier
    GBC = GradientBoostingClassifier(n_estimators=n_trees, random_state=42)

    # Fit the model
    print(f'Fitting model with {n_trees} trees')
    GBC.fit(X_train.values, y_train.values)
    y_pred = GBC.predict(X_test)

    # Get the error
    error = 1.0 - accuracy_score(y_test, y_pred)

    # Store it
    error_list.append(pd.Series({'n_trees': n_trees, 'error': error}))

error_df = pd.concat(error_list, axis=1).T.set_index('n_trees')

error_df
```

```
Fitting model with 1 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 2 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 3 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 4 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 5 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
Fitting model with 6 trees
C:\Users\Zahra\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has fe
ature names, but GradientBoostingClassifier was fitted without feature names
  warnings.warn(
```

Out[150]:

n_trees	error
1.0	0.362188
2.0	0.362188
3.0	0.122841
4.0	0.000128
5.0	0.000128
6.0	0.000128

In [151…
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
```

In [152…
```
dt.tree_.node_count, dt.tree_.max_depth
```

Out[152]:
```
(21, 9)
```

In [153…
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
                      'recall': recall_score(y_true, y_pred),
                      'f1': f1_score(y_true, y_pred)},
                      name=label)
```

In [154…
```
# The error on the training and test data sets
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
### END SOLUTION
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [154], in <cell line: 5>()
      2 y_train_pred = dt.predict(X_train)
      3 y_test_pred = dt.predict(X_test)
----> 5 train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'trai
n'),
      6                                   measure_error(y_test, y_test_pred, 'test')],
      7                                   axis=1)
      9 train_test_full_error

Input In [153], in measure_error(y_true, y_pred, label)
      3 def measure_error(y_true, y_pred, label):
      4     return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
----> 5                       'precision': precision_score(y_true, y_pred),
      6                       'recall': recall_score(y_true, y_pred),
      7                       'f1': f1_score(y_true, y_pred)},
      8                      name=label)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1757, in precis
ion_score(y_true, y_pred, labels, pos_label, average, sample_weight, zero_division)
   1628 def precision_score(
   1629     y_true,
   1630     y_pred,
   (...)
   1636     zero_division="warn",
   1637 ):
   1638     """Compute the precision.
   1639
   1640     The precision is the ratio ``tp / (tp + fp)`` where ``tp`` is the number
 of
   (...)
   1755     array([0.5, 1. , 1. ])
   1756     """
-> 1757     p, _, _, _ = precision_recall_fscore_support(
   1758         y_true,
   1759         y_pred,
   1760         labels=labels,
   1761         pos_label=pos_label,
   1762         average=average,
   1763         warn_for=("precision",),
   1764         sample_weight=sample_weight,
   1765         zero_division=zero_division,
   1766     )
   1767     return p

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1544, in precis
ion_recall_fscore_support(y_true, y_pred, beta, labels, pos_label, average, warn_for,
sample_weight, zero_division)
   1542 if beta < 0:
   1543     raise ValueError("beta should be >=0 in the F-beta score")
-> 1544 labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1546 # Calculate tp_sum, pred_sum, true_sum ###
   1547 samplewise = average == "samples"

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1365, in _check
_set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1363         if y_type == "multiclass":
   1364             average_options.remove("samples")
-> 1365         raise ValueError(
```

```
   1366             "Target is %s but average='binary'. Please "
   1367             "choose another average setting, one of %r." % (y_type, average_o
ptions)
   1368         )
   1369 elif pos_label not in (None, 1):
   1370     warnings.warn(
   1371         "Note that pos_label (set to %r) is ignored when "
   1372         "average != 'binary' (got %r). You may use "
   (...)
   1375         UserWarning,
   1376     )

ValueError: Target is multiclass but average='binary'. Please choose another average
 setting, one of [None, 'micro', 'macro', 'weighted'].
```

In [155…

```
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth':range(1, dt.tree_.max_depth+1, 2),
              'max_features': range(1, len(dt.feature_importances_)+1)}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
                  param_grid=param_grid,
                  scoring='accuracy',
                  n_jobs=-1)

GR = GR.fit(X_train, y_train)
```

In [156…

```
GR.best_estimator_.tree_.node_count, GR.best_estimator_.tree_.max_depth
```

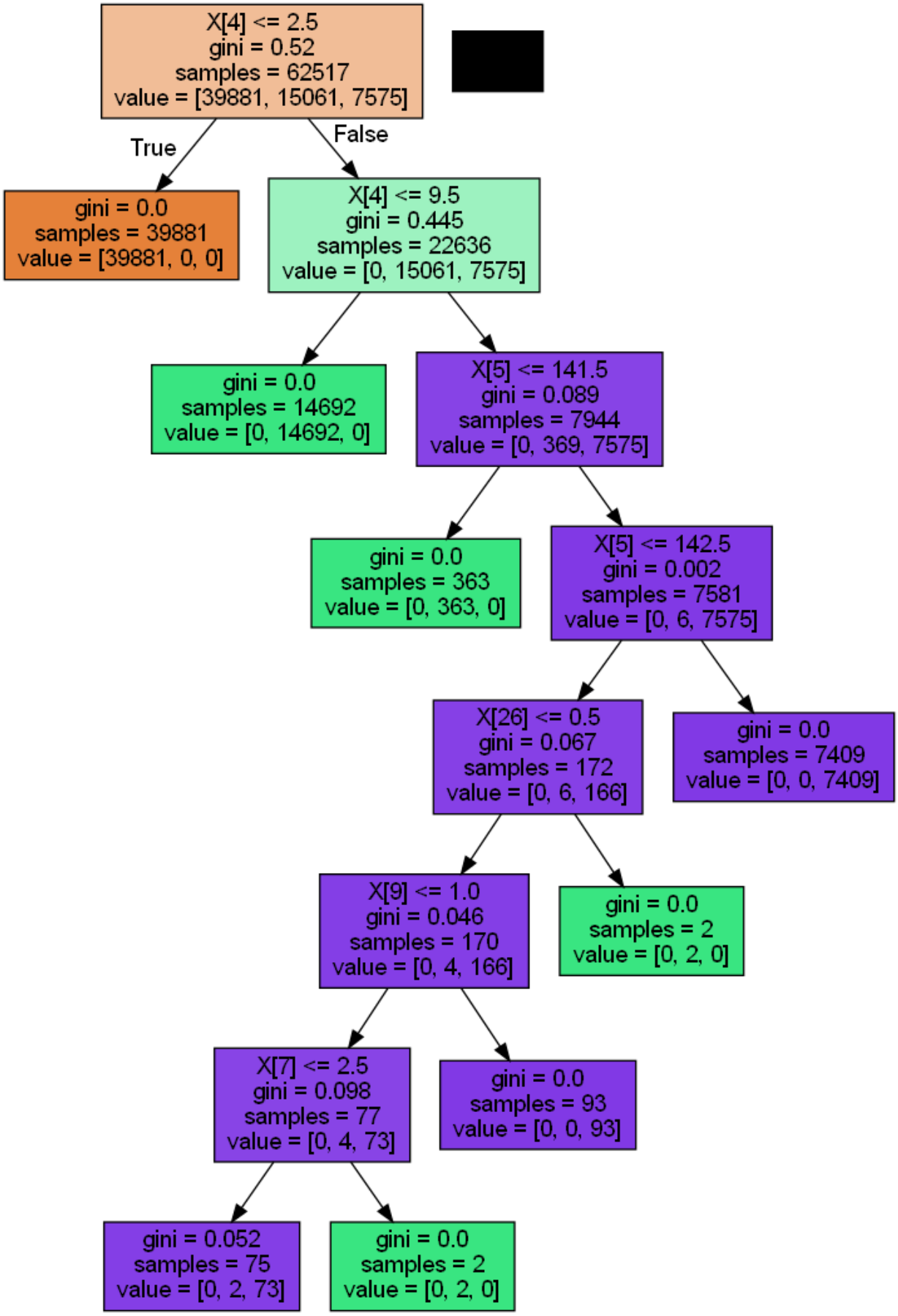
Out[156]:

```
(15, 7)
```

In [157…

```
dot_data = StringIO()

export_graphviz(GR.best_estimator_, out_file=dot_data, filled=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

# View the tree image
filename = 'Pepsicotest.png'
graph.write_png(filename)
Image(filename=filename)
### END SOLUTION
```

Out[157]:



In [158…

```
X = df_addedsugar2_ohe.drop('AddedSugarClass', axis = 1)
y = df_addedsugar2_ohe['AddedSugarClass']
dt = DecisionTreeClassifier(max_depth=7, max_leaf_nodes=15)
```

```
model = dt.fit(X,y)
```

In [159…

```
import graphviz
from graphviz import Source
dot_data = export_graphviz(model, out_file=None, feature_names=X.columns)
graph = graphviz.Source(dot_data)
graph.render("TestDecisionTree",view = True)
```

Out[159]: 'TestDecisionTree.pdf'

Model drift was observed. There were several changes between the training and test datasets. There were no beverage product type in the test data sets, Vietnam and Philippines were not included as well. The correlation results also looked different on the test datasets. The decision tree for training data predict low added sugar <2.5g (most likely beverage) from Ukraine while testing data from South Africa.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: