**Московский государственный технический университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»
Отчёт по лабораторной работе №1
# «Разведочный анализ данных. Исследование и визуализация данных»

| | |
|---|---|
| Выполнил: | Проверил: |
| Бондаренко Д.К., | Нардид А.Н., |
| группа ИУ5-61Б | каф. ИУ5 |
| Дата:16.02.23 | Дата: 28.02.23 |
| Подпись: | Подпись: |

2023
Москва

## Цель работы

Изучение различных методов визуализация данных.

## Выполнение работы

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import squarify
from pandas.plotting import parallel_coordinates
import math
import seaborn as sns
import scipy
import plotly
```

In [2]:
```
dataset = pd.read_csv('menu.csv')
```

In [3]:
```
# Первые 5 строк датасета
dataset.head()
```

Out[3]:

| | Category | Item | Serving Size | Calories | Calories from Fat | Total Fat | Total Fat (% Daily Value) | Saturated Fat | Saturated Fat (% Daily Value) | Trans Fat | ... | Carbohydrates | Carbohydrates (% Daily Value) | Dietary Fiber | Dietary Fiber (% Daily Value) | Sugars | Prot |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Breakfast | Egg McMuffin | 4.8 oz (136 g) | 300 | 120 | 13.0 | 20 | 5.0 | 25 | 0.0 | ... | 31 | 10 | 4 | 17 | 3 | |
| **1** | Breakfast | Egg White Delight | 4.8 oz (135 g) | 250 | 70 | 8.0 | 12 | 3.0 | 15 | 0.0 | ... | 30 | 10 | 4 | 17 | 3 | |
| **2** | Breakfast | Sausage McMuffin | 3.9 oz (111 g) | 370 | 200 | 23.0 | 35 | 8.0 | 42 | 0.0 | ... | 29 | 10 | 4 | 17 | 2 | |
| **3** | Breakfast | Sausage McMuffin with Egg | 5.7 oz (161 g) | 450 | 250 | 28.0 | 43 | 10.0 | 52 | 0.0 | ... | 30 | 10 | 4 | 17 | 2 | |
| **4** | Breakfast | Sausage McMuffin with Egg Whites | 5.7 oz (161 g) | 400 | 210 | 23.0 | 35 | 8.0 | 42 | 0.0 | ... | 30 | 10 | 4 | 17 | 2 | |

5 rows × 24 columns

In [4]:
```
# Размер датасета - 260 строк, 24 колонки
dataset.shape
```

Out[4]:
```
(260, 24)
```

In [5]:
```
total_count = dataset.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 260
```

In [6]:
```
# Список колонок
dataset.columns
```

Out[6]:
```
Index(['Category', 'Item', 'Serving Size', 'Calories', 'Calories from Fat',
       'Total Fat', 'Total Fat (% Daily Value)', 'Saturated Fat',
       'Saturated Fat (% Daily Value)', 'Trans Fat', 'Cholesterol',
       'Cholesterol (% Daily Value)', 'Sodium', 'Sodium (% Daily Value)',
       'Carbohydrates', 'Carbohydrates (% Daily Value)', 'Dietary Fiber',
       'Dietary Fiber (% Daily Value)', 'Sugars', 'Protein',
       'Vitamin A (% Daily Value)', 'Vitamin C (% Daily Value)',
       'Calcium (% Daily Value)', 'Iron (% Daily Value)'],
      dtype='object')
```

```
In [7]: # Список колонок с типами данных
        dataset.dtypes
```

```
Out[7]: Category                          object
        Item                              object
        Serving Size                      object
        Calories                           int64
        Calories from Fat                  int64
        Total Fat                        float64
        Total Fat (% Daily Value)          int64
        Saturated Fat                    float64
        Saturated Fat (% Daily Value)      int64
        Trans Fat                        float64
        Cholesterol                        int64
        Cholesterol (% Daily Value)        int64
        Sodium                             int64
        Sodium (% Daily Value)             int64
        Carbohydrates                      int64
        Carbohydrates (% Daily Value)      int64
        Dietary Fiber                      int64
        Dietary Fiber (% Daily Value)      int64
        Sugars                             int64
        Protein                            int64
        Vitamin A (% Daily Value)          int64
        Vitamin C (% Daily Value)          int64
        Calcium (% Daily Value)            int64
        Iron (% Daily Value)               int64
        dtype: object
```

In [8]:
```python
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in dataset.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = dataset[dataset[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
Category - 0
Item - 0
Serving Size - 0
Calories - 0
Calories from Fat - 0
Total Fat - 0
Total Fat (% Daily Value) - 0
Saturated Fat - 0
Saturated Fat (% Daily Value) - 0
Trans Fat - 0
Cholesterol - 0
Cholesterol (% Daily Value) - 0
Sodium - 0
Sodium (% Daily Value) - 0
Carbohydrates - 0
Carbohydrates (% Daily Value) - 0
Dietary Fiber - 0
Dietary Fiber (% Daily Value) - 0
Sugars - 0
Protein - 0
Vitamin A (% Daily Value) - 0
Vitamin C (% Daily Value) - 0
Calcium (% Daily Value) - 0
Iron (% Daily Value) - 0
```

In [9]:
```python
# Основные статистические характеристки набора данных
dataset.describe()
```

Out[9]:

|  | Calories | Calories from Fat | Total Fat | Total Fat (% Daily Value) | Saturated Fat | Saturated Fat (% Daily Value) | Trans Fat | Cholesterol | Cholesterol (% Daily Value) | Sodium | ... | Carbohydrates | Carboh ( |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | 260.000000 | ... | 260.000000 | 260 |
| **mean** | 368.269231 | 127.096154 | 14.165385 | 21.815385 | 6.007692 | 29.965385 | 0.203846 | 54.942308 | 18.392308 | 495.750000 | ... | 47.346154 | 15 |
| **std** | 240.269886 | 127.875914 | 14.205998 | 21.885199 | 5.321873 | 26.639209 | 0.429133 | 87.269257 | 29.091653 | 577.026323 | ... | 28.252232 | 9 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0 |
| **25%** | 210.000000 | 20.000000 | 2.375000 | 3.750000 | 1.000000 | 4.750000 | 0.000000 | 5.000000 | 2.000000 | 107.500000 | ... | 30.000000 | 10 |
| **50%** | 340.000000 | 100.000000 | 11.000000 | 17.000000 | 5.000000 | 24.000000 | 0.000000 | 35.000000 | 11.000000 | 190.000000 | ... | 44.000000 | 15 |
| **75%** | 500.000000 | 200.000000 | 22.250000 | 35.000000 | 10.000000 | 48.000000 | 0.000000 | 65.000000 | 21.250000 | 865.000000 | ... | 60.000000 | 20 |
| **max** | 1880.000000 | 1060.000000 | 118.000000 | 182.000000 | 20.000000 | 102.000000 | 2.500000 | 575.000000 | 192.000000 | 3600.000000 | ... | 141.000000 | 47 |

8 rows × 21 columns

```
In [10]: dataset.drop(['Total Fat (% Daily Value)','Saturated Fat','Saturated Fat (% Daily Value)',
                       'Cholesterol (% Daily Value)', 'Sodium', 'Sodium (% Daily Value)',
                       'Carbohydrates (% Daily Value)', 'Dietary Fiber', 'Dietary Fiber (% Daily Value)',
                      'Calcium (% Daily Value)'], axis=1, inplace=True)
```

```
In [11]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260 entries, 0 to 259
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Category                   260 non-null    object 
 1   Item                       260 non-null    object 
 2   Serving Size               260 non-null    object 
 3   Calories                   260 non-null    int64  
 4   Calories from Fat          260 non-null    int64  
 5   Total Fat                  260 non-null    float64
 6   Trans Fat                  260 non-null    float64
 7   Cholesterol                260 non-null    int64  
 8   Carbohydrates              260 non-null    int64  
 9   Sugars                     260 non-null    int64  
 10  Protein                    260 non-null    int64  
 11  Vitamin A (% Daily Value)  260 non-null    int64  
 12  Vitamin C (% Daily Value)  260 non-null    int64  
 13  Iron (% Daily Value)       260 non-null    int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 28.6+ KB
```

```
In [12]: dataset.duplicated().sum()

Out[12]: 0
```

```
In [13]: copied_ds = dataset.copy()

         for col in dataset.select_dtypes(include=['object']).columns:
             copied_ds[col] = copied_ds[col].astype('string')
         for col in dataset.select_dtypes(include=['float64']).columns:
             copied_ds[col] = copied_ds[col].astype('float16')
         for col in dataset.select_dtypes(include=['int64']).columns:
             copied_ds[col] = copied_ds[col].astype('int16')

         dataset = copied_ds
```

```
In [14]: dataset.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260 entries, 0 to 259
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Category                   260 non-null    string 
 1   Item                       260 non-null    string 
 2   Serving Size               260 non-null    string 
 3   Calories                   260 non-null    int16  
 4   Calories from Fat          260 non-null    int16  
 5   Total Fat                  260 non-null    float16
 6   Trans Fat                  260 non-null    float16
 7   Cholesterol                260 non-null    int16  
 8   Carbohydrates              260 non-null    int16  
 9   Sugars                     260 non-null    int16  
 10  Protein                    260 non-null    int16  
 11  Vitamin A (% Daily Value)  260 non-null    int16  
 12  Vitamin C (% Daily Value)  260 non-null    int16  
 13  Iron (% Daily Value)       260 non-null    int16  
dtypes: float16(2), int16(9), string(3)
memory usage: 63.5 KB
```

```
In [15]: vitA=pd.DataFrame(dataset.groupby('Category')['Vitamin A (% Daily Value)'].mean())
         vitC=pd.DataFrame(dataset.groupby('Category')['Vitamin C (% Daily Value)'].mean())
```

In [16]:
```
vitA
```

Out[16]:

|  | Vitamin A (% Daily Value) |
|---|---|
| **Category** |  |
| **Beef & Pork** | 6.933333 |
| **Beverages** | 0.740741 |
| **Breakfast** | 6.928571 |
| **Chicken & Fish** | 20.444444 |
| **Coffee & Tea** | 10.736842 |
| **Desserts** | 5.142857 |
| **Salads** | 146.666667 |
| **Smoothies & Shakes** | 18.750000 |
| **Snacks & Sides** | 4.846154 |

In [17]:
```
vitC
```

Out[17]:

|  | Vitamin C (% Daily Value) |
|---|---|
| **Category** |  |
| **Beef & Pork** | 7.333333 |
| **Beverages** | 23.481481 |
| **Breakfast** | 8.904762 |
| **Chicken & Fish** | 12.629630 |
| **Coffee & Tea** | 0.000000 |
| **Desserts** | 4.142857 |
| **Salads** | 28.333333 |
| **Smoothies & Shakes** | 6.964286 |
| **Snacks & Sides** | 28.153846 |

In [18]:
```python
vits=vitA.merge(vitC, left_on='Category', right_on='Category',
                suffixes=('_left', '_right'))
```

In [19]:
```python
vits
```

Out[19]:

| Category | Vitamin A (% Daily Value) | Vitamin C (% Daily Value) |
|---|---|---|
| **Beef & Pork** | 6.933333 | 7.333333 |
| **Beverages** | 0.740741 | 23.481481 |
| **Breakfast** | 6.928571 | 8.904762 |
| **Chicken & Fish** | 20.444444 | 12.629630 |
| **Coffee & Tea** | 10.736842 | 0.000000 |
| **Desserts** | 5.142857 | 4.142857 |
| **Salads** | 146.666667 | 28.333333 |
| **Smoothies & Shakes** | 18.750000 | 6.964286 |
| **Snacks & Sides** | 4.846154 | 28.153846 |

In [20]:
```python
d = {'Category': ['Beef & Pork', 'Beverages', 'Breakfast', 'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',
                  'Smoothies & Shakes', 'Snacks & Sides'], 'Category': ['Beef & Pork', 'Beverages', 'Breakfast',
                                                                        'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',
                  'Smoothies & Shakes', 'Snacks & Sides']}
df = pd.DataFrame(data=d)
df
```

Out[20]:

| | Category |
|---|---|
| **0** | Beef & Pork |
| **1** | Beverages |
| **2** | Breakfast |
| **3** | Chicken & Fish |
| **4** | Coffee & Tea |
| **5** | Desserts |
| **6** | Salads |
| **7** | Smoothies & Shakes |
| **8** | Snacks & Sides |

In [21]:
```
vits = vits.merge(df, left_on='Category', right_on='Category',
           suffixes=('_left', '_right'))
```

In [22]:
```
vits
```

Out[22]:

| | Category | Vitamin A (% Daily Value) | Vitamin C (% Daily Value) |
|---|---|---|---|
| **0** | Beef & Pork | 6.933333 | 7.333333 |
| **1** | Beverages | 0.740741 | 23.481481 |
| **2** | Breakfast | 6.928571 | 8.904762 |
| **3** | Chicken & Fish | 20.444444 | 12.629630 |
| **4** | Coffee & Tea | 10.736842 | 0.000000 |
| **5** | Desserts | 5.142857 | 4.142857 |
| **6** | Salads | 146.666667 | 28.333333 |
| **7** | Smoothies & Shakes | 18.750000 | 6.964286 |
| **8** | Snacks & Sides | 4.846154 | 28.153846 |

In [23]:
```
sug=pd.DataFrame(dataset.groupby('Category')['Sugars'].mean())
carb=pd.DataFrame(dataset.groupby('Category')['Carbohydrates'].mean())
```

In [24]:
```
merged=carb.merge(sug, left_on='Category', right_on='Category',
           suffixes=('_left', '_right'))
```

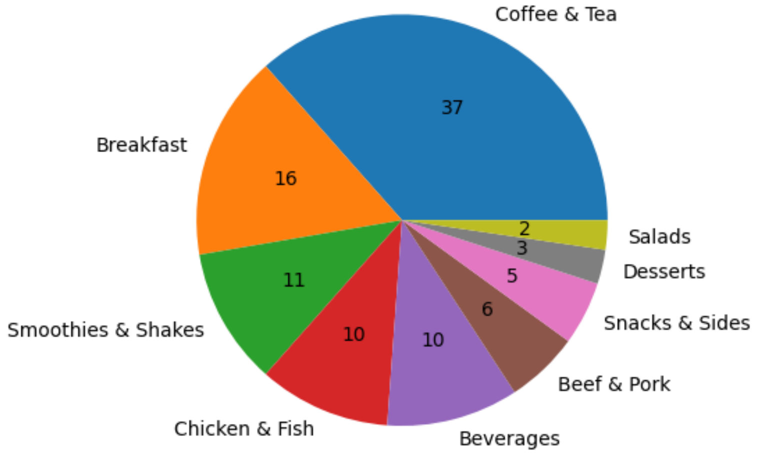
In [25]:
```
d = {'Category': ['Beef & Pork', 'Beverages', 'Breakfast', 'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',
                  'Smoothies & Shakes', 'Snacks & Sides'], 'Category': ['Beef & Pork', 'Beverages', 'Breakfast',
                                                                         'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',

                  'Smoothies & Shakes', 'Snacks & Sides']}
df = pd.DataFrame(data=d)
```

In [26]:
```
merged= merged.merge(df, left_on='Category', right_on='Category',
           suffixes=('_left', '_right'))
merged
```

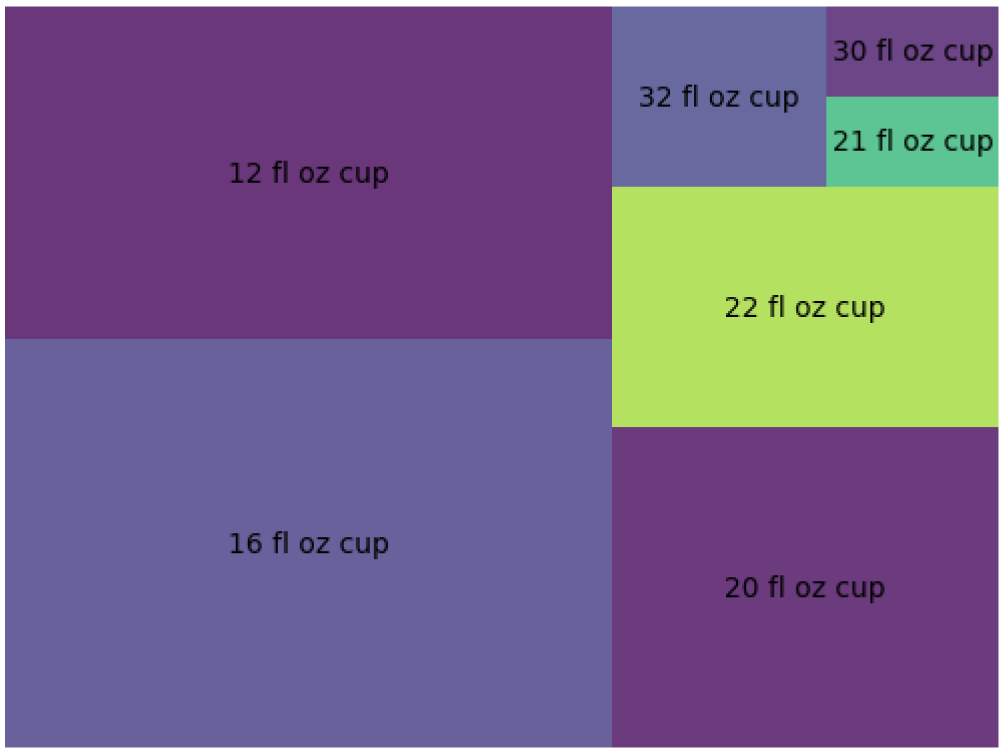
Out[26]:

| | Category | Carbohydrates | Sugars |
|---|---|---|---|
| **0** | Beef & Pork | 40.133333 | 8.800000 |
| **1** | Beverages | 28.814815 | 27.851852 |
| **2** | Breakfast | 49.761905 | 8.261905 |
| **3** | Chicken & Fish | 49.074074 | 7.333333 |
| **4** | Coffee & Tea | 44.526316 | 39.610526 |
| **5** | Desserts | 34.857143 | 26.142857 |
| **6** | Salads | 21.666667 | 6.833333 |
| **7** | Smoothies & Shakes | 90.428571 | 77.892857 |
| **8** | Snacks & Sides | 29.153846 | 4.076923 |

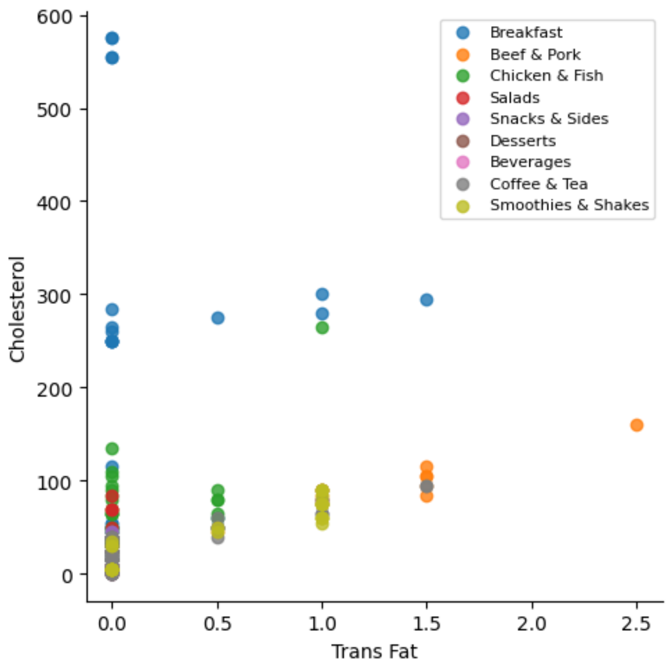
In [27]:
```
types = dataset['Category'].value_counts()
types = pd.DataFrame({'Category':types.index, 'Count':types.values})
plt.pie(types['Count'], labels=types['Category'], autopct = lambda p: format(p,'.0f') if p > 1 else None)
plt.show()
```



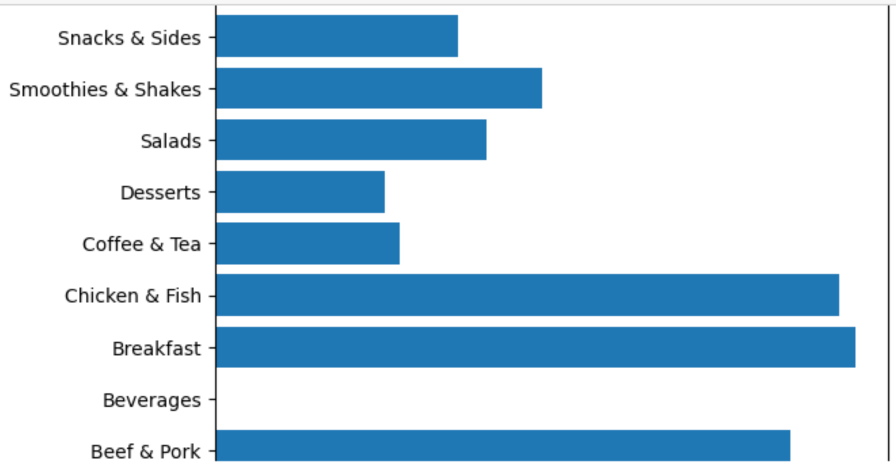
In [28]:
```
data = dataset.loc[dataset['Category'] == 'Coffee & Tea']
data = data['Serving Size'].value_counts()
squarify.plot(sizes=data.values, label=data.index, alpha=.8 )
plt.axis('off')
plt.show()
```



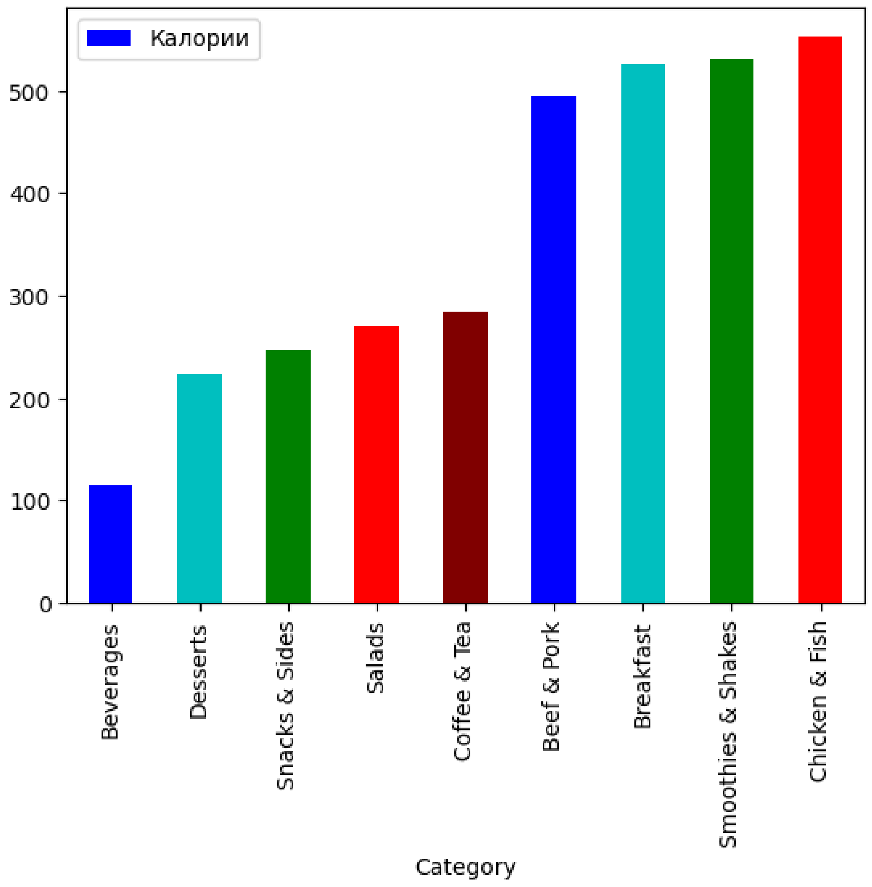
```
In [29]: sns.lmplot( x='Trans Fat', y='Cholesterol', data=dataset, fit_reg=False, hue='Category', legend=False)

         # Move the legend to an empty part of the plot
         plt.legend(loc='upper right', fontsize=8)

         plt.show()
```



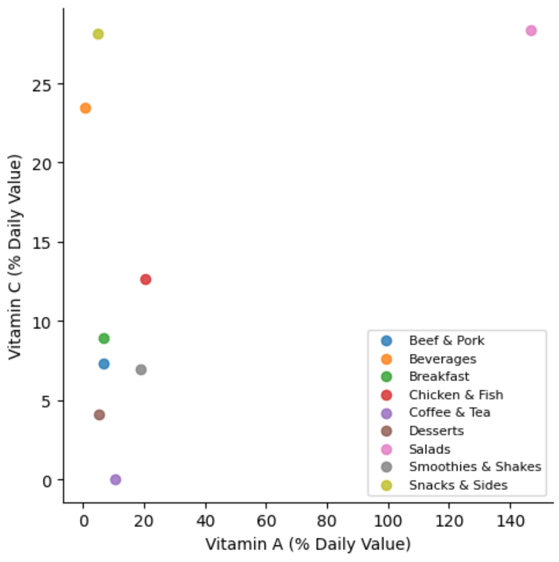
```
In [30]: height = dataset.groupby('Category')['Total Fat'].mean()
         bars = ('Beef & Pork', 'Beverages', 'Breakfast', 'Chicken & Fish', 'Coffee & Tea', 'Desserts',
                 'Salads', 'Smoothies & Shakes', 'Snacks & Sides')

         y_pos = np.arange(len(bars))

         # Create bars
         plt.barh(y_pos, height)

         # Create names on the x-axis
         plt.yticks(y_pos, bars)

         # Show graphic
         plt.show()
```



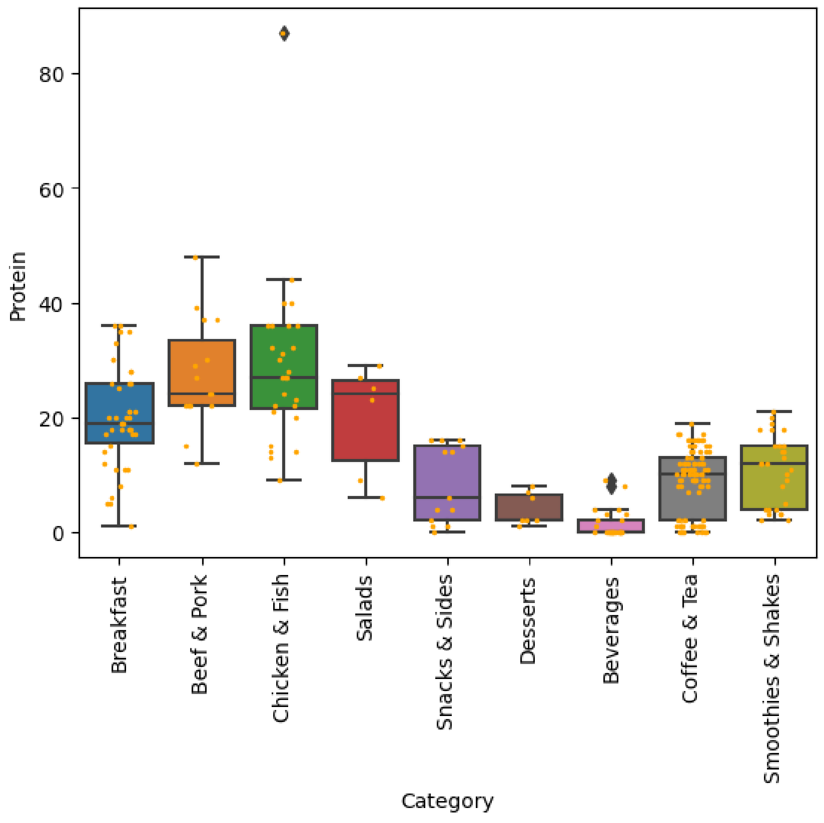
```
In [31]: cal = pd.DataFrame(dataset.groupby('Category')['Calories'].mean()).sort_values('Calories')
         plot = cal.plot.bar(y='Calories', color=['b','c','g','r','maroon'], label='Калории')
```



```
In [32]: sns.lmplot(x='Vitamin A (% Daily Value)', y='Vitamin C (% Daily Value)', data=vits, fit_reg=False, hue='Category', legend=False)

         # Move the legend to an empty part of the plot
         plt.legend(loc='lower right', fontsize=8)

         plt.show()
```



```
In [34]: sns.boxplot(x='Category', y='Protein', data=dataset)
         ax = sns.stripplot(x='Category', y='Protein', data=dataset, color="orange", jitter=0.2, size=2.5)
         plt.xticks(rotation=90)
         plt.show()
```



```
In [35]: sorted_df = dataset.sort_values('Protein', ascending=False)
         sorted_df
```

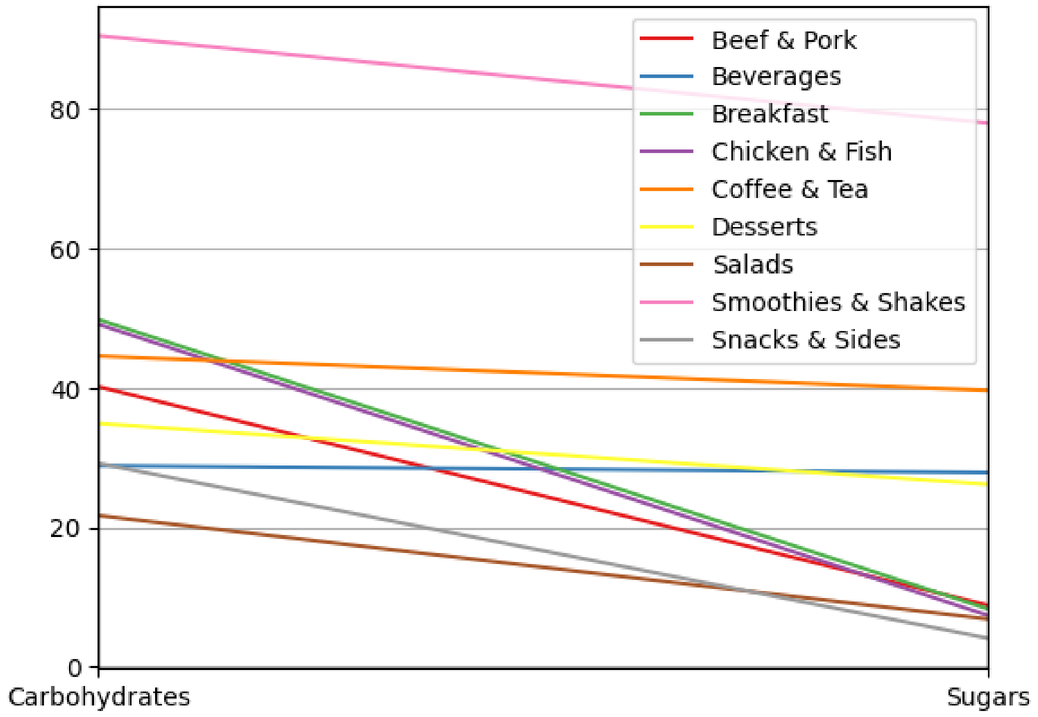
Out[35]:

| | Category | Item | Serving Size | Calories | Calories from Fat | Total Fat | Trans Fat | Cholesterol | Carbohydrates | Sugars | Protein | Vitamin A (% Daily Value) | Vitamin C (% Daily Value) | Iron (% Daily Value) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **82** | Chicken & Fish | Chicken McNuggets (40 piece) | 22.8 oz (646 g) | 1880 | 1060 | 118.0 | 1.0 | 265 | 118 | 1 | 87 | 0 | 15 | 25 |
| **47** | Beef & Pork | Double Quarter Pounder with Cheese | 10 oz (283 g) | 750 | 380 | 43.0 | 2.5 | 160 | 42 | 10 | 48 | 10 | 2 | 35 |
| **81** | Chicken & Fish | Chicken McNuggets (20 piece) | 11.4 oz (323 g) | 940 | 530 | 59.0 | 0.0 | 135 | 59 | 0 | 44 | 0 | 8 | 10 |
| **60** | Chicken & Fish | Premium Grilled Chicken Club Sandwich | 8.3 oz (235 g) | 510 | 180 | 20.0 | 0.0 | 105 | 44 | 9 | 40 | 8 | 20 | 20 |
| **64** | Chicken & Fish | Bacon Clubhouse Grilled Chicken Sandwich | 9.5 oz (270 g) | 590 | 230 | 25.0 | 0.0 | 110 | 51 | 14 | 40 | 8 | 30 | 15 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **110** | Beverages | Coca-Cola Classic (Small) | 16 fl oz cup | 140 | 0 | 0.0 | 0.0 | 0 | 39 | 39 | 0 | 0 | 0 | 0 |
| **136** | Beverages | Dasani Water Bottle | 16.9 fl oz | 0 | 0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **127** | Beverages | Sprite (Medium) | 21 fl oz cup | 200 | 0 | 0.0 | 0.0 | 0 | 54 | 54 | 0 | 0 | 0 | 0 |
| **138** | Coffee & Tea | Iced Tea (Medium) | 21 fl oz cup | 0 | 0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **137** | Coffee & Tea | Iced Tea (Small) | 16 fl oz cup | 0 | 0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

260 rows × 14 columns

```
In [36]: #sns.pairplot(dataset)
         #plt.show()
```

In [37]:
```
parallel_coordinates(merged, 'Category', colormap=plt.get_cmap("Set1"))
#plt.xticks(rotation=90)
plt.show()
```

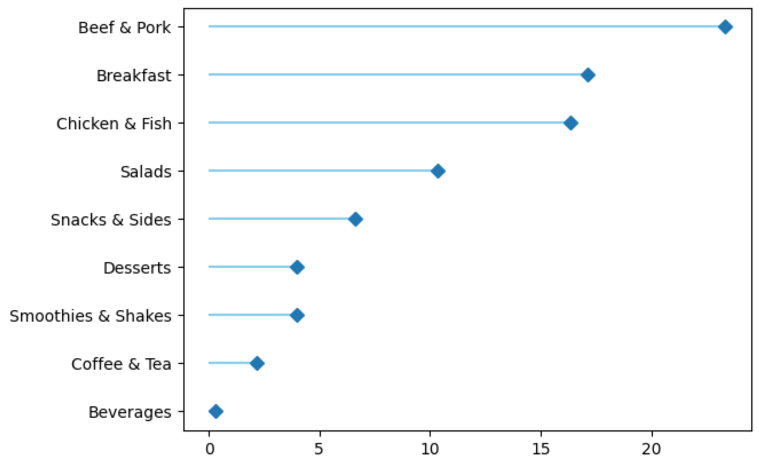


In [38]:
```
iron=pd.DataFrame(dataset.groupby('Category')['Iron (% Daily Value)'].mean())
```

In [39]:
```
d = {'Category': ['Beef & Pork', 'Beverages', 'Breakfast', 'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',
                  'Smoothies & Shakes', 'Snacks & Sides'], 'Category': ['Beef & Pork', 'Beverages', 'Breakfast',
                                                                         'Chicken & Fish', 'Coffee & Tea', 'Desserts', 'Salads',
                  'Smoothies & Shakes', 'Snacks & Sides']}
df = pd.DataFrame(data=d)
```

In [40]:
```
iron = iron.merge(df, left_on='Category', right_on='Category',
          suffixes=('_left', '_right'))
```

In [41]:
```
ordered_df = iron.sort_values(by='Iron (% Daily Value)')
my_range=range(1,len(iron.index)+1)

plt.hlines(y=my_range, xmin=0, xmax=ordered_df['Iron (% Daily Value)'], color='skyblue')
plt.plot(ordered_df['Iron (% Daily Value)'], my_range, "D")
plt.yticks(my_range, ordered_df['Category'])
plt.show()
```

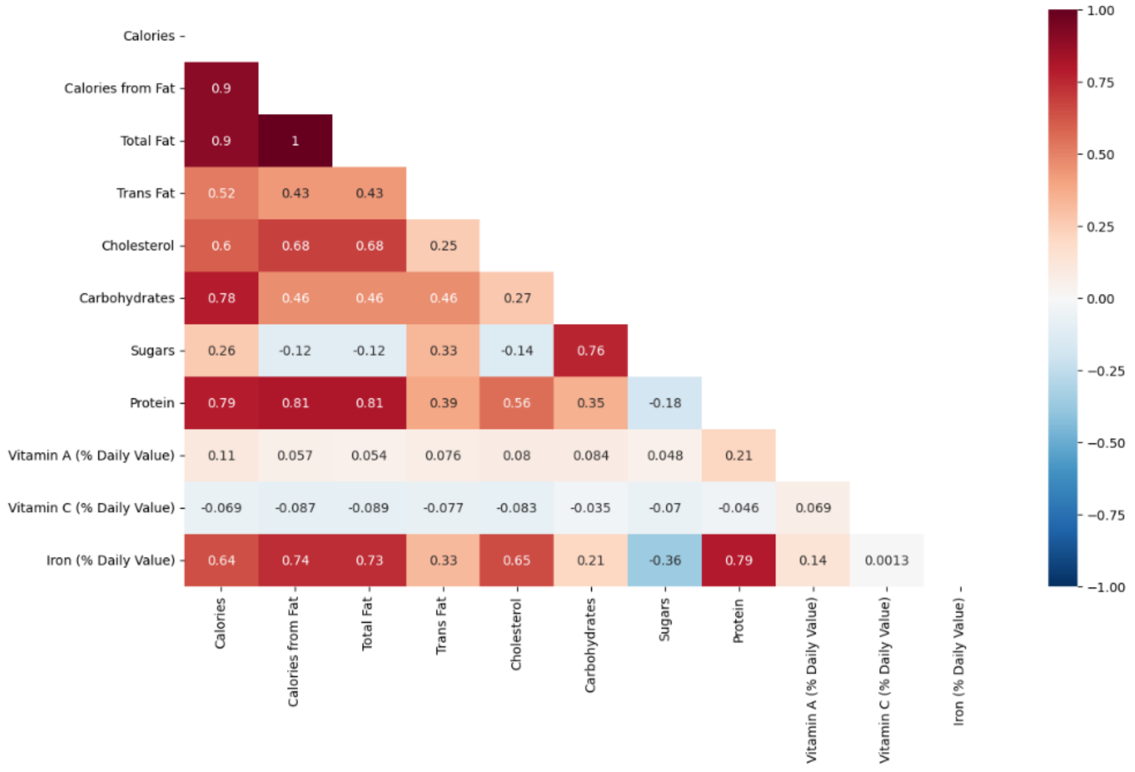


```
In [42]: iron
```

Out[42]:

| | Category | Iron (% Daily Value) |
|---|---|---|
| 0 | Beef & Pork | 23.333333 |
| 1 | Beverages | 0.296296 |
| 2 | Breakfast | 17.142857 |
| 3 | Chicken & Fish | 16.370370 |
| 4 | Coffee & Tea | 2.147368 |
| 5 | Desserts | 4.000000 |
| 6 | Salads | 10.333333 |
| 7 | Smoothies & Shakes | 3.964286 |
| 8 | Snacks & Sides | 6.615385 |

```
In [33]: plt.figure(figsize = (14,8))

         m =np.triu(np.ones_like(dataset.corr(), dtype=bool))

         sns.heatmap(dataset.corr(), mask = m, annot = True, vmin= -1.0, vmax= 1.0, center = 0, cmap = 'RdBu_r');
```


Calories
Calories from Fat 0.9
Total Fat 0.9 1
Trans Fat 0.52 0.43 0.43
Cholesterol 0.6 0.68 0.68 0.25
Carbohydrates 0.78 0.46 0.46 0.46 0.27
Sugars 0.26 -0.12 -0.12 0.33 -0.14 0.76
Protein 0.79 0.81 0.81 0.39 0.56 0.35 -0.18
Vitamin A (% Daily Value) 0.11 0.057 0.054 0.076 0.08 0.084 0.048 0.21
Vitamin C (% Daily Value) -0.069 -0.087 -0.089 -0.077 -0.083 -0.035 -0.07 -0.046 0.069
Iron (% Daily Value) 0.64 0.74 0.73 0.33 0.65 0.21 -0.36 0.79 0.14 0.0013
Calories
Calories from Fat
Total Fat
Trans Fat
Cholesterol
Carbohydrates
Sugars
Protein
Vitamin A (% Daily Value)
Vitamin C (% Daily Value)
Iron (% Daily Value)