SI 618 Final Report (Part A)

Name: Omkar R Sunkersett ; Date: April 13, 2017

Motivation

My report tries to answer a research question about online recipes — “Are online recipes heart-healthy?” There are millions of recipes available on the internet nowadays but we do not know if they are good for people with heart-disease. My report focuses on such recipes and analyzes their nutritional profile to answer the following key questions:

1. Which are the top-10 healthiest recipes from a given dataset of recipes?
2. Which are the top-10 unhealthiest recipes from a given dataset of recipes?

Data Sources

I used two sources of data for my analysis so that it could be double-checked with data from two disparate sources for the purpose of verification.

Dataset 1: Epicurious Website

“Epicurious" is an online food resource providing millions of users with over 33,000 recipes, menus, ingredients, food preparation tips and expert advice. I am focusing only on the highest rated recipes related to lunch for my analysis.

URL: http://www.epicurious.com/search/?meal=lunch&sort=highestRated&content=recipe

Data Format: HTML (parsed using BeautifulSoup in Python)

Important Variables and Their Types:

Recipe: This variable indicates the name of the recipe scraped from “Epicurious”. It is a string.

Source: This variable indicates the source of the recipe. It is a string.

Calories: This variable indicates the total calories present in the recipe. It is a floating-point value and is measured in the unit of "cal".

Protein: This variable indicates the amount of protein present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Carbohydrates: This variable indicates the amount of carbohydrates present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Fat: This variable indicates the total fat present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Saturated Fat: This variable indicates the amount of saturated fat present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Cholesterol: This variable indicates the amount of cholesterol present in the recipe. It is a floating-point value and is measured in the unit of "milligrams".

Fiber: This variable indicates the amount of fiber present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Sodium: This variable indicates the amount of sodium present in the recipe. It is a floating-point value and is measured in the unit of "milligrams".

Number of recipes obtained from Epicurious = 1,827

These recipes have been published on the internet from over a decade ago.

Dataset 2: Spoonacular API

The Spoonacular API contains recipe related information for over 365,000 recipes and 86,000 food products, including price and nutrition related data for each recipe. I am extracting one recipe from Spoonacular per recipe of Epcurious based on the condition of the maximum commonality of ingredients between each pair of recipe from both the sources.

URL: https://market.mashape.com/spoonacular/recipe-food-nutrition

Data Format: JSON (using Python)

Important Variables and Their Types:

Recipe: This string variable indicates the name of the recipe obtained from "Spoonacular".

Source: This variable indicates the source of the recipe. It is a string.

Calories: This variable indicates the total calories present in the recipe. It is a floating-point value and is measured in the unit of "cal".

Protein: This variable indicates the amount of protein present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Carbohydrates: This variable indicates the amount of carbohydrates present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Fat: This variable indicates the total fat present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Saturated Fat: This variable indicates the amount of saturated fat present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Cholesterol: This variable indicates the amount of cholesterol present in the recipe. It is a floating-point value and is measured in the unit of "milligrams".

Fiber: This variable indicates the amount of fiber present in the recipe. It is a floating-point value and is measured in the unit of "grams".

Sodium: This variable indicates the amount of sodium present in the recipe. It is a floating-point value and is measured in the unit of "milligrams".

Number of recipes obtained from Spoonacular = 1,827

These recipes have been published on the internet from over a decade ago.

Data Manipulation Method (used for each of the two sources)

I scraped the data from the Epicurious website using BeautifulSoup in Python, extracting the name of the recipe, ingredients and nutritional information from the webpage. While extracting the data, I removed the special characters (mainly # and ,) from the recipe names and ignored those recipes that had no nutritional information (i.e. NaN values). There were around 2,400 recipes out of which 1,827 had their nutritional information available. The nutrients had their units associated with them and daily recommended values (%) that I had to split from the actual integer value (Protein 40 g (80% DV)). I stored these 1,827 recipes in a local cache called "epicurious-cache.txt" as a list of tuples of the form [(recipe-name-n, [(nutrient-1, value-1), …, (nutrient-n, value-n)], [ingredient-1, …, ingredient-n]),…]. I then used the Spoonacular API and passed these values into the API using HTTP GET requests with the help of the requests module in Python. The Spoonacular API took the following parameters as input —

X-Mashape-Key: This is the client token used for authentication with the web server.

Accept: This specifies the format in which to retrieve the data from the web server (i.e. application/json, etc).

fillIngredients: This is a Boolean parameter used to indicate that the list of ingredients will be provided by the user.

ingredients: This is a string of comma-separated values of ingredients given as the input.

limitLicense: This is a Boolean parameter used to indicate whether the user should be charged beyond the maximum usage limit of the API (number of requests) based on the plan of the user or the HTTP GET request be aborted.

number: This is an integer parameter used to indicate the number of recipe IDs to retrieve from the web server.

ranking: This is an integer parameter used to indicate the mechanism to be used to retrieve the results. An integer value of 1 indicates that the recipes should be selected on the basis of the maximum ingredient usage (i.e. the maximum number of common ingredients between the "ingredients" parameter and the actual ingredients present in the "Spoonacular" recipe) whereas an integer value of 2 indicates that the recipes should be selected on the basis of the minimum ingredient usage (i.e. the minimum number of common ingredients between the "ingredients" parameter and the actual ingredients present in the "Spoonacular" recipe).

I obtained the recipe ID of a "Spoonacular" recipe during each HTTP GET request as the feedback from the web server and passed this recipe ID as an input to another HTTP GET request to extract the nutritional information about the recipe. The additional parameter used during this HTTP GET request was "includeNutrition", which takes a Boolean value to indicate whether the user wants to retrieve the nutritional information about the recipe. Finally, I got a dictionary in JSON format as the return value from the web server for every recipe ID. I stored all these dictionaries in a local cache called "spoonacular-cache.txt" as a list of dictionaries. I also ensured that I ignored some recipes that had missing calorie values (i.e. 'NA' values) from the final dataset. These recipes were "Mini Chorizo Corn Dogs" and "Steak Rancheros". I also assigned 0.0 values to those nutrients that were missing and ignored recipes with greater than 2000 calories from my analysis. I also did not include "sugar (g)" values from "Spoonacular" because they were completely missing from the "Epicurious" data and I needed at least two datasets to corroborate my results. The final dataset was prepared by combining the "cleaned" data from both the sources in TSV (tab-separated value) format. The final dataset has 3,655 recipes in total.

The assumption that I made was that heart-healthy recipes are those recipes that have the least amount of saturated fat, cholesterol and sodium in total. The sugar content would have been useful to determine which recipes were healthy but one of my sources of data (i.e. "Epicurious") did not have its sugar content publicly available over the internet. Hence, I could not include the sugar content of the recipes in my analysis. Moreover, the units of the saturated fat content, cholesterol and sodium were not the same. For instance, saturated fat was measured in grams (g) but cholesterol and sodium were measured in milligrams (mg). Hence, I had to convert the cholesterol and sodium content into grams (g) while performing my analysis. As we know that 1 gram (g) = 1000 milligrams (mg), so I had to divide the cholesterol and sodium content by 1000.

Finally, I used the sorted() function in Python to sort the dataset in ascending order on the basis of its total saturated fat, cholesterol and sodium content. The formula for doing this is as follows —

epicurious = sorted(epicurious, key = lambda x: x['Saturated Fat'] + (x['Cholesterol'] / 1000) + (x['Sodium'] / 1000))

spoonacular = sorted(spoonacular, key = lambda x: x['Saturated Fat'] + (x['Cholesterol'] / 1000) + (x['Sodium'] / 1000))

where "epicurious" and "spoonacular" are lists of dictionaries from the respective sources of data in which each dictionary corresponds to one recipe from its local cache (i.e. either epicurious-cache.txt or spoonacular-cache.txt).

I prepared three final datasets for my analysis -

1. Epicurious recipes sorted in the ascending order on the basis of their total saturated fat, cholesterol and sodium content.

2. Spoonacular recipes sorted in the ascending order on the basis of their total saturated fat, cholesterol and sodium content.

3. Combined dataset containing all the rows from the above datasets mentioned in points 1 and 2. This dataset has been used exclusively for Part-B of the project.

Note: Datasets from points 1 and 2 have been used to create visualizations for Part-A of the project.

Workflow of Source Code:

The source code of my project has been divided into three files:

1. **build-cache-epicurious.py:** This file contains the code that creates the "epicurious-cache.txt" file, which contains the cached results of scraping the "Epicurious" website. This website has totally 139 pages of recipes (related to lunch) with 15-18 recipes on each page on an average. The code scrapes the HTML page obtained from the url "http://www.epicurious.com/search/?meal=lunch&sort=highestRated&content=recipe&page=N" where N is the page number ranging from 1 to 139 inclusive and stores the URL of each recipe in a list of recipe-URLs called "recipe_urls". The code then iterates through each URL and scrapes the HTML page that contains the nutritional data pertaining to that recipe. The name of the recipe, list of ingredients and list of nutrients are scraped and stored in separate lists ("recipe_name", "ingredients_name", "nutrients_name", "nutrients_value"). Further, the zip() function is applied to combine them into a tuple and this tuple is appended to a list called "recipe_info". Next, the recipes that contain no nutritional data are removed from this list in the data-cleanup process and a final list called "recipe_nutrition" is created. Finally, the code creates a text file called "epicurious-cache.txt" and writes each tuple from the list "recipe_nutrition" as a new line, caching the data into this file for future use.

2. **build-cache-spoonacular.py:** This file contains the code that creates the "spoonacular-cache.txt" file, which contains the cached results obtained from the "Spoonacular" API. The code reads the cached results obtained from "Epicurious" from the file "epicurious-cache.txt" into a tuple of the form (name, nutrients, ingredients). It then "transforms" them from the tuple into a dictionary containing the source, name and nutrients of the recipe. The code further passes the list of ingredients as a string of ingredients using the join() method to the "Spoonacular" API (using a HTTP GET request) to obtain one recipe that contains the maximum number of common ingredients with the "Epicurious" recipe. The recipe ID of this recipe is then passed back to the "Spoonacular" API to obtain the nutritional data pertaining to that recipe. Further, the code creates a list of dictionaries from Spoonacular in which each dictionary contains some recipe related nutritional information and finally writes this list of

dictionaries to a cache file called “spoonacular-cache.txt” in which each line corresponds to a dictionary stored as a string.

3. **build-all-datasets.py:** This file contains the code that creates the final datasets (dataset-epicurious.tsv, dataset-spoonacular.tsv, dataset-combined.tsv) in TSV (tab-separated value) format. It reads the “epicurious-cache.txt” cache file containing the cached data obtained from “Epicurious” into a list called “epicurious”, ensuring that all records containing “NaN” values are ignored. It replaces the commas (,) and hashes (#) from the recipe-names with spaces and assigns zero values to nutrients that have missing values. The code then sorts the recipes in the list in ascending order based on the total content of saturated fat, cholesterol and sodium present in each recipe. It then reads the “spoonacular-cache.txt” file containing the cached data obtained from “Spoonacular” but ignores two recipes ('Mini Chorizo Corn Dogs’ and 'Steak Rancheros’) that correspond to two recipes with “NaN” values obtained from “Epicurious”, storing the rest of the recipes in a list called “spoonacular”. It replaces the commas (,) and hashes (#) from the recipe-names with spaces and assigns zero values to nutrients that have missing values. The code then sorts the recipes in the list in ascending order based on the total content of saturated fat, cholesterol and sodium present in each recipe. Finally, it creates three output files (“dataset-epicurious.tsv”, “dataset-spoonacular.tsv” and “dataset-combined.tsv”), storing the recipe and nutrient-related data in them. Please note that these datasets have been intentionally segregated to perform some analysis in Tableau. The combined dataset of recipes has been used exclusively in Part-B of this project.

Analysis and Visualization

For my analysis and visualization, I used Tableau to create bar charts of the top-10 healthiest and top-10 unhealthiest heart-healthy recipes based on the following criteria:

1. **Calories (cal):** The healthiest recipes would have the least number of calories per serving; whereas, the unhealthiest recipes would have the most number of calories per serving.

2. **Carbohydrates (g):** The healthiest recipes would have the least amount of carbohydrates per serving; whereas, the unhealthiest recipes would have the most amount of carbohydrates per serving.

3. **Fat (g):** The healthiest recipes would have the least amount of total fat per serving; whereas, the unhealthiest recipes would have the most amount of total fat per serving.

4. **Protein (g):** The healthiest recipes would have the most amount of protein per serving; whereas, the unhealthiest recipes would have the least amount of protein per serving.

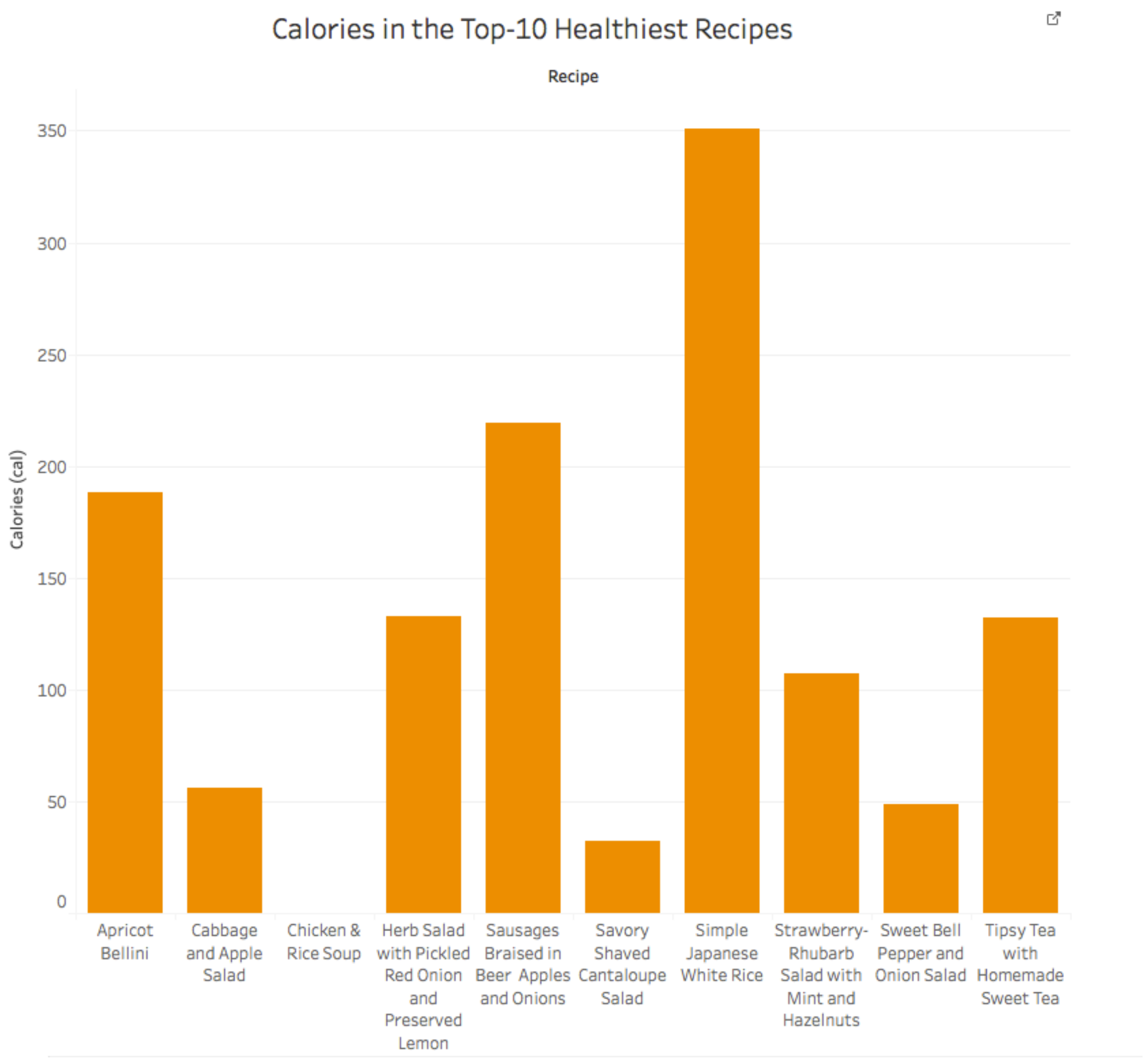
5. **Fiber (g):** The healthiest recipes would have the most amount of fiber per serving; whereas, the unhealthiest recipes would have the least amount of fiber per serving.

6. **Sodium (mg):** The healthiest recipes would have the least amount of sodium per serving; whereas, the unhealthiest recipes would have the most amount of sodium per serving.

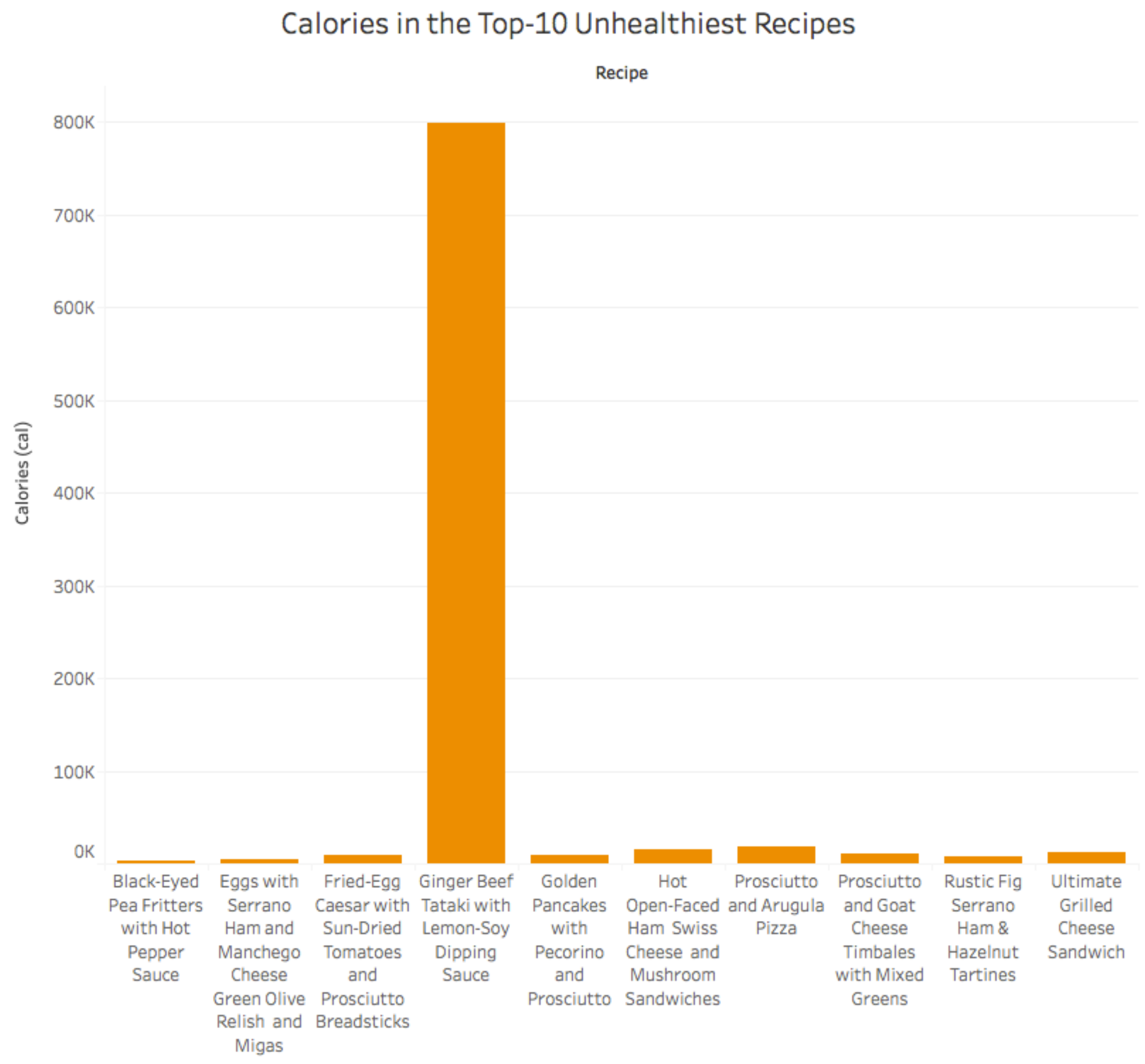
The above analysis was performed for both the sources of data (“Epicurious” and “Spoonacular”).

Bar Chart Based Analysis for “Epicurious" Recipes

A. Calories (cal)

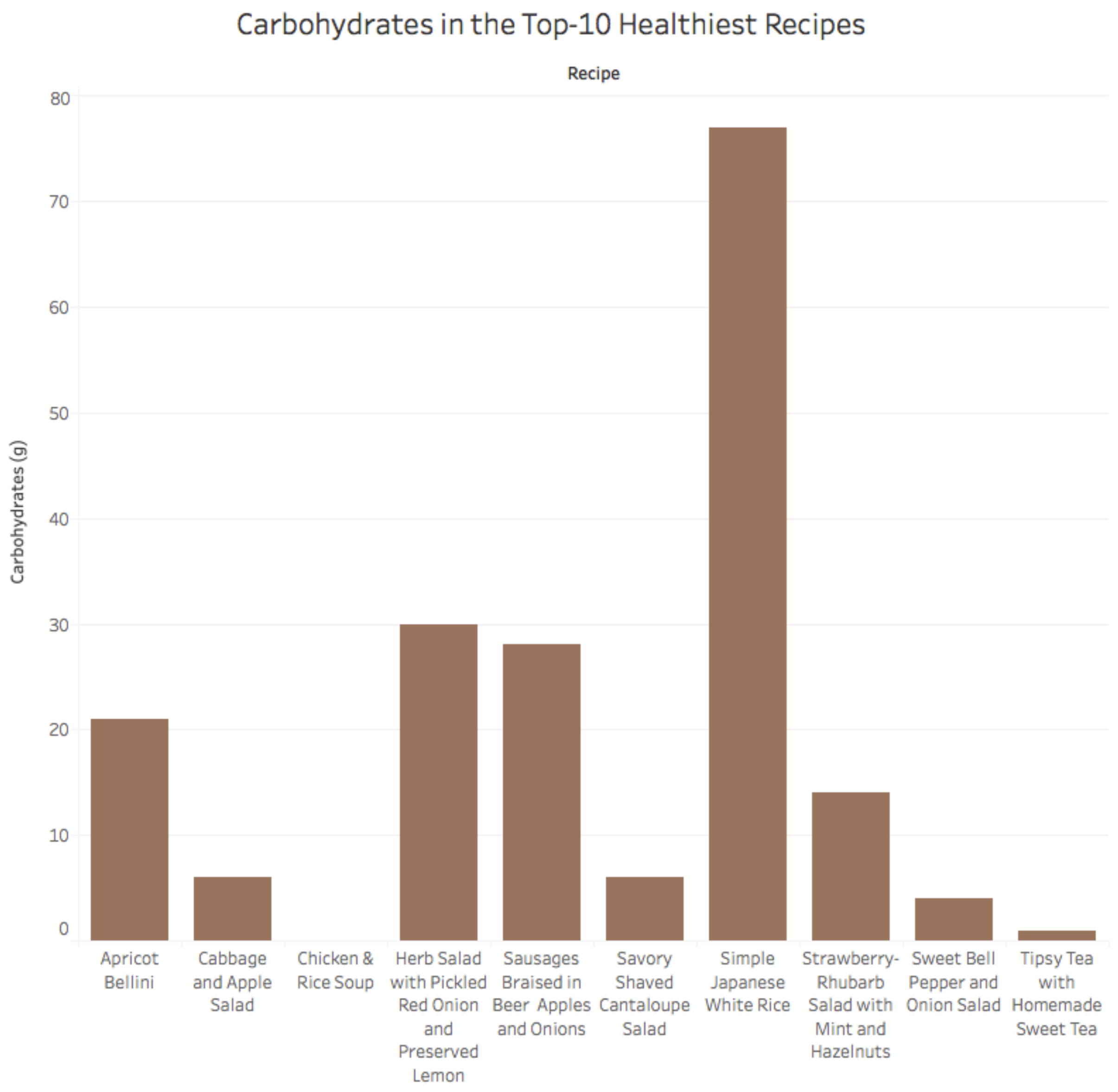


Finding: We find that the “Cabbage and Apple Salad”, “Chicken & Rice Soup”, “Savory Shaved Cantaloupe Salad” and “Sweet Bell Pepper and Onion Salad” have the least number of calories on an average, making them suitable for consumption because of their low calorific value.

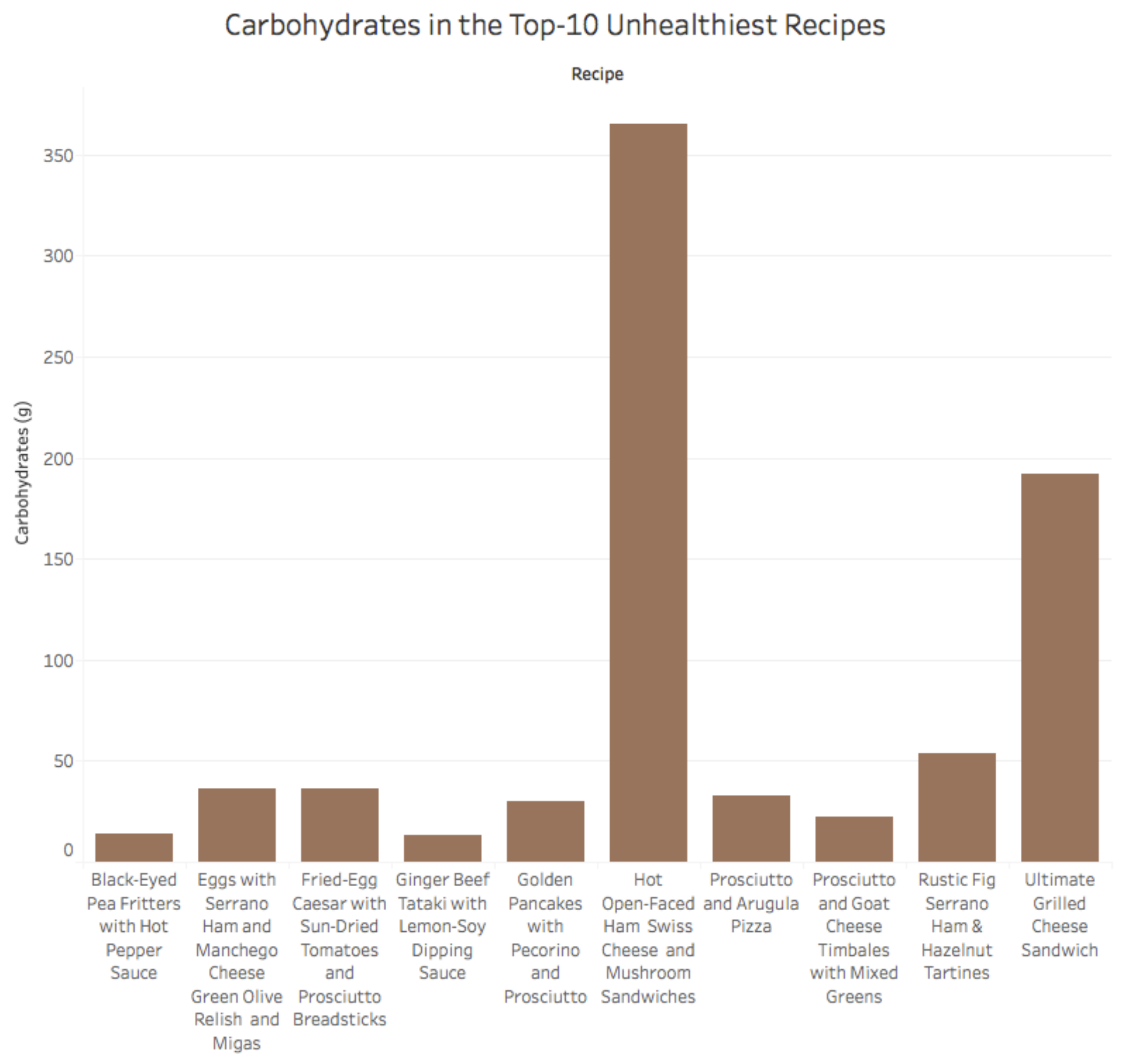


Finding: From the above chart, we notice that the "Ginger Beef Tataki with Lemon-Soy Dipping Sauce" is the most calorific in value and should be avoided since it is not healthy. The other two recipes that can be avoided are the "Hot Open-Faced Ham Swiss Cheese and Mushroom Sandwiches" and the "Prosciutto and Arugula Pizza" that are also high in the number of calories.

B. Carbohydrates (g):

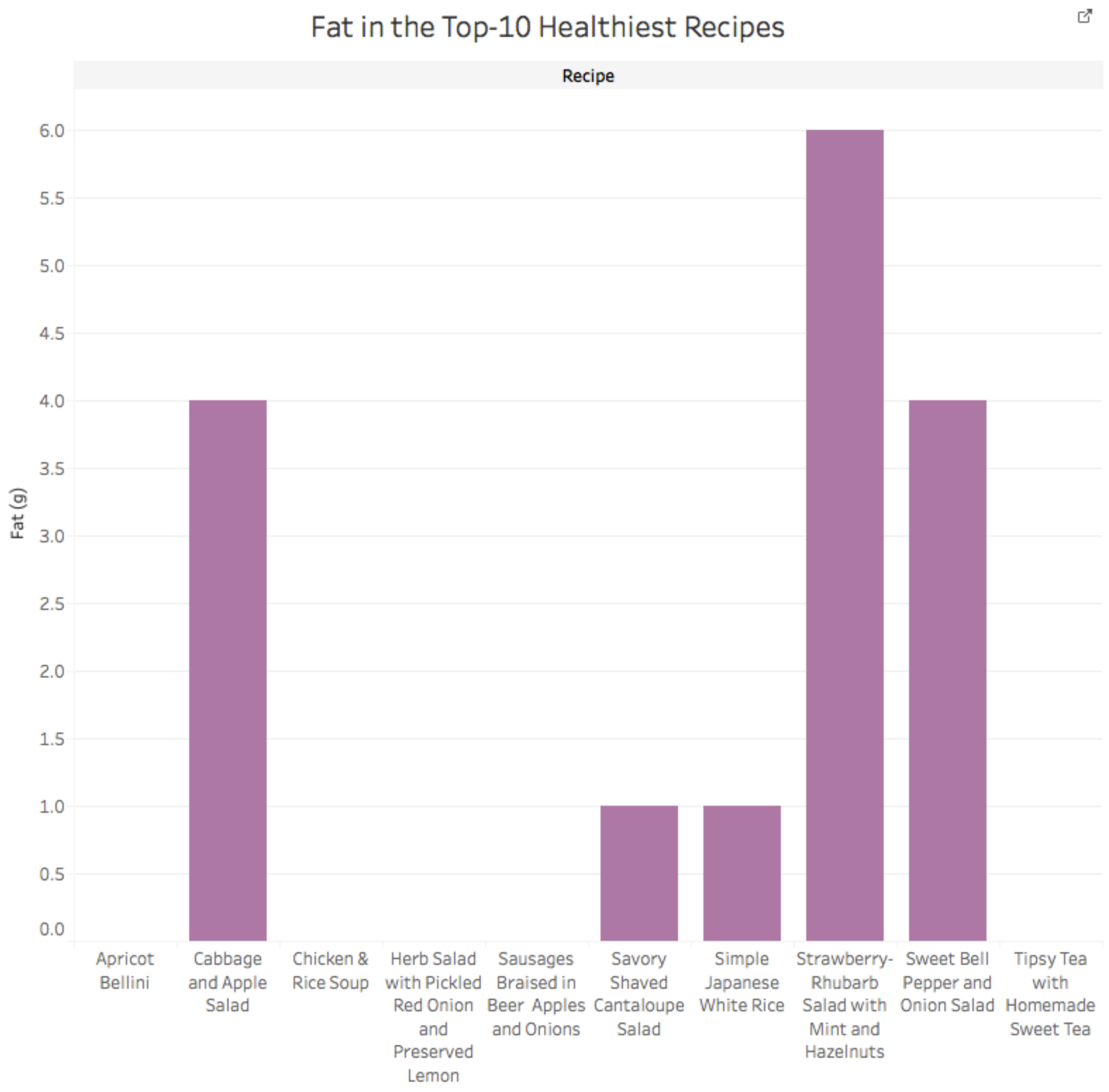


Finding: The above bar chart indicates that the "Cabbage and Apple Salad", "Chicken & Rice Soup", "Savory Shaved Cantaloupe Salad" and "Sweet Bell Pepper and Onion Salad" have the least amount of carbohydrates on an average, making them suitable for consumption because of their low carbohydrate content.

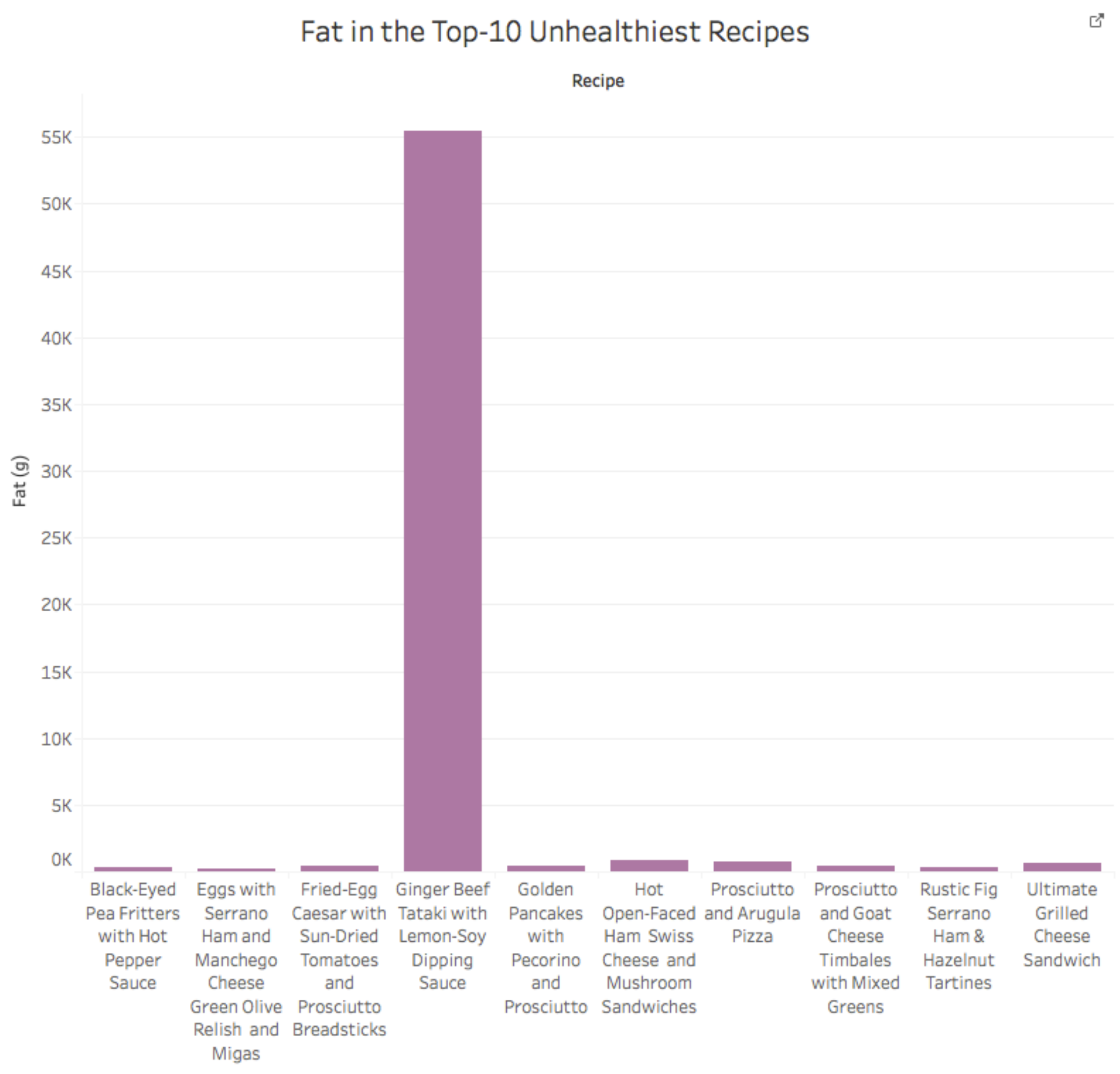


Finding: From the above bar chart, we can observe that the "Hot Open-Faced Ham Swiss Cheese and Mushroom Sandwiches", "Ultimate Grilled Cheese Sandwich" and "Rustic Fig Serrano Ham & Hazelnut Tartines" have the most amount of carbohydrates and must be avoided because of their high carbohydrate content.

C. Fat (g):

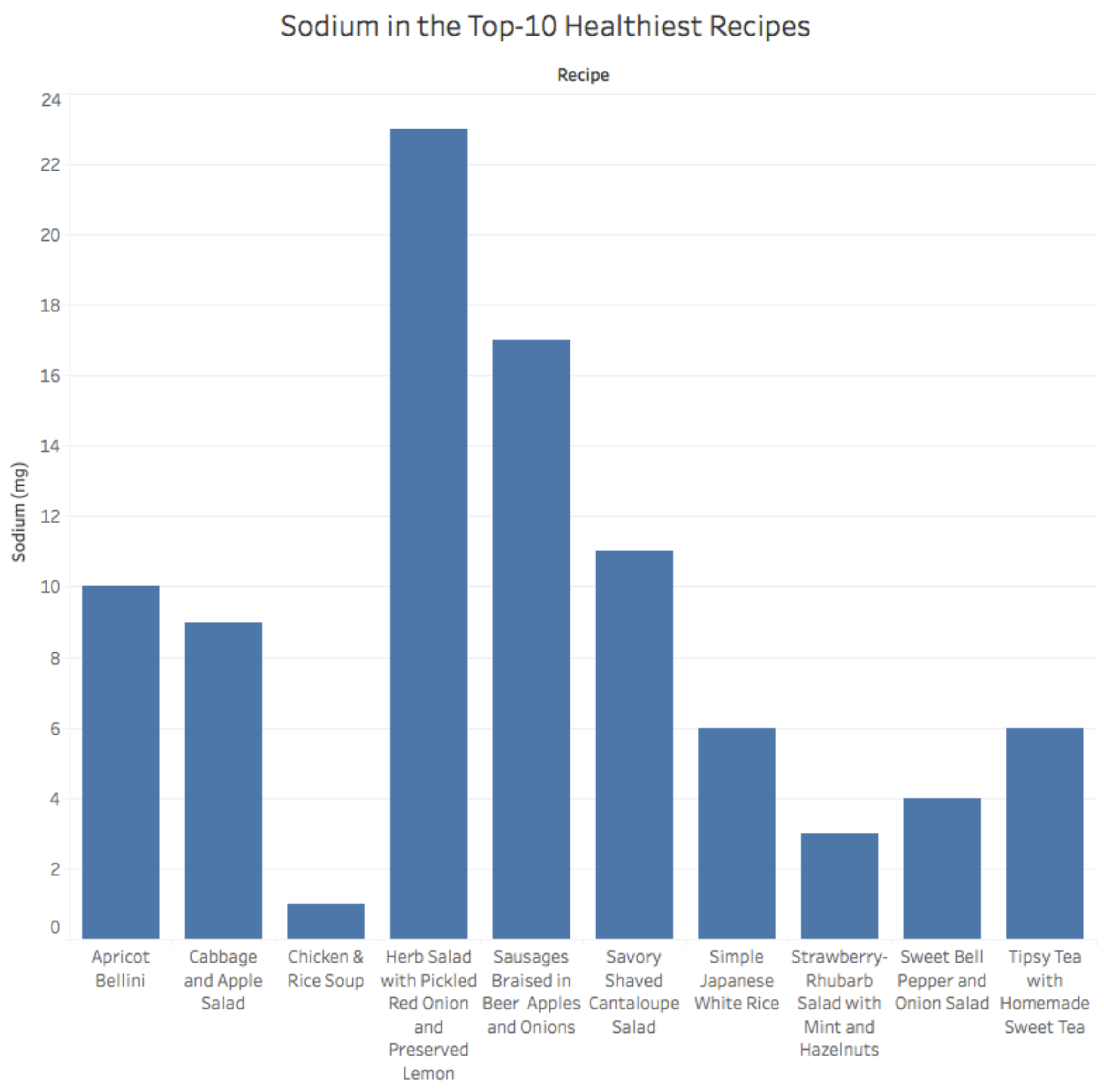


Finding: The above bar chart indicates that the “Apricot Bellini”, “Chicken & Rice Soup”, “Herb Salad with Pickled Red Onion and Preserved Lemon”, “Sausages Braised in Beer Apples and Onions” and “Tipsy Tea with Homemade Sweet Tea” have the lowest amount of overall fat content amongst the healthiest recipes, making them the most suitable for consumption.

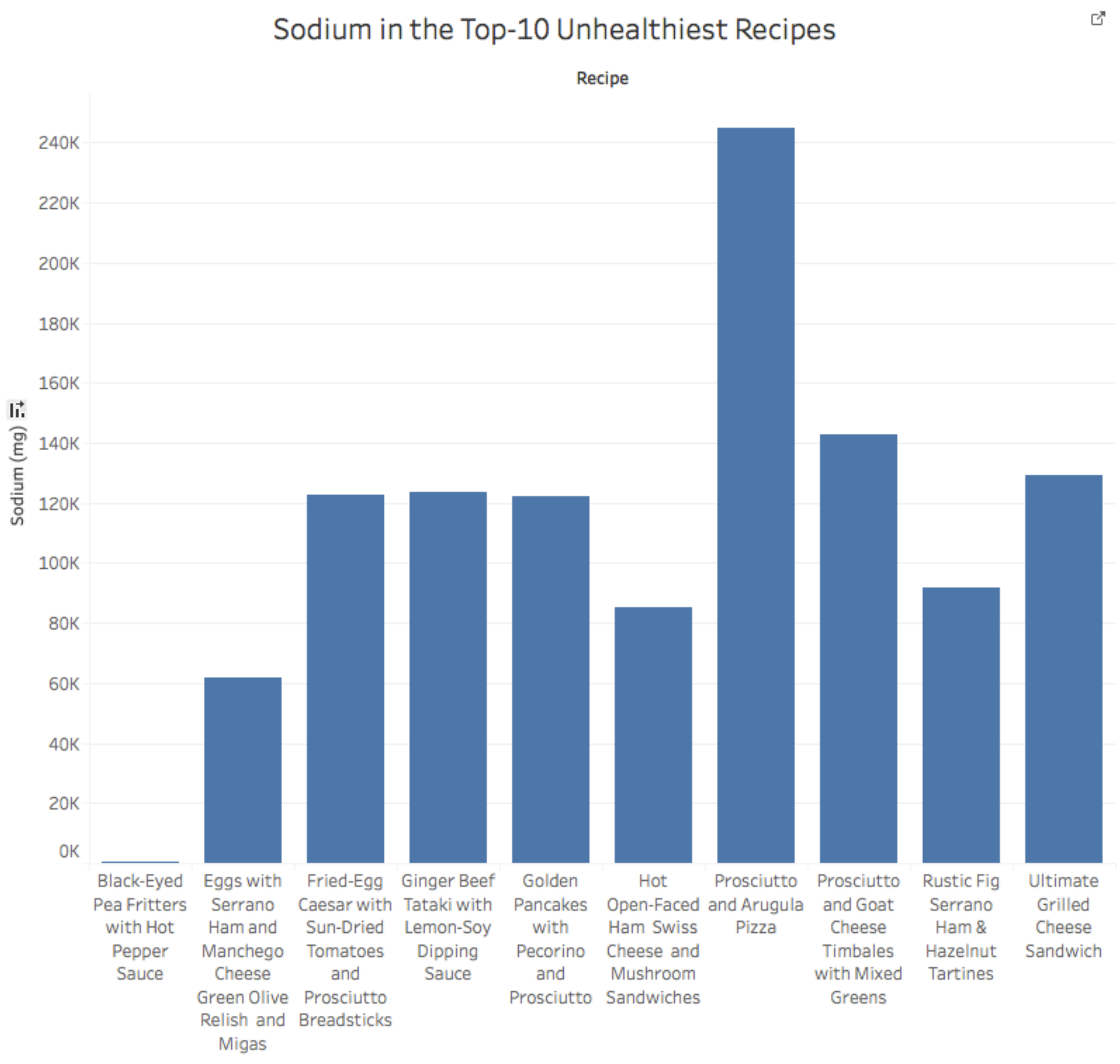


Finding: From the above chart, we notice that the "Ginger Beef Tataki with Lemon-Soy Dipping Sauce" has the most amount of fat and should be avoided since it is not healthy. The other two recipes that can be avoided are the "Hot Open-Faced Ham Swiss Cheese and Mushroom Sandwiches" and the "Prosciutto and Arugula Pizza" that have also got high amount of fat in them.

D. Sodium (mg):

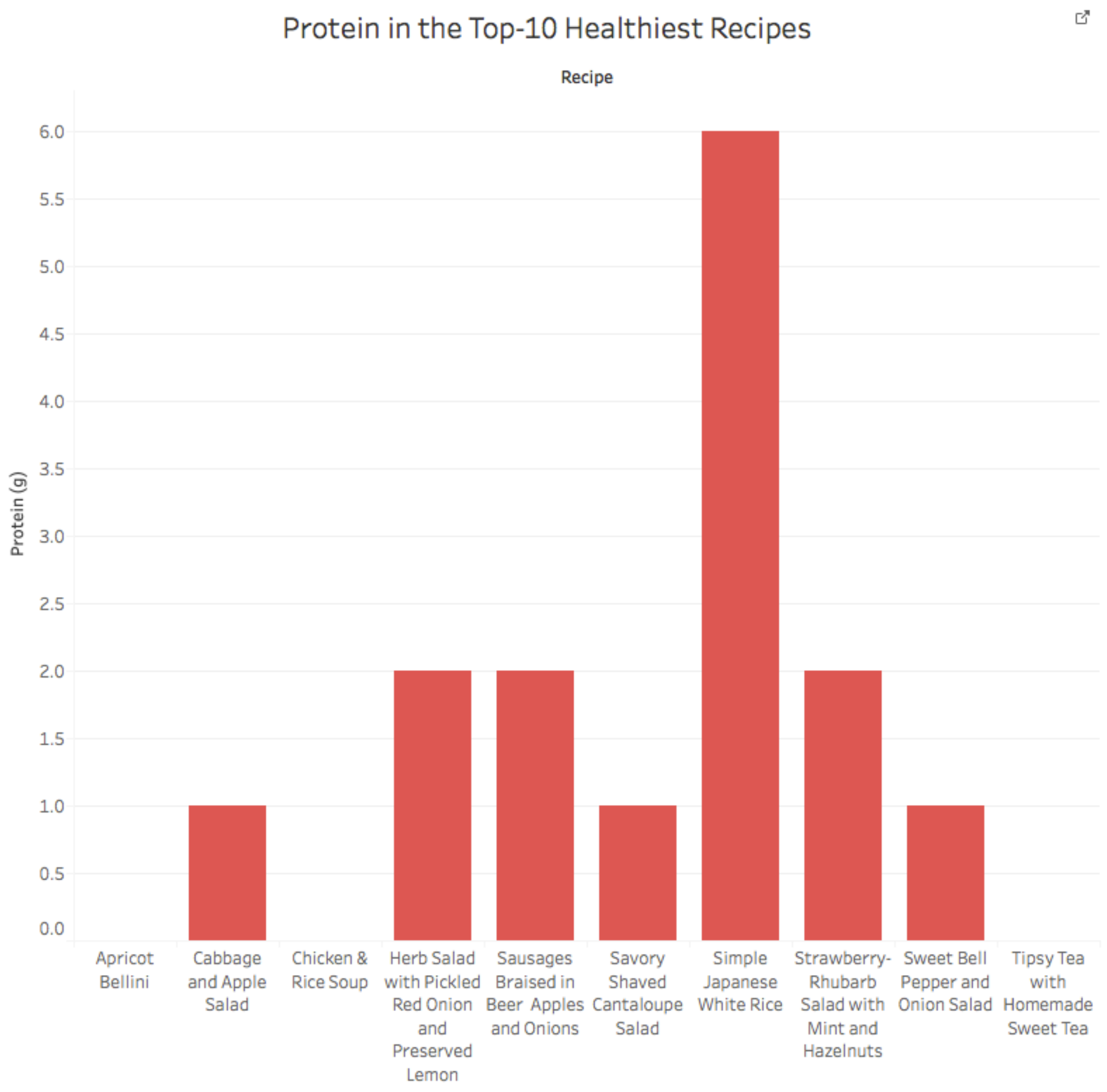


Finding: The above bar chart indicates that the “Chicken & Rice Soup”, “Strawberry Rhubarb Salad with Mint and Hazelnuts” and “Sweet Bell Pepper and Onion Salad” have the lowest amount of sodium in them, making them very heart-healthy for consumption.

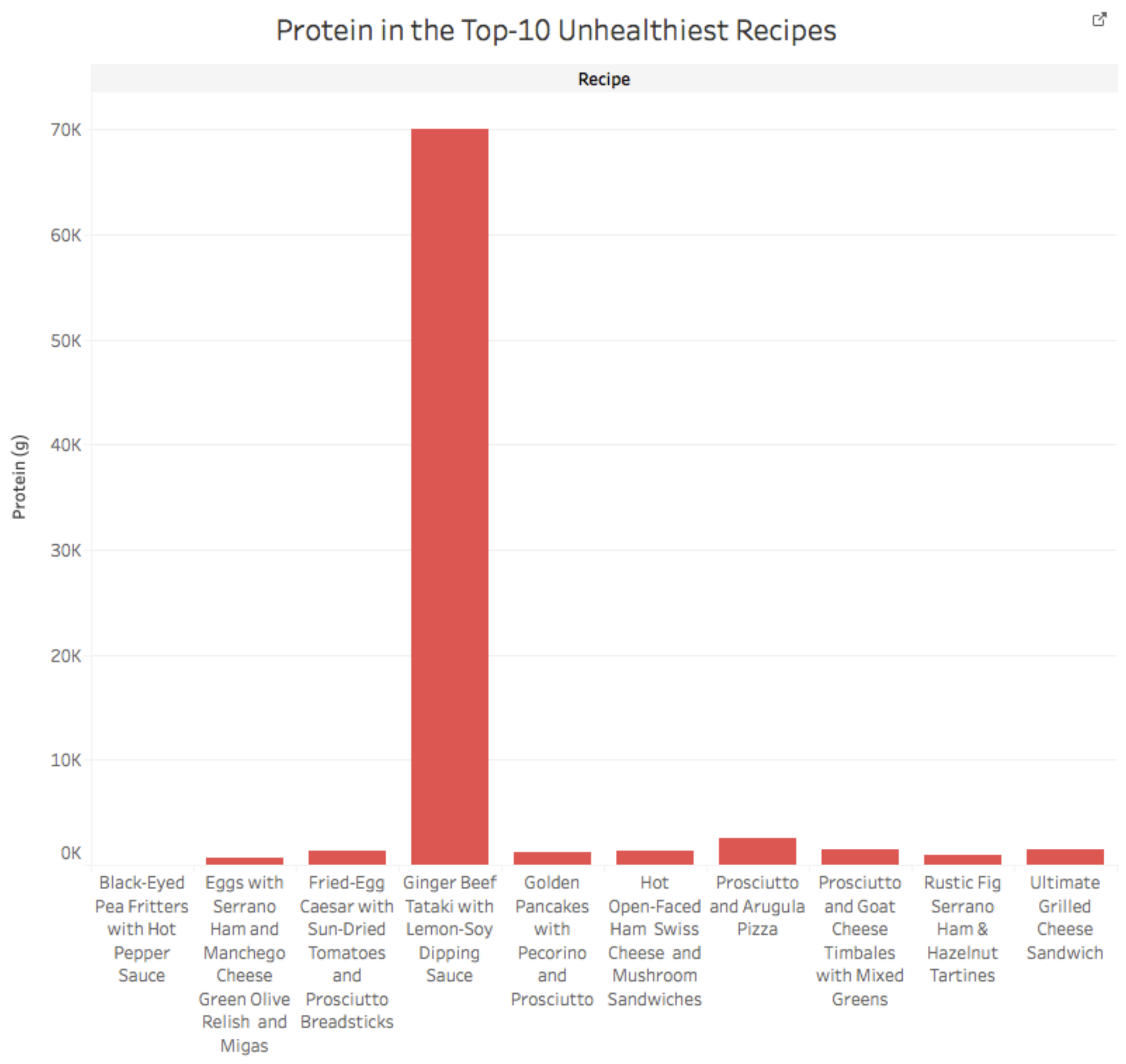


Finding: From the above bar chart, we can observe that the “Prosciutto and Arugula Pizza”, “Prosciutto and Goat Cheese Timbales with Mixed Greens” and “Ultimate Grilled Cheese Sandwich” have the highest amount of sodium amongst the unhealthiest recipes and should be avoided since they are not heart-healthy.

E. Protein (g):

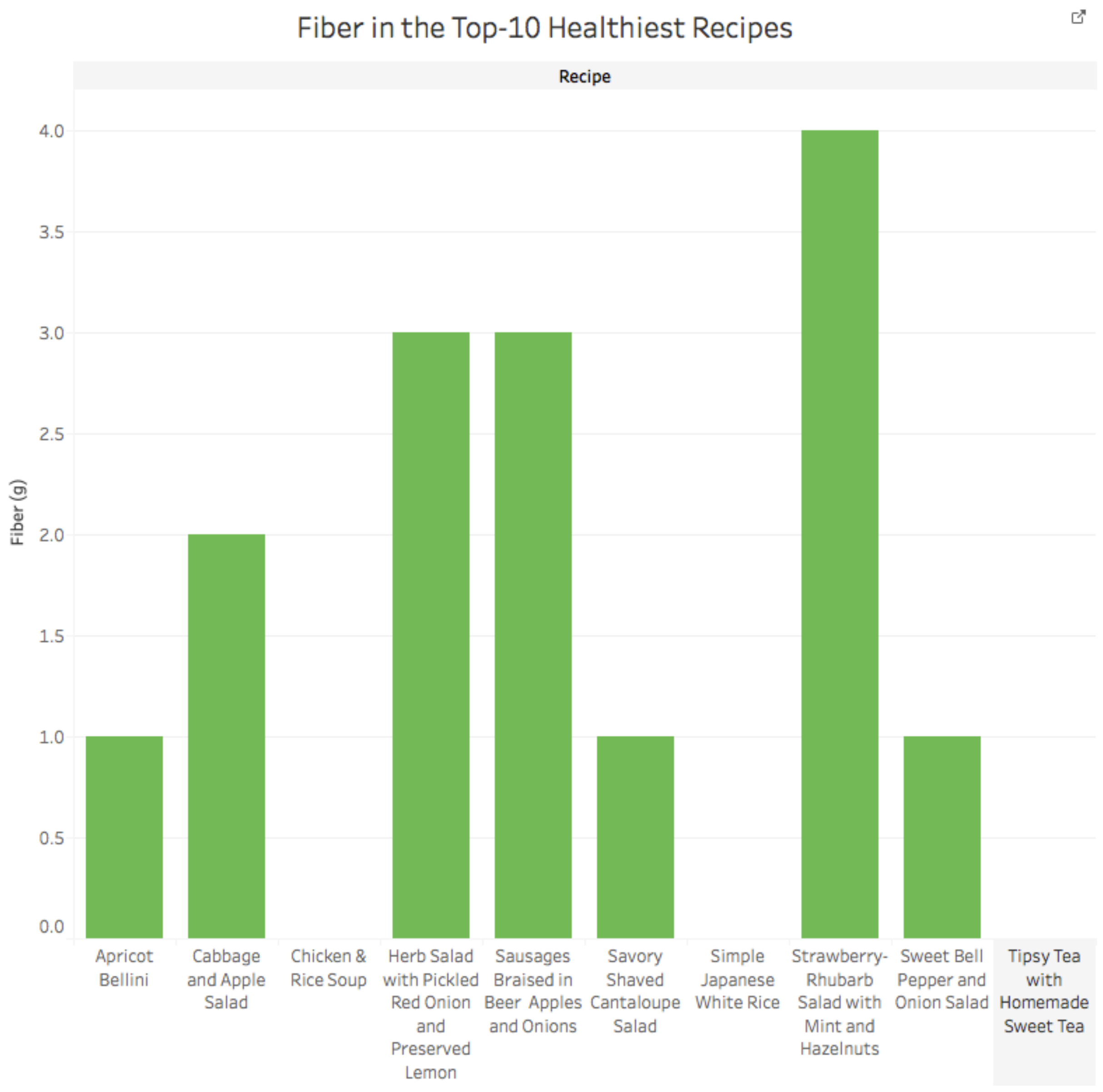


Finding: The above bar chart indicates that the "Simple Japanese White Rice", "Herb Salad with Pickled Red Onion and Preserved Lemon", "Sausages Braised in Beer Apples and Onions" and "Strawberry Rhubarb Salad with Mint and Hazelnuts" have the highest amount of protein amongst the healthiest heart-healthy recipes, making them a good choice for consumption.

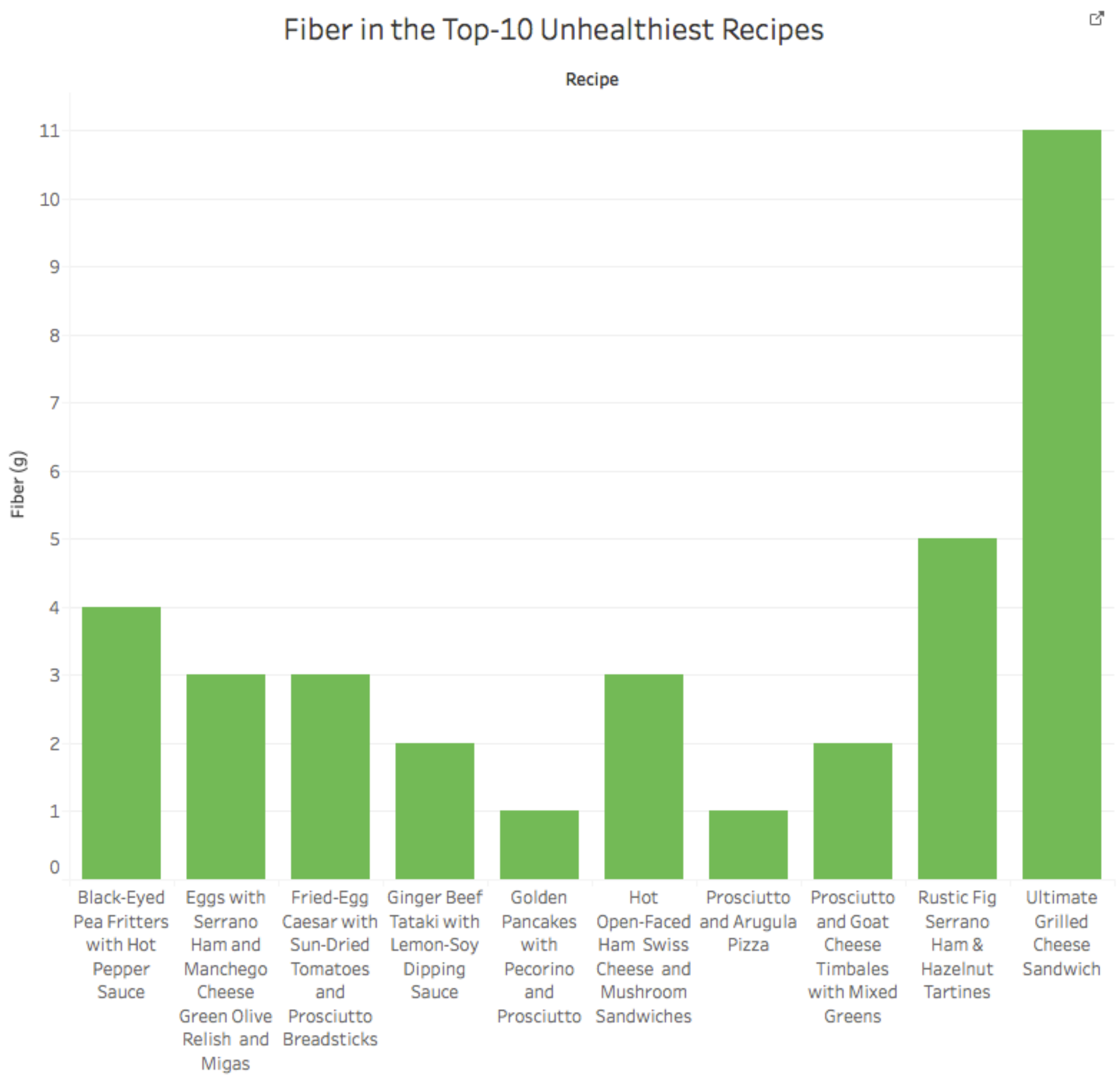


Finding: From the above bar chart, we can observe that the "Black-Eyed Pea Fritters with Hot Pepper Sauce", "Eggs with Serrano Ham and Manchego Cheese Green Olive Relish and Migas" and "Rustic Fig Serrano Ham & Hazelnut Tartines" have the lowest amount of protein amongst the unhealthiest recipes for the heart, making them not fit for consumption.

F. Fiber (g):



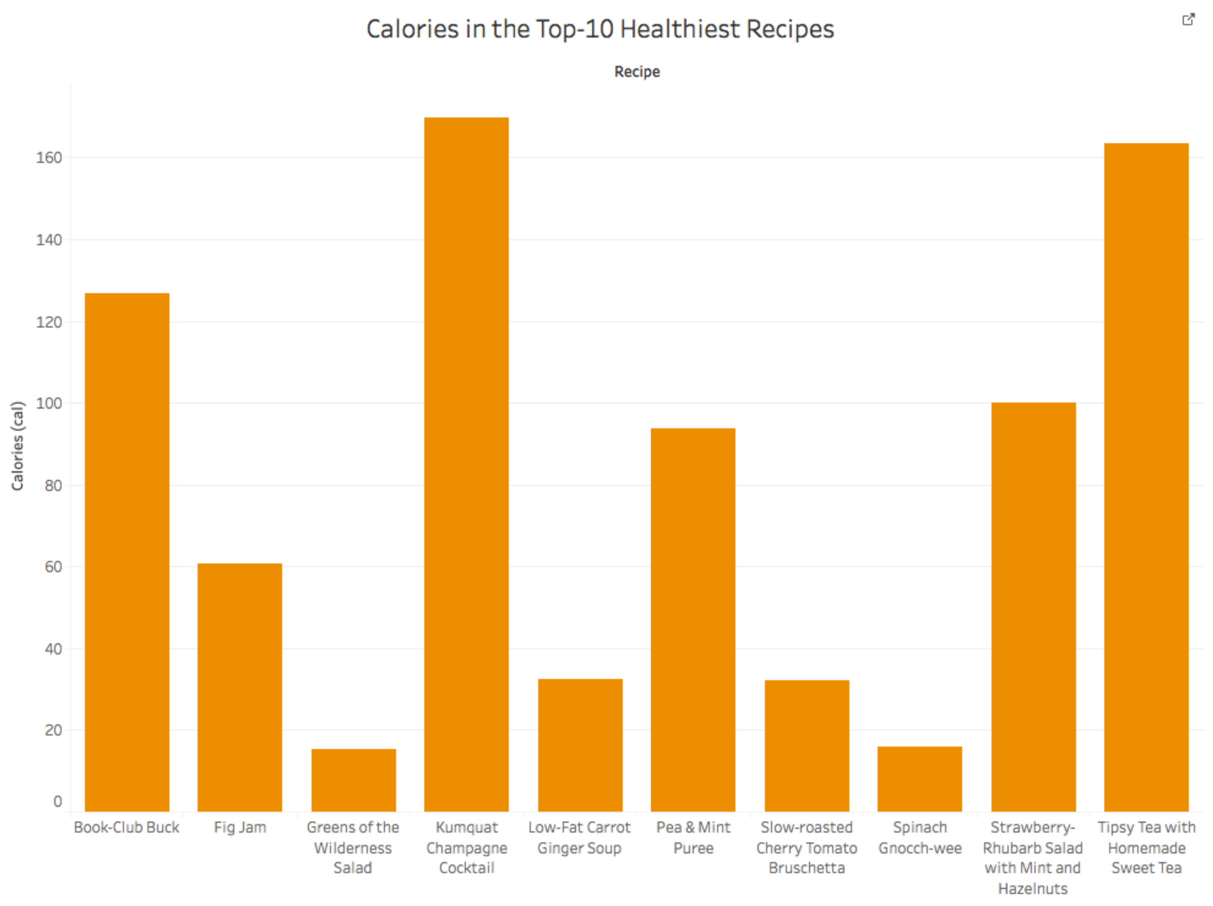
Finding: The above bar chart indicates that the “Strawberry Rhubarb Salad with Mint and Hazelnuts”, “Herb Salad with Pickled Red Onion and Preserved Lemon” and “Sausages Braised in Beer Apples and Onions” have the highest amount of fiber amongst the healthiest recipes for the heart, making them a good choice for consumption.



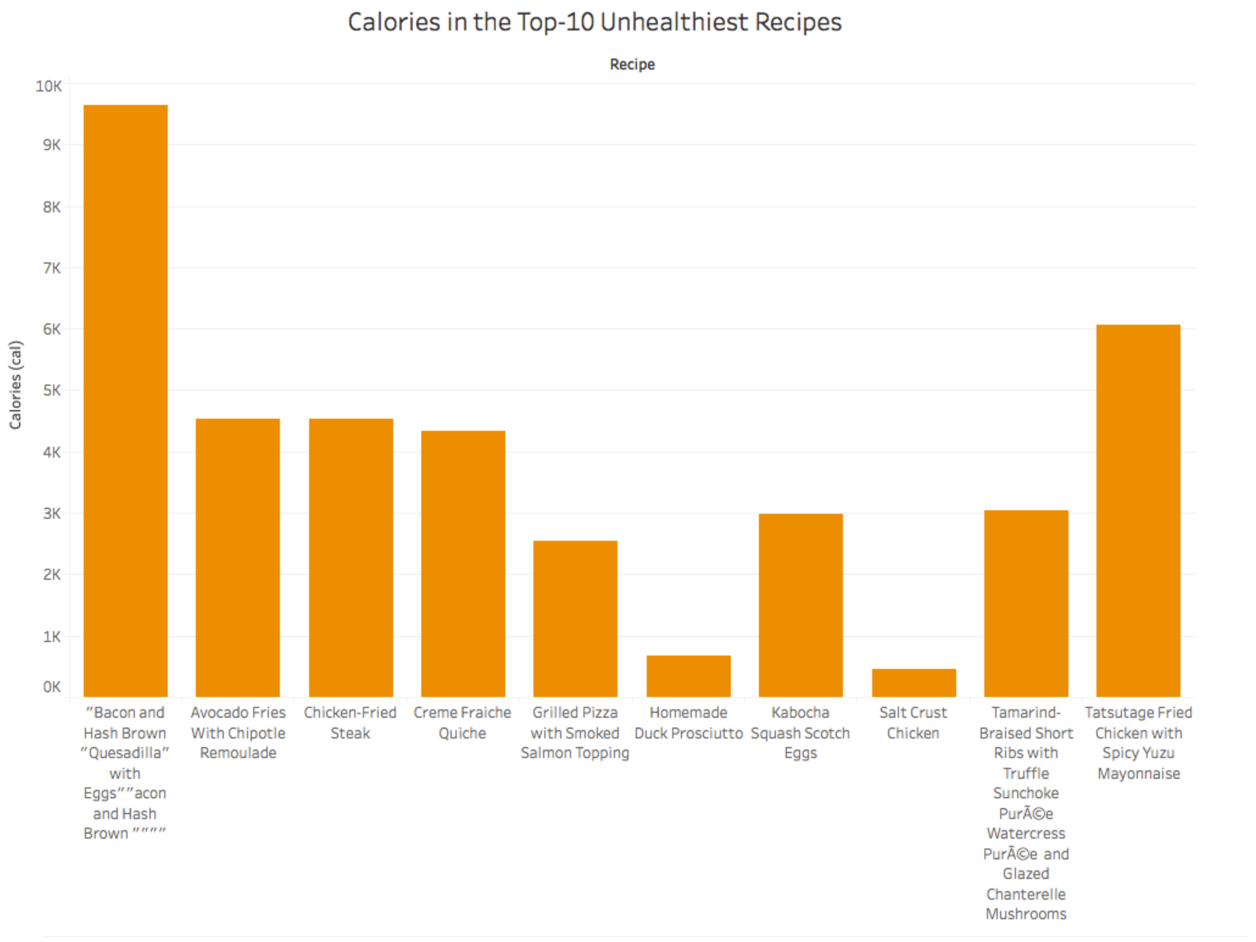
Finding: From the above bar chart, we can observe that the "Prosciutto and Arugula Pizza", "Golden Pancakes with Pecorino and Prosciutto", "Ginger Beef Tataki with Lemon-Soy Dipping Sauce" and "Prosciutto and Goat Cheese Timbales with Mixed Greens" have the lowest amount of fiber amongst the unhealthiest recipes for the heart, making them not fit for consumption.

Bar Chart Based Analysis for “Spoonacular" Recipes

A. Calories (cal):

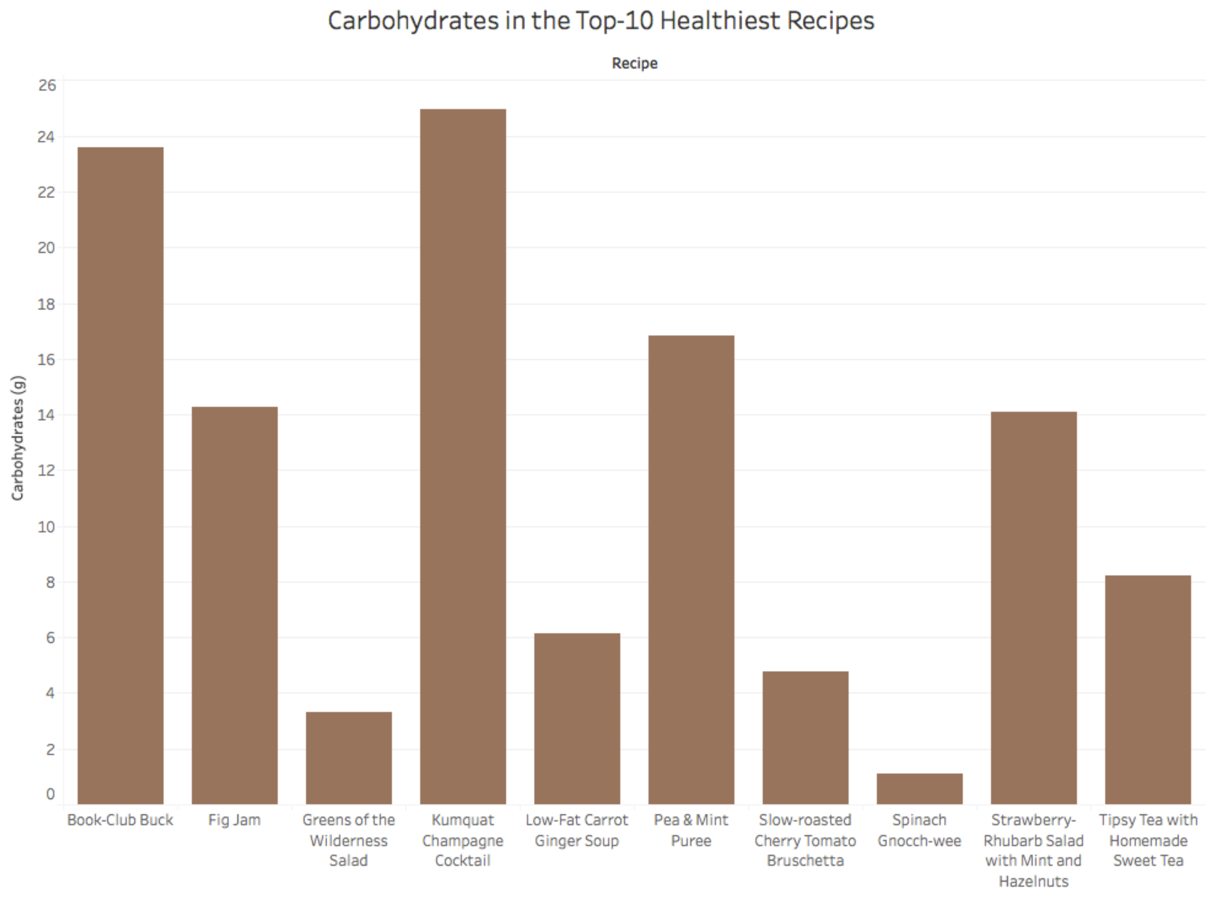


Finding: The above bar chart indicates that the “Greens of the Wilderness Salad”, “Low-Fat Carrot Ginger Soup”, “Slow-roasted Cherry Tomato Bruschetta” and “Spinach Gnocch-wee” have the least number of calories amongst the healthiest heart-healthy recipes obtained from Spoonacular. They are the most suitable for consumption because of their low calorific value.

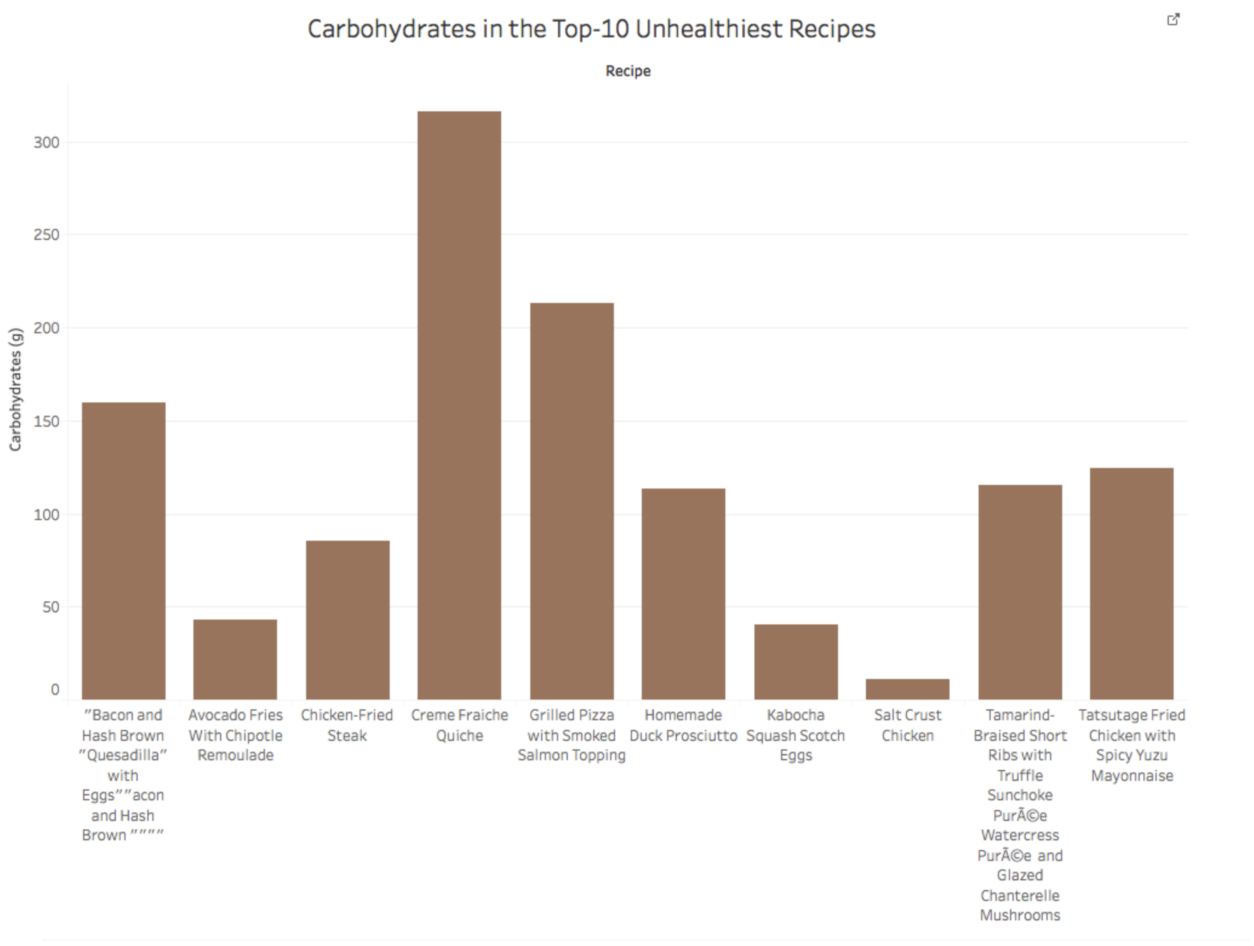


Finding: From the above bar chart, we observe that the "Bacon and Hash Brown Quesadilla with Eggs", "Tatsutage Fried Chicken with Spicy Yuzu Mayonnaise", "Avocado Fries with Chipotle Remoulade" and "Chicken Fried Steak" have the most number of calories amongst the unhealthiest recipes obtained from Spoonacular and should be avoided as much as possible due to their high calorific value.

B. Carbohydrates (g):

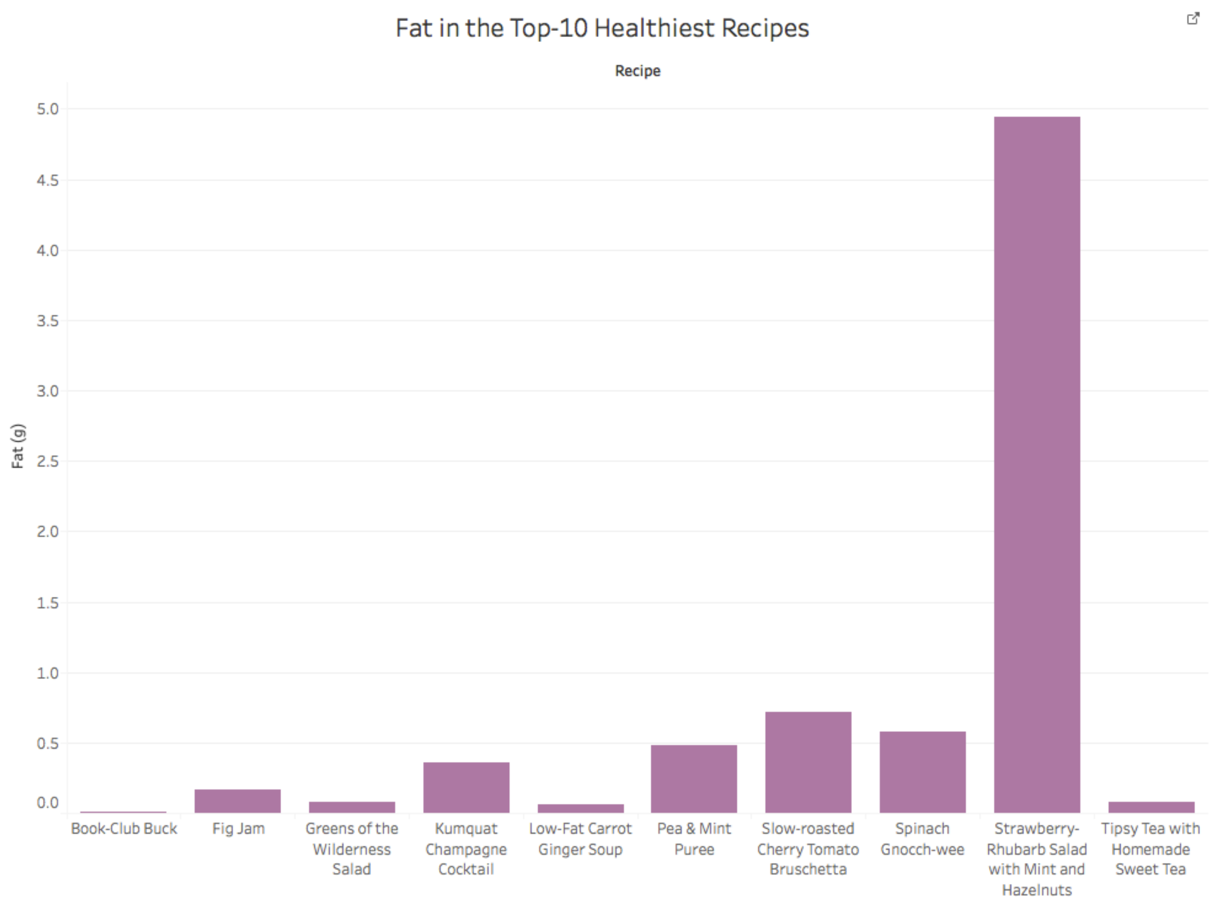


Finding: The above bar chart indicates that the "Greens of the Wilderness Salad", "Low-Fat Carrot Ginger Soup", "Slow-roasted Cherry Tomato Bruschetta" and "Spinach Gnocch-wee" have the least number of calories amongst the top-10 healthiest recipes and are fit for consumption on account of their low calorific value.

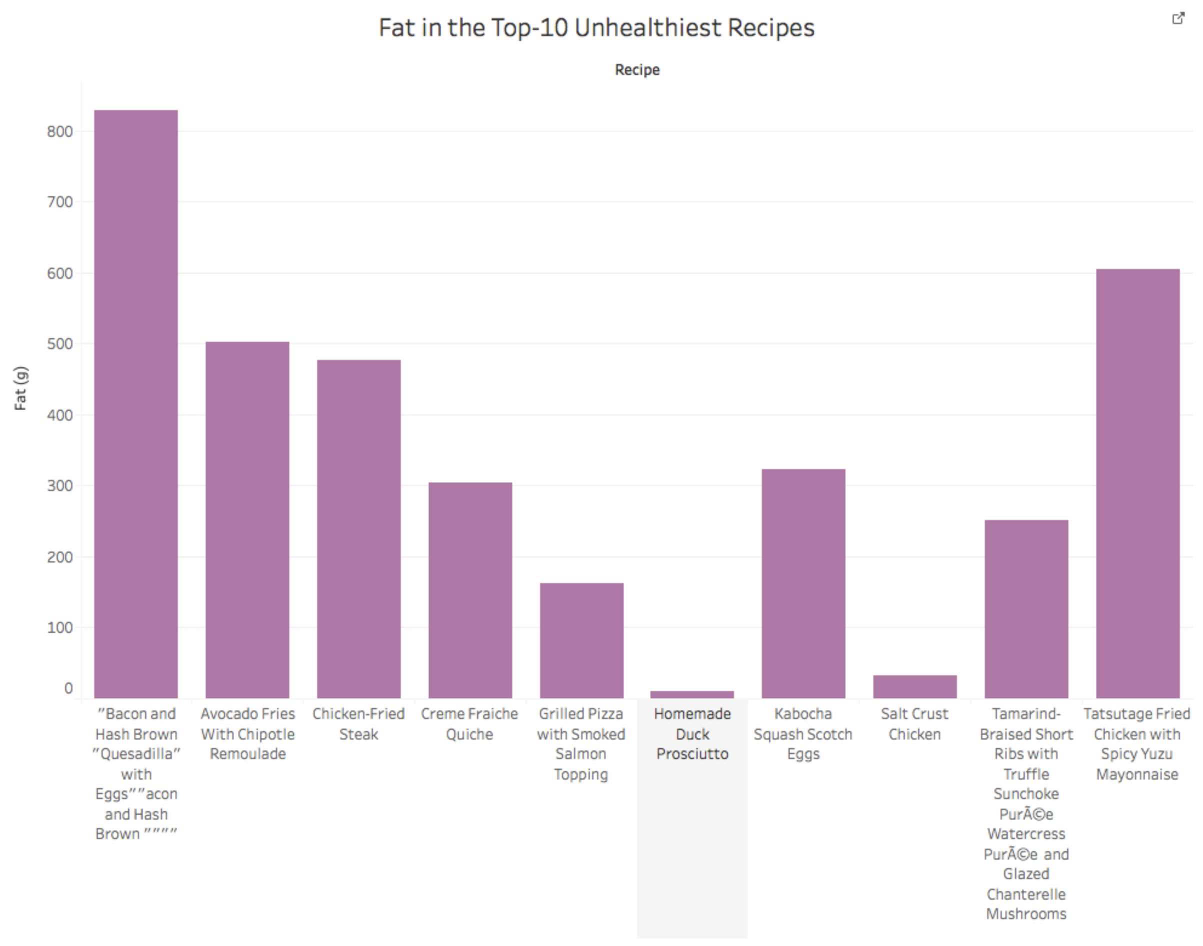


Finding: From the above bar chart, we can observe that the "Creme Fraiche Quiche", "Grilled Pizza with Smoked Salmon Topping" and "Bacon and Hash Brown Quesadilla with Eggs" have the most number of carbohydrates amongst the top unhealthiest recipes and should be avoided as much as possible due to their high carbohydrate content.

C. Fat (g):

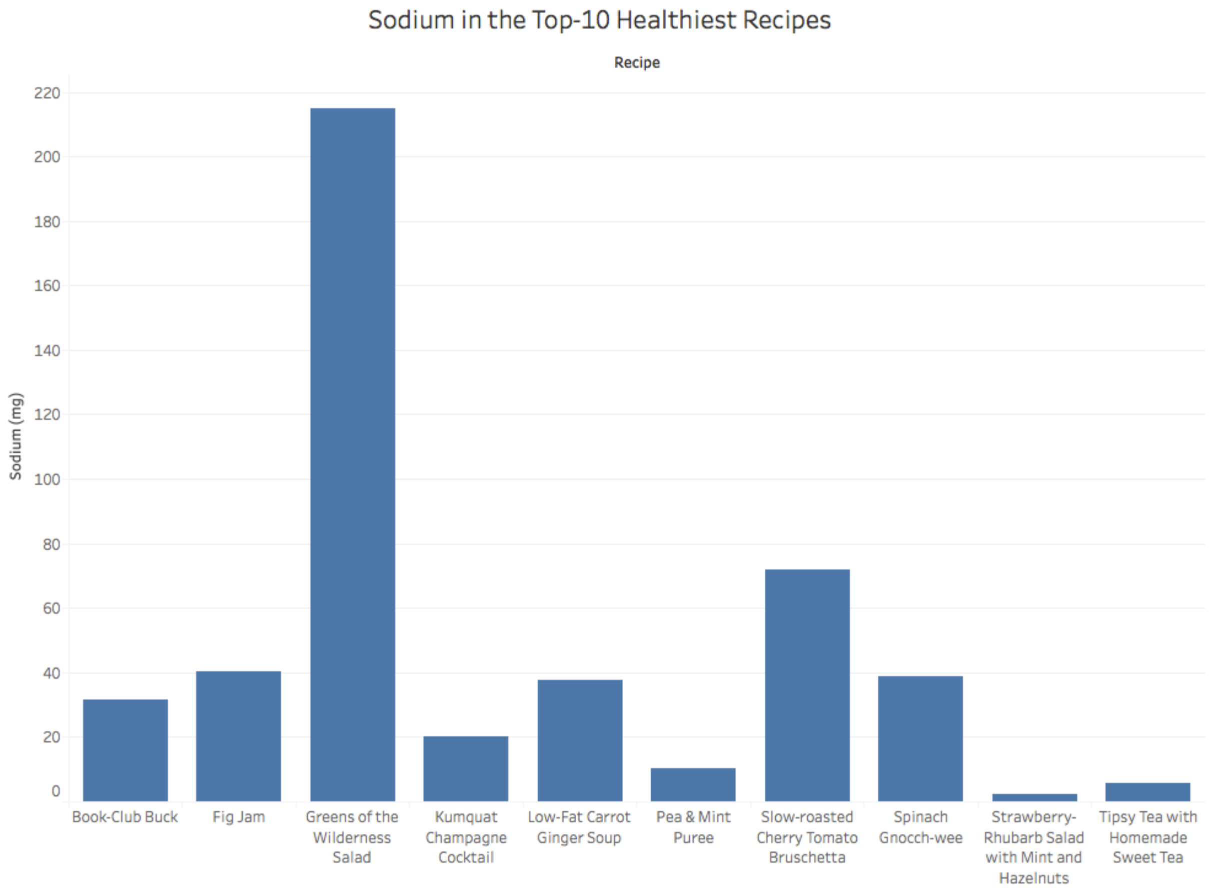


Finding: The above bar chart indicates that the "Book-Club Buck", "Greens of the Wilderness Salad", "Low-Fat Carrot Ginger Soup" and "Tipsy Tea with Homemade Sweet Tea" have the least amount of fat amongst the top healthiest recipes and are hence fit for consumption.

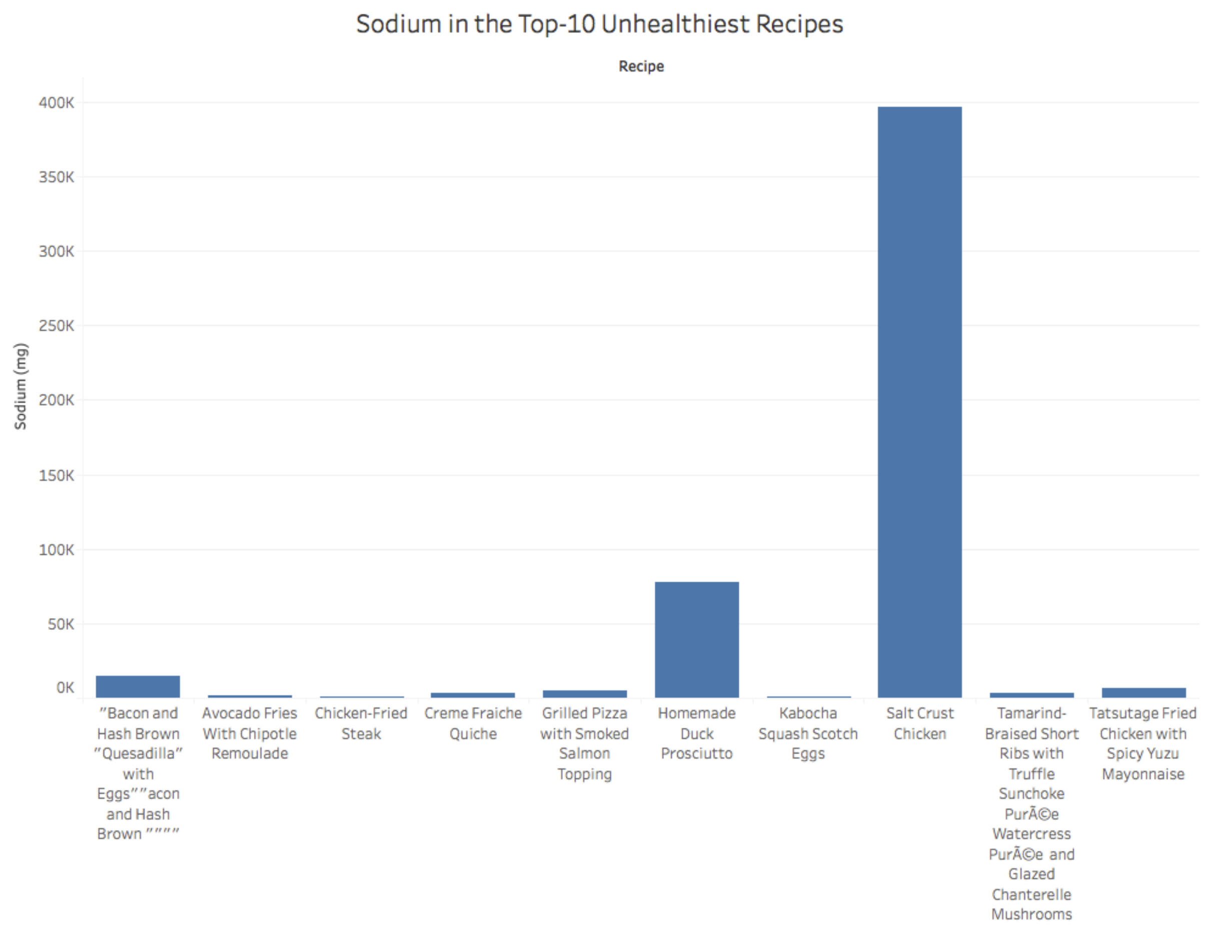


Finding: From the above bar chart, we can observe that the "Bacon and Hash Brown Quesadilla with Eggs", "Avocado Fries with Chipotle Remoulade" and "Tatsutage Fried Chicken with Spicy Yuzu Mayonnaise" have the highest amount of fat amongst the top unhealthiest recipes and should be avoided as much as possible for consumption.

D. Sodium (mg):

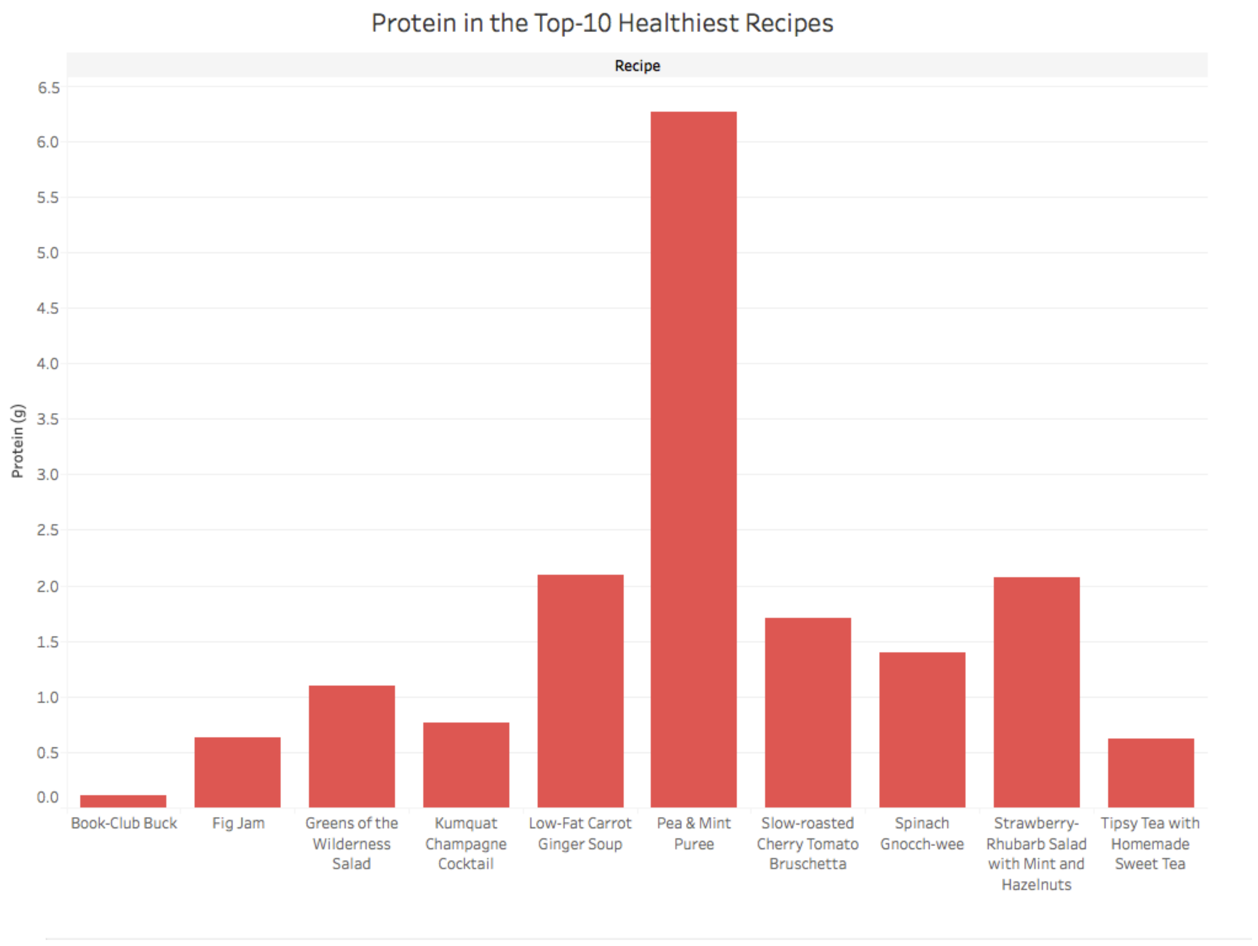


Finding: The above bar chart indicates that the “Strawberry Rhubarb Salad with Mint and Hazelnuts”, “Tipsy Tea with Homemade Sweet Tea” and “Pea & Mint Puree” have the lowest amount of sodium amongst the top healthiest recipes and are hence fit for consumption.

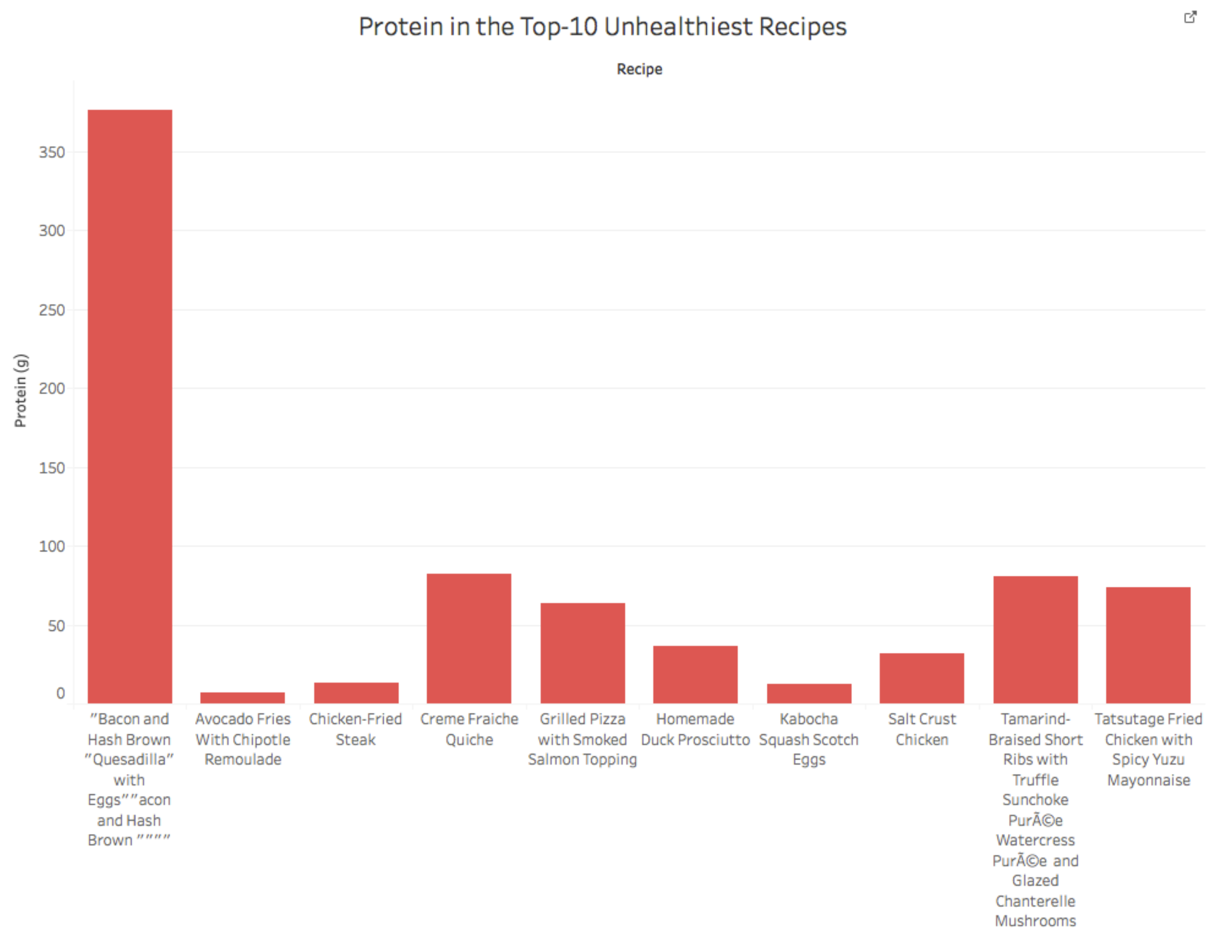


Finding: From the above bar chart, we can observe that the "Salt Crust Chicken" and "Homemade Duck Prosciutto" have the highest amount of sodium amongst the top unhealthiest recipes and should be hence avoided as much as possible.

E. Protein (g):

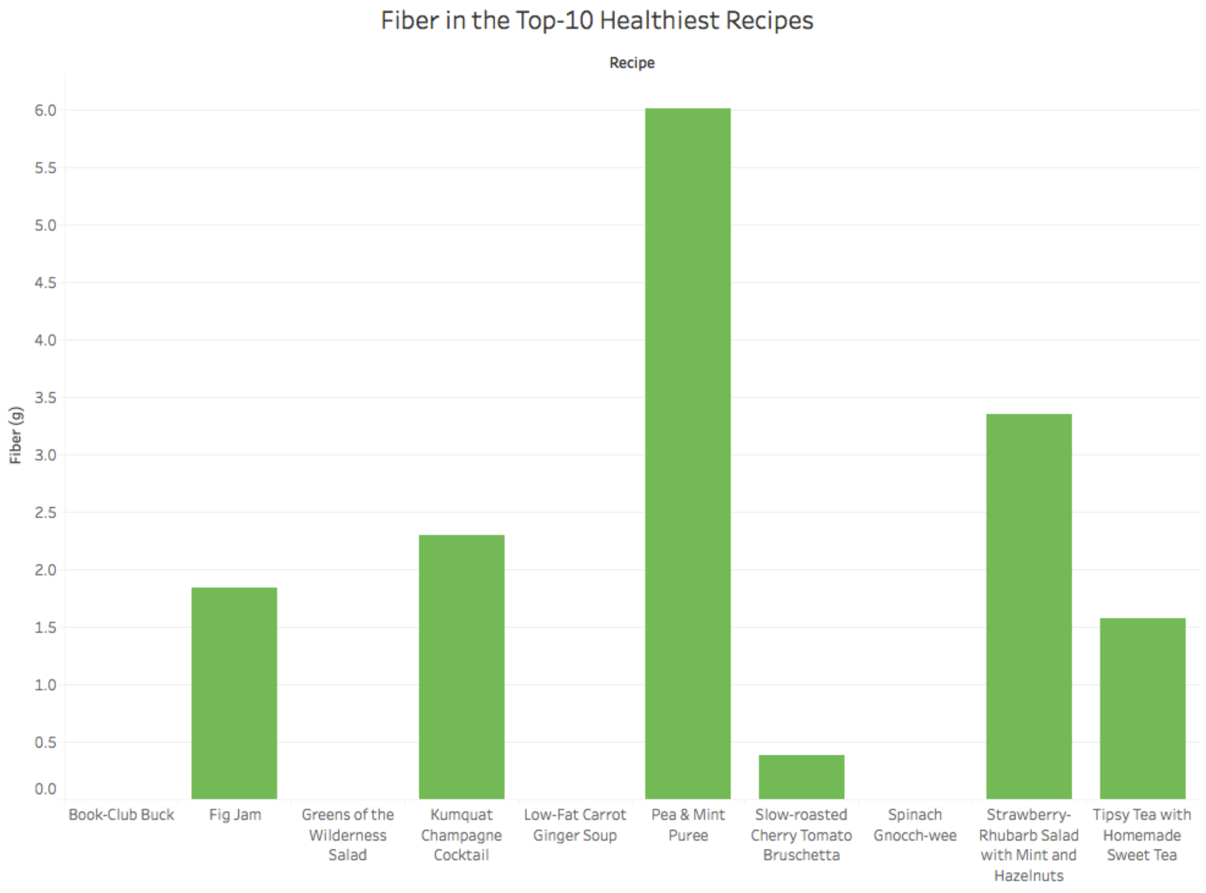


Finding: The above bar chart indicates that the "Pea & Mint Puree", "Low-Fat Carrot Ginger Soup" and "Strawberry Rhubarb Salad with Mint and Hazelnuts" have the highest amount of protein amongst the top healthiest recipes and are hence fit for consumption.

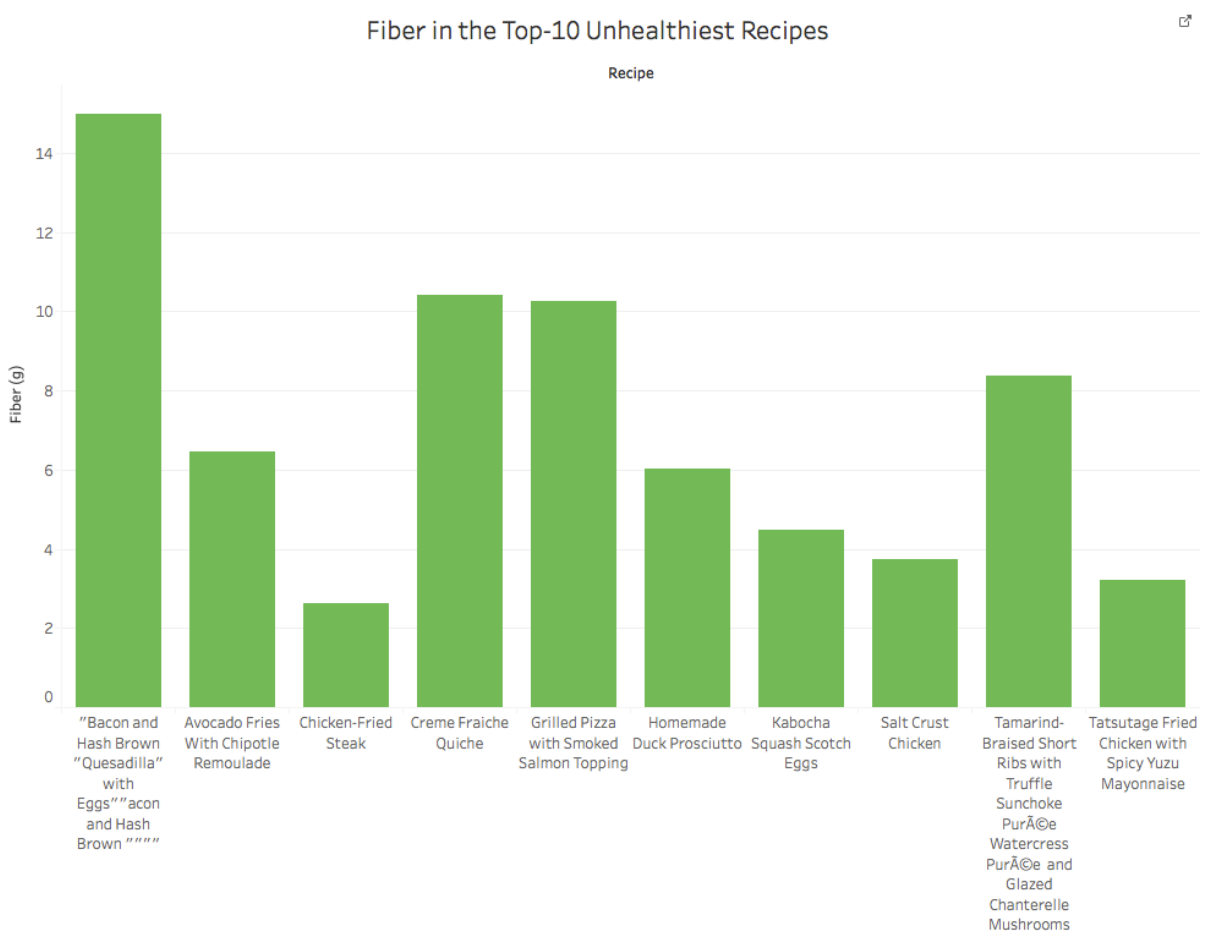


Finding: From the above bar chart, we can observe that the "Avocado Fries with Chipotle Remoulade", "Chicken-Fried Steak" and "Kabocha Squash Scotch Eggs" have the lowest amount of protein amongst the top unhealthiest recipes and should be avoided as much as possible.

F. Fiber (g):



Finding: The above bar chart indicates that the "Pea & Mint Puree", "Strawberry-Rhubarb Salad with Mint and Hazelnuts" and "Kumquat Champagne Cocktail" have the highest amount of fiber amongst the top healthiest recipes and should be hence fit for consumption.



Finding: From the above bar chart, we can observe that the "Chicken-Fried Steak", "Tatsutage Fried Chicken with Spicy Yuzu Mayonnaise" and "Salt Crust Chicken" have the lowest amount of fiber amongst the top unhealthiest recipes and should be avoided from consumption as much as possible.

Challenges & Solutions

The challenges that I encountered while performing my analysis were as follows —

1. Initially, my goal was to scrape the ingredients from the “Epicurious” website and find their nutritional information from the “Spoonacular API” to show how each ingredient was contributing to the overall nutritional profile of the recipes from “Epicurious”. However, this idea had to be scrapped later because there were too many intricacies and discrepancies in the data. Moreover, the nutritional information was not available on “Spoonacular” for all the units of ingredients from “Epicurious", making the resulting information appear incomplete and erroneous. Instead, I decided to focus on choosing recipes from “Spoonacular” having maximum number of common ingredients with those from “Epicurious”.

2. The recipe names from “Epicurious” had a lot of special characters such as “#” and “,” that were causing problems during data processing and analysis. For example, the “#” value was causing an issue while reading the TSV file into a data table in R, etc. I had to ensure to replace all the special characters with spaces before producing the final datasets. Also, I had to encode all of the characters in unicode format (UTF-8) to prevent issues during processing.

3. The Spoonacular API was not free of cost initially. I had to contact the owner of the API on Mashape to obtain a limited student license to make 5000 requests per day to the API. This delayed my data gathering phase by 2-3 weeks in time but I eventually managed to obtain the student license. Please note that this student license has now expired.

4. Some of the recipes from “Spoonacular” were not returned in the correct format (i.e. dictionary) and my code was failing with them. I had to use the try and except clauses to filter out such anomalous return values and ignore them.

5. Some of the recipes from “Epicurious” had no nutritional information and some of them had “NaN” values in their nutritional information. I had to ignore such recipes because they were of no utility in my analysis.

6. The nutrients from “Epicurious" also had their units and daily recommended values concatenated to them and I had to separate the integer part from the units and % values. I accomplished this by using the split() function in Python.

7. Including the sugar (g) content from the recipes in the analysis would have been advantageous but this information was missing from the “Epicurious" dataset and I had to hence ignore the sugar content entirely from my analysis. I consider this as a major drawback in my report as sugar is a big culprit in heart-disease.