MentorDiet

Food Classification with Calorie Analysis

Final Paper

Darshika Kesarwani, Michelle Taylor, Poonam Angne, Rohit Shekhar,

CIS 8395

Big Data Analytics Experience

Table of Contents

Abstract

By 2020, an estimated two-thirds of the global burden of disease will be caused by chronic non-communicable diseases, most of which are associated with diet. While hunger is a tremendous global health concern that cannot be minimized, overnutrition should similarly be addressed as a top priority. Nowadays, people especially millennials are getting more conscious when it comes to overnutrition, what, when and how much their consumption is, and overall healthy well-being in general *("Functional Foods: Key Trends & Developments in Ingredients", 2014)*. This culture has been supported by the rise of food trackers and often with calories embedded next to the restaurant's menu. This lifestyle could only be more effective for a larger audience if there is an easier way to log your meal, as easy as taking your food picture and uploading it on Instagram. To address this problem, we are building an application that automatically identifies the food calories from the food images that the user uploads. The application analyzes the photo of the meal and compares it against images from the existing data sources. We are also looking at providing common ingredient details to help people with health-specific and allergic concern. This application helps users to log their meals, track calories, stay on track with customized targets based on their weight loss goals, keep a visual diary of their meals and more.

Features of the Data

- Our data is in the form of images, text, numbers and more
- Images data consist of the food item images
- The textual data are the image label and food portion sizes
- Numeric data is the nutritional facts of the food

Data Sourcing

We collected our data from four different sources per below:

Primary Data Source for Images - Food 101 Dataset

Our primary data has been collected from Kaggle. The dataset consists of 101 different types of food, ranging from steak all the way to bibimbap, Korean traditional dish. Each type of food consists of 1,000 different images (750 train sample and 250 test sample), which combined together in a total of 101,000 different images.

The dataset structure is as follows:

1. train.txt — the list of images that belong to the training set
2. test.txt — the list of images that belong to the test set
3. classes.txt — the list of all classes of food

Secondary Data Source for Images - Our Own Food Picture Dataset

Additionally, we kept adding more images of data on our own. We added 19 different food types from different cuisines. This helped us in our first experiment by creating our own dataset. We realized that the amount of data that we added would not be as huge as the available dataset. Therefore, we utilized Generative Adversarial Networks (GANs) for data augmentation, which created the same amount of data as the available dataset. Thus, it would eliminate the bias that might initially occur.

Primary Data Source for Nutrition Data – USDA Food Composition Database

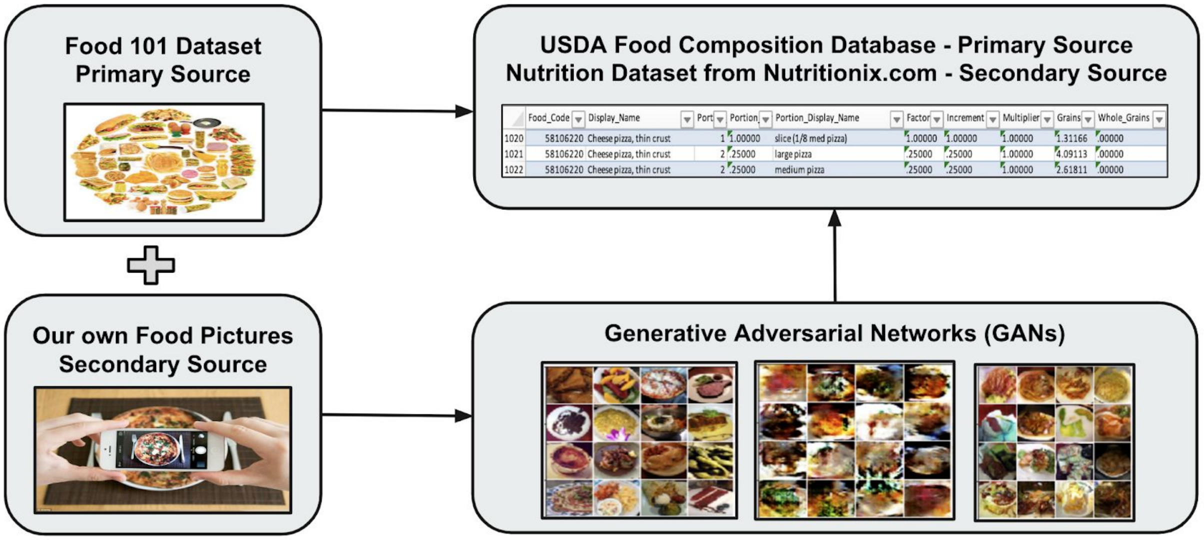
We have collected nutrition data from the USDA Food Composition Database as our primary source. The database consists of different manufacturers and ways of serving and we are planning to use the general one. We combined data from both the primary and secondary data sources into a final image dataset

Secondary Data Source for Nutrition Data - Nutritionix Database

We were in the anticipation that not all food that is available in the image dataset will have the nutritional facts on the USDA Database. Therefore, we looked at another data source from Nutritionix.com, which is also a renowned database for food nutritional facts.

Based on our generic audience for whom we are building the application, we are choosing the main components of nutritional facts that people value the most: *Energy (Calories), Protein, Total Lipid (Fat), Carbohydrate, Fiber (total dietary), Sugars and Ingredients*. We combined data from both the primary and secondary Nutrition data sources into a final Nutrition dataset with selected variables for "per serving standard" data.

Data Sourcing Flow Diagram


Food 101 Dataset
Primary Source
Our own Food Pictures
Secondary Source
USDA Food Composition Database - Primary Source
Nutrition Dataset from Nutritionix.com - Secondary Source
Generative Adversarial Networks (GANs)

	Food_Code	Display_Name	Port	Portion_	Portion_Display_Name	Factor	Increment	Multiplier	Grains	Whole_Grains
1020	58106220	Cheese pizza, thin crust	1	1.00000	slice (1/8 med pizza)	1.00000	1.00000	1.00000	1.31166	.00000
1021	58106220	Cheese pizza, thin crust	2	.25000	large pizza	.25000	.25000	1.00000	4.09113	.00000
1022	58106220	Cheese pizza, thin crust	2	.25000	medium pizza	.25000	.25000	1.00000	2.61811	.00000

Data Cleansing

Image Preprocessing - using OpenCV library

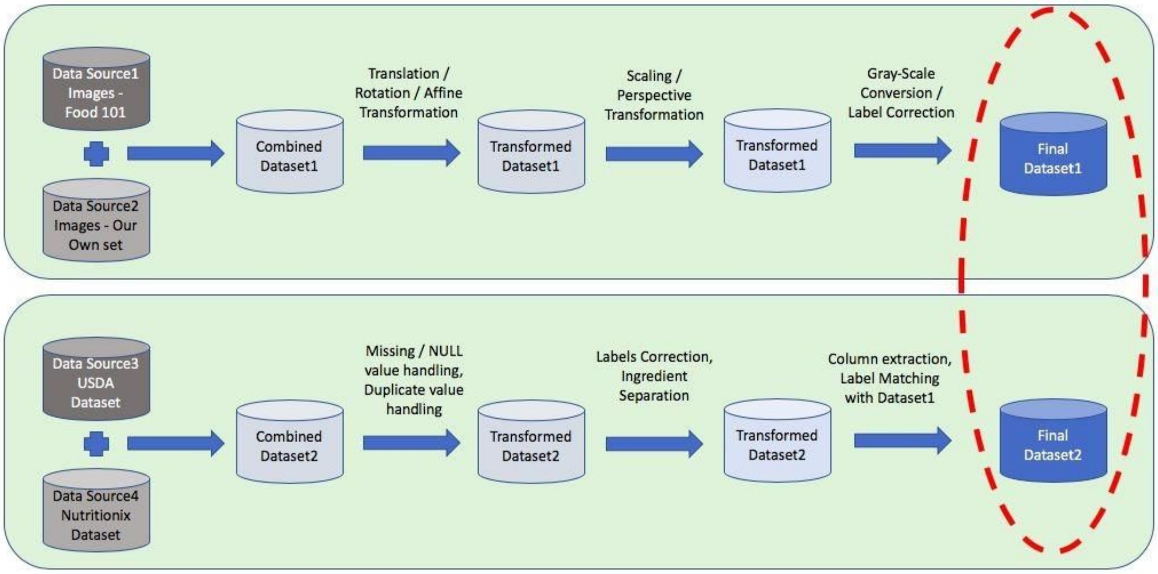
- *Geometric Transformations of Images:* Applying a different geometric transformation to images like translation, rotation (Rotation of an image for an angle), affine transformation

- *Scaling:* Scaling the image by specifying the size of the image manually or by specifying the scaling factor. Different interpolation methods will be used
- *Perspective Transformation:* Calculating a perspective transform from four pairs of the corresponding points
- *Gray-Scale Format:* A colored Image is made up of 3 channels, i.e 3 arrays of red, green and blue pixel values. We used 1 channel which would read our images in a gray-scale format

Nutrition Data Cleansing

- *Missing Values:* We looked at different datasets to get nutrition and calorie information for the food items. But these datasets have missing or NULL values. So we compared both the datasets to see if one missing value is available in the other and replace it. When we were unable to find a replacement within both the datasets, then we looked for the information from another source if required
- *Duplicate Values:* There were duplicate columns in the datasets which were also be removed
- *Label Correction:* Some of the labels are incorrect, which were corrected
- *Column Extraction:* We kept only the required columns and remove all others to make the dataset concise
- *Label Matching:* The label name was matched with the images dataset to include all corresponding labels

Data Cleansing Flow Diagram

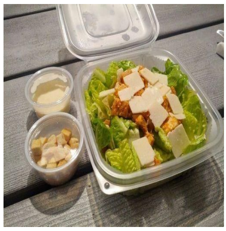

Data Source1 Images - Food 101
Data Source2 Images - Our Own set
Combined Dataset1
Translation / Rotation / Affine Transformation
Transformed Dataset1
Scaling / Perspective Transformation
Transformed Dataset1
Gray-Scale Conversion / Label Correction
Final Dataset1
Data Source3 USDA Dataset
Data Source4 Nutritionix Dataset
Combined Dataset2
Missing / NULL value handling, Duplicate value handling
Transformed Dataset2
Labels Correction, Ingredient Separation
Transformed Dataset2
Column extraction, Label Matching with Dataset1
Final Dataset2

Data Sample

Food 101 Dataset Description

- Training images had some noise like intense color, additional food items, and sometimes wrong label.
- All images were rescaled to have a maximum side length of 512 pixels.
- Size of the dataset is around 5GB

Label Name	**Image1**	**Image2**	**Image3**	**Image4**
deviled_eggs				

caesar_salad				
lasagna				

USDA Dataset Description

Data for each food item contains details of nutrient value for the standard serving quantity.

Our transformed data contain the following columns:

1. Energy
2. Protein
3. Total lipid (fat)
4. Carbohydrate
5. Fiber
6. Sugars
7. Ingredients

Raw Data Sample

Nutrient data for: CAESAR SALAD, UPC: 085239009802					
Food Group: Branded Food Products Database					

Common Name:					
Nutrient	**Unit**	**Data points**	**Std. Error**	**1 SALAD = 99.0g**	**1Value per 100 g**
Proximates					
Energy	kcal	--	--	290	293
Protein	g	--	--	7	7.07
Total lipid (fat)	g	--	--	24	24.24
Carbohydrate, by difference	g	--	--	13	13.13
Fiber, total dietary	g	--	--	1	1
Sugars, total	g	--	--	3	3.03
Minerals					
Calcium, Ca	mg	--	--	150	152
Iron, Fe	mg	--	--	0.36	0.36
Sodium, Na	mg	--	--	700	707
Vitamins					
Vitamin C, total ascorbic acid	mg	--	--	1.2	1.2
Vitamin A, IU	IU	--	--	3500	3535

Transformed Data Sample

Label	**Serving Standard**	**Per in Kcal**	**Protein**	**Total lipid (fat)**	**Carbohydrate**	**Fiber**	**Sugars**	**Ingredients**
deviled_eggs		80	7	24	13	1	3	BOILED EGGS, DRESSING

caesar_salad	1 SALAD = 99.0g	290	10.71	25	0	0	0	ROMAINE LETTUCE, CAESAR DRESSING
lasagna		301	6.51	5.58	16.28	1.4	5.12	TOMATO PUREE (WATER, TOMATO PASTE), COOKED LASAGNA PASTA

Delivery (ETL)

We have two different types of data: food images (unstructured data) and nutrient dataset (structured data).

Food Images Dataset:

The 101,000 food images (101 different types and 1,000 images for each type) were being acquired from Kaggle. Although we mentioned before that the dataset acquired have been split into 750 train data and 250 test data, but we are going to split the data differently since we are combining our own food images (18 types of food and 10 images each, 180 in total) into the dataset, to make it more dynamic. We added more images in the future to increase the robustness of the final model for image classification. Although, in the current scope, we added only the mentioned 19 types of food images, which were currently in different sizes and resolutions, which we processed in the next steps. Each image has an image label that was matched with the respective information in the nutrient dataset.

Nutrient Dataset:

For our nutrient dataset, we used the USDA database as our primary source and Nutritionix.com database as our secondary source. From the USDA database, we downloaded the food nutrient information in the form of CSV by searching the image label for each of our 120 food images. Nutrient information for some image labels was not available on the USDA database, which is when we referred to the Nutritionix database. The Nutritionix database did not provide a CSV file download feature, hence information was entered manually into the nutrition dataset for that image label. Based on our generic audience for whom we are building the application, we extracted only the main components of nutritional facts that people value the most: *Energy (Calories), Protein, Total Lipid (Fat), Carbohydrate, Fiber (total dietary), Sugars, Ingredients, along with basic info like label, portion and serving standard*. Each of these components were extracted from the individual food label CSV into a nutrient dataset CSV file. In a real scenario, we would utilize the available APIs for the Nutritionix database to get the nutrition information and store it in the nutrition dataset.

Transform

Food Images Dataset:

For the newly added 19 food types with 10 images each, we used Generative Adversarial Networks (GANs) to create new images eerily similar to the corresponding food type. A GAN posits a multidimensional latent space and uses deep learning to learn a mapping from this latent space to the domain of interest, in our case a food image type. GANs can produce the most life-like, realistic images with sharp edges and defined contours.

GANs are composed of two models:

Generator:

It aims to generate new data similar to the expected one. Generative Network takes a latent variable vector as input and returns a valued vector, which corresponds to a flattened image. The output is itself a new image.

Discriminator:

It aims to recognize if an input data is ‘real’ — belongs to the original dataset — or if it is ‘fake’ — generated by a forger. This network takes a flattened image as its input, and return the probability of it belonging to the real dataset, or the newly created dataset. The structure of this network has hidden layers, each followed by a dropout layer to prevent overfitting.

The Generator needs to learn how to create data in such a way that the Discriminator isn’t able to distinguish it as fake anymore. The competition between these two teams is what improves their knowledge until the Generator succeeds in creating realistic data.

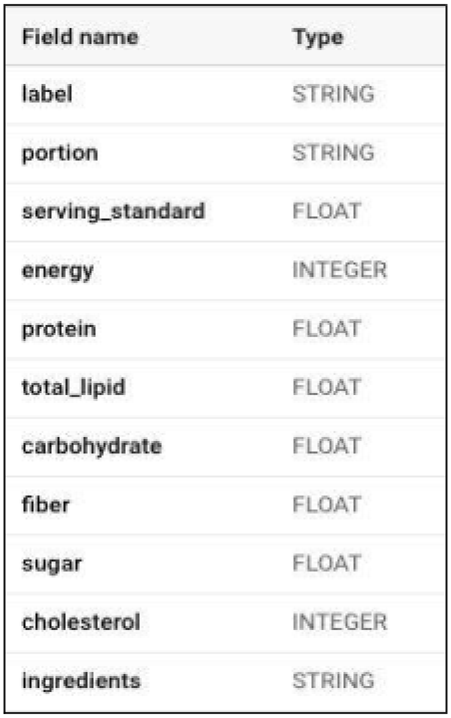
Thus, with the help of GAN’s we created additional images (1000 of each type to match with master data) for all the 19 food types and adding them to our master images dataset.

Nutrient Dataset:

Firstly, we modified the nutrient dataset features into a single column name for easier processing. Features such as serving standard and total lipid are being modified by adding an underscore to replace the space. The features were renamed as label, portion, serving_standard, energy, protein, total_lipid, carbohydrate, fiber, sugar, cholesterol and ingredients.

Similarly, the 120 food labels were also renamed to remove spaces if present and replaced with an underscore. We ensured that the image labels are unique and no duplicates are present since the image labels will act as the primary key in the database.

Nutrient Dataset Schema



Field name	Type
label	STRING
portion	STRING
serving_standard	FLOAT
energy	INTEGER
protein	FLOAT
total_lipid	FLOAT
carbohydrate	FLOAT
fiber	FLOAT
sugar	FLOAT
cholesterol	INTEGER
ingredients	STRING

Load

We used Google Cloud Platform (GCP) for both images and dataset storage as well as the analytics. We were maintaining centralized storage due to the scope of our project. The nutrient data was loaded on Google BigQuery in order to avoid periodical management of the database as BigQuery provides the key features below:

- BigQuery is a fully managed service provided by Google
- BigQuery automatically replicates data between zones to enable high availability. It also automatically load balances to provide optimal performance and to minimize the impact of any hardware failures
- Flexible pricing models which are suitable for our project

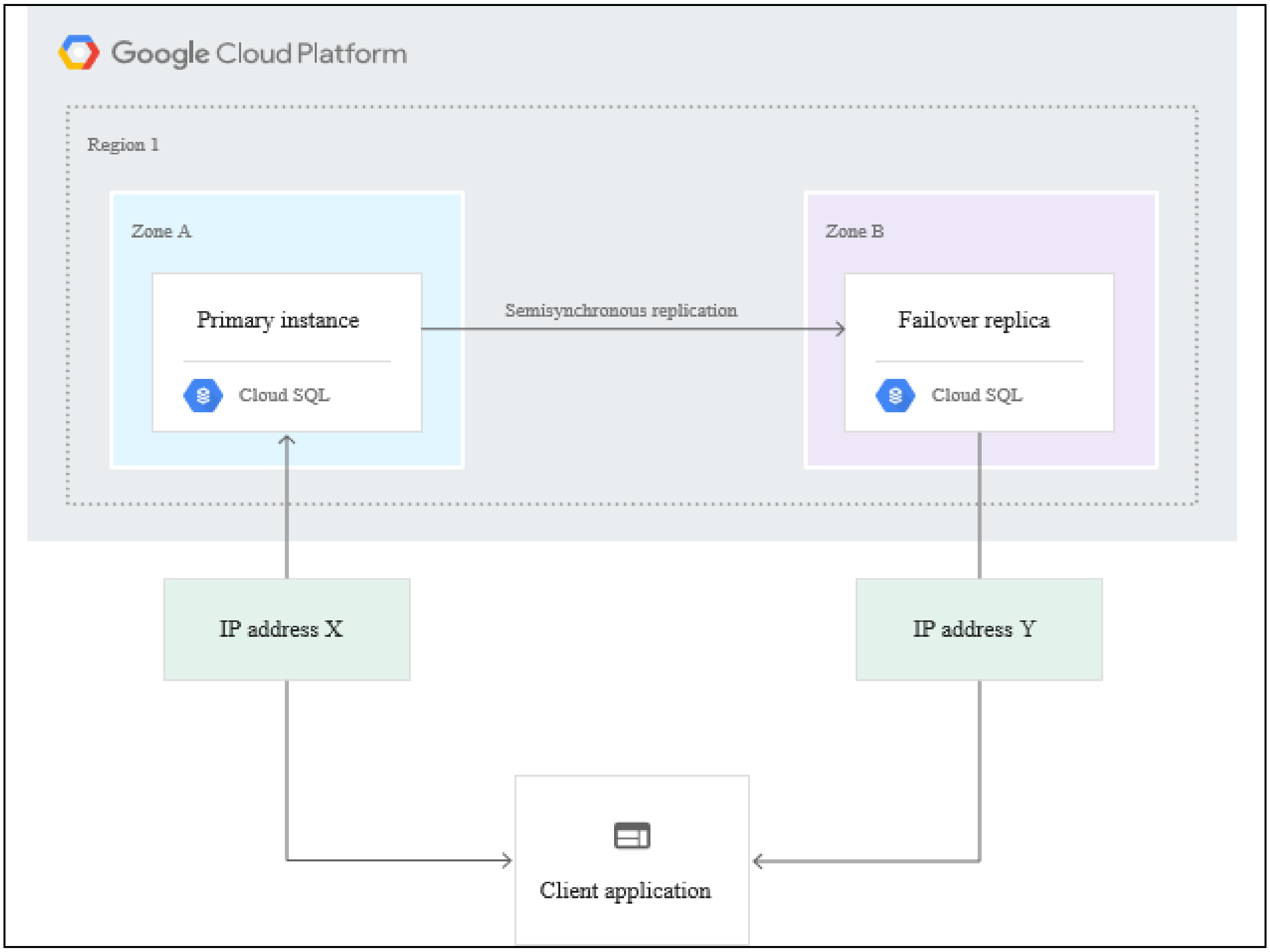
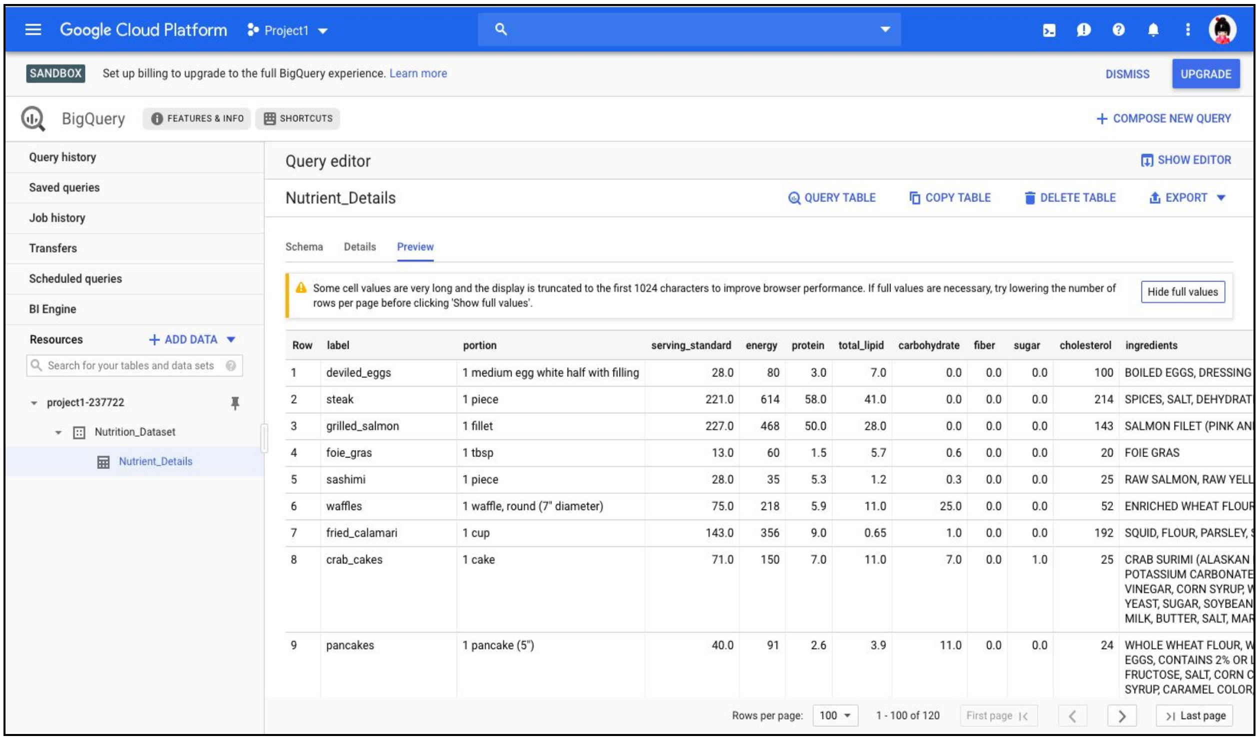


Figure 1: *Automatic Replication between zones in BigQuery*



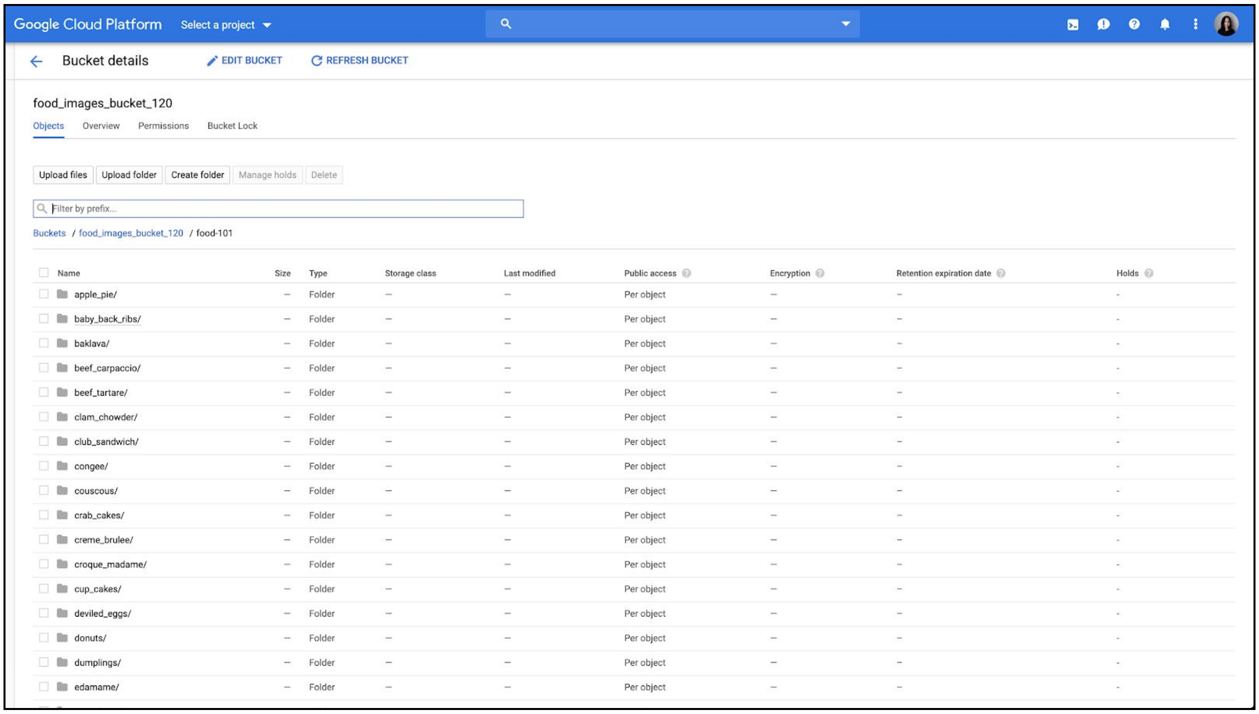
Some cell values are very long and the display is truncated to the first 1024 characters to improve browser performance. If full values are necessary, try lowering the number of rows per page before clicking 'Show full values'.

Row	label	portion	serving_standard	energy	protein	total_lipid	carbohydrate	fiber	sugar	cholesterol	ingredients
1	deviled_eggs	1 medium egg white half with filling	28.0	80	3.0	7.0	0.0	0.0	0.0	100	BOILED EGGS, DRESSING
2	steak	1 piece	221.0	614	58.0	41.0	0.0	0.0	0.0	214	SPICES, SALT, DEHYDRAT
3	grilled_salmon	1 fillet	227.0	468	50.0	28.0	0.0	0.0	0.0	143	SALMON FILET (PINK AN
4	foie_gras	1 tbsp	13.0	60	1.5	5.7	0.6	0.0	0.0	20	FOIE GRAS
5	sashimi	1 piece	28.0	35	5.3	1.2	0.3	0.0	0.0	25	RAW SALMON, RAW YELL
6	waffles	1 waffle, round (7" diameter)	75.0	218	5.9	11.0	25.0	0.0	0.0	52	ENRICHED WHEAT FLOUR
7	fried_calamari	1 cup	143.0	356	9.0	0.65	1.0	0.0	0.0	192	SQUID, FLOUR, PARSLEY,
8	crab_cakes	1 cake	71.0	150	7.0	11.0	7.0	0.0	1.0	25	CRAB SURIMI (ALASKAN POTASSIUM CARBONATE VINEGAR, CORN SYRUP, W YEAST, SUGAR, SOYBEAN MILK, BUTTER, SALT, MAR
9	pancakes	1 pancake (5")	40.0	91	2.6	3.9	11.0	0.0	0.0	24	WHOLE WHEAT FLOUR, W EGGS, CONTAINS 2% OR L FRUCTOSE, SALT, CORN C SYRUP, CARAMEL COLOR,

Figure 2: *Sample stored dataset in BigQuery*

We uploaded the nutrient CSV file into BigQuery directly, due to faster performance with uncompressed files, since it will be read in parallel. Our file size is less than 60 KB, therefore we were not worried about the bandwidth limitation with using uncompressed files. We decided to use insert (drop table and load again) to add more data because our entire dataset is in CSV file, not only appending individual rows.

Meanwhile, we stored the images dataset as blob storage in Google Cloud Storage. This approach would allow us to directly access the object using API. We had access for the entire bucket and for the BigQuery storage for each of the group member using our email IDs in the IAM console (Identity and Access Management). This approach ensures the data is secured from public or unauthorized access. Lastly, we also have data backups in our Google Drive in case data loss.



Bucket details

food_images_bucket_120

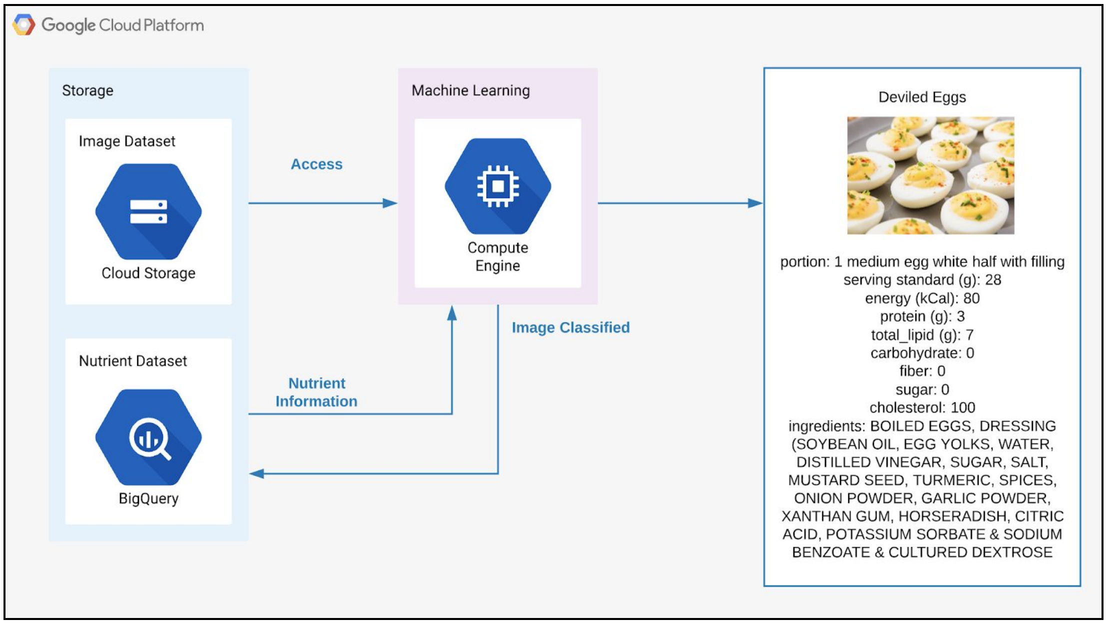
Buckets / food_images_bucket_120 / food-101

□	Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
□	apple_pie/	–	Folder	–	–	Per object	–	–	-
□	baby_back_ribs/	–	Folder	–	–	Per object	–	–	-
□	baklava/	–	Folder	–	–	Per object	–	–	-
□	beef_carpaccio/	–	Folder	–	–	Per object	–	–	-
□	beef_tartare/	–	Folder	–	–	Per object	–	–	-
□	clam_chowder/	–	Folder	–	–	Per object	–	–	-
□	club_sandwich/	–	Folder	–	–	Per object	–	–	-
□	congee/	–	Folder	–	–	Per object	–	–	-
□	couscous/	–	Folder	–	–	Per object	–	–	-
□	crab_cakes/	–	Folder	–	–	Per object	–	–	-
□	creme_brulee/	–	Folder	–	–	Per object	–	–	-
□	croque_madame/	–	Folder	–	–	Per object	–	–	-
□	cup_cakes/	–	Folder	–	–	Per object	–	–	-
□	deviled_eggs/	–	Folder	–	–	Per object	–	–	-
□	donuts/	–	Folder	–	–	Per object	–	–	-
□	dumplings/	–	Folder	–	–	Per object	–	–	-
□	edamame/	–	Folder	–	–	Per object	–	–	-

Figure 3: *Image Dataset on Google Cloud Storage*

Architecture

The machine learning model would run in the compute engine, accessing the image dataset in the Cloud Storage, classifying the image in the compute engine. Once the image has been classified, it went to the BigQuery to get the respective nutrient information, and the result was brought back into the compute engine and from there it generated the result.



portion: 1 medium egg white half with filling
serving standard (g): 28
energy (kCal): 80
protein (g): 3
total_lipid (g): 7
carbohydrate: 0
fiber: 0
sugar: 0
cholesterol: 100
ingredients: BOILED EGGS, DRESSING (SOYBEAN OIL, EGG YOLKS, WATER, DISTILLED VINEGAR, SUGAR, SALT, MUSTARD SEED, TURMERIC, SPICES, ONION POWDER, GARLIC POWDER, XANTHAN GUM, HORSERADISH, CITRIC ACID, POTASSIUM SORBATE & SODIUM BENZOATE & CULTURED DEXTROSE

Figure 4: *Architecture: Image Classification and Nutrient Information*

Operations

Scheduling

Due to our large size image dataset, we uploaded 1000 images daily over a week period of time. Since we also incorporated GANs in our dataset, we planned to add new images and retrain model in order to improve the model. We planned to have a backup for the image dataset on a different region for disaster recovery, similar to what the BigQuery feature has.

Monitoring and Support

We planned to use StackDriver Monitoring powered by Google. Stackdriver Monitoring provides visibility into the performance, uptime, and overall health of cloud-powered applications. Stackdriver collects metrics, events, and metadata from GCP and ingests that data and generates insights via dashboards, charts, and alerts. This approach integrated uptime monitoring and health checks to be notified quickly when endpoints become inaccessible to your users.

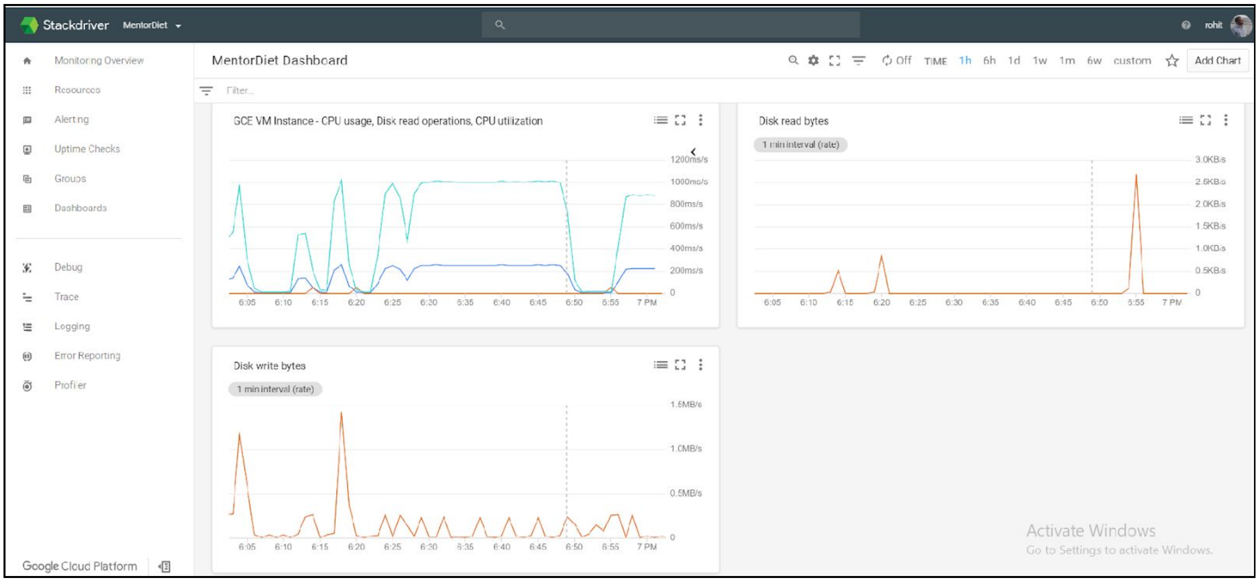


Figure 5: *Stackdriver Monitoring Dashboard on our VM Engine and Data Consumption*

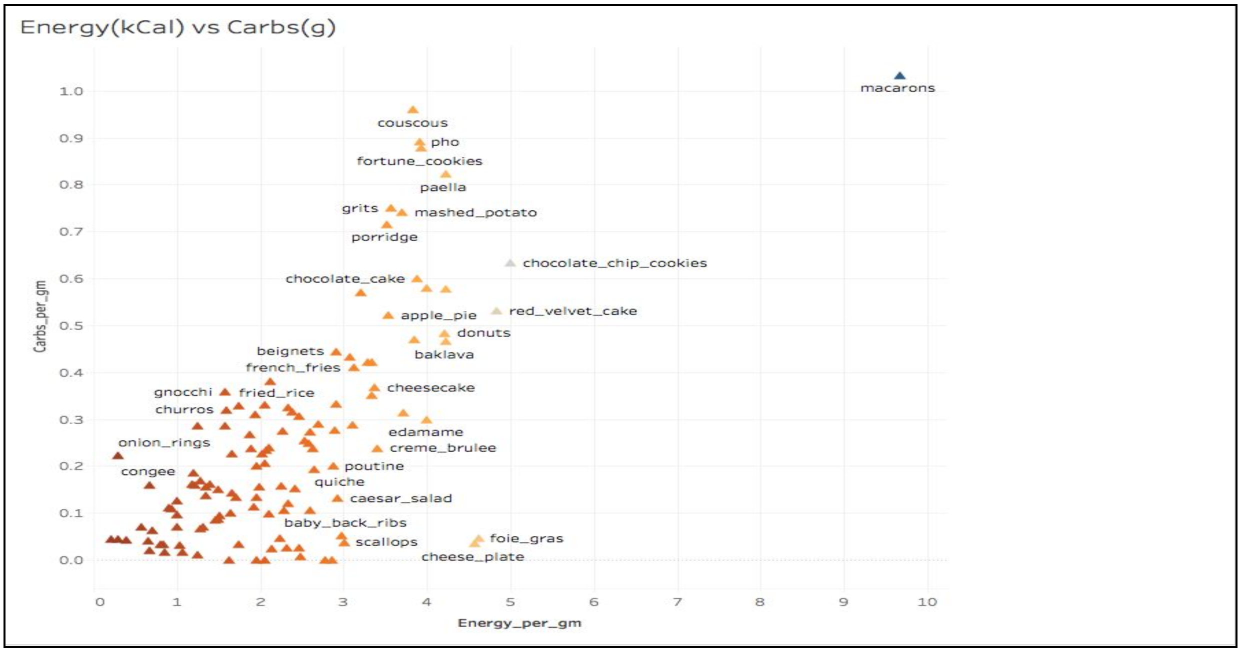
Data Visualization

Data visualization is the key to actionable insights. Visualization allows us to take our complex findings and present them in a way that is informative and engaging to all stakeholders. On the other hand, data analytics helps us to understand the data we have collected and provided a clearer picture of the business conditions that are of the greatest concern to decision-makers.

As our MentorDiet application is targeted towards people who are interested in mindful eating, maintaining a healthy lifestyle, avoiding eating disorders or exceeding required daily calories intake and more, we have created different visualizations in Tableau to understand the data we have collected and to draw preliminary insights from it.

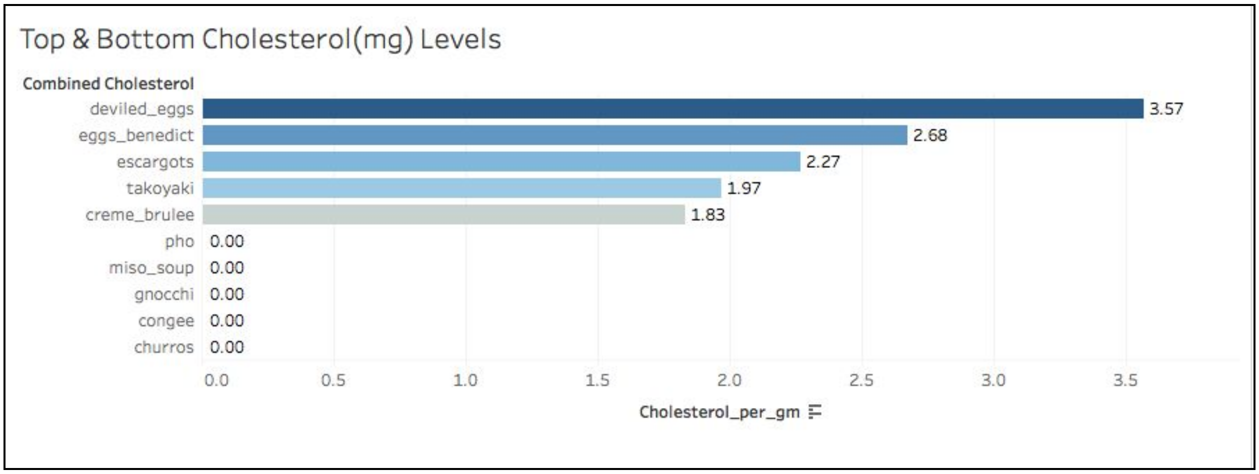
<u>Energy and Carbohydrate Comparison</u>

Each individual has a different requirement when it comes to daily calorie intake and other dietary needs. For instance, if someone wants to monitor their calorie with carb intake and make decisions on which food item would meet their specific needs, the above visualization created using scatter plot would give them a clearer picture. Through this visualization, we can see that macarons are high on calories and carbs whereas congee is low on calories and carbs. Some people prefer to follow a low-calorie high carb diet for losing weight or to maintain a healthy balanced diet in general. Such people would prefer to eat couscous based on the visualization.



<u>Food Items with highest and lowest Cholesterol</u>

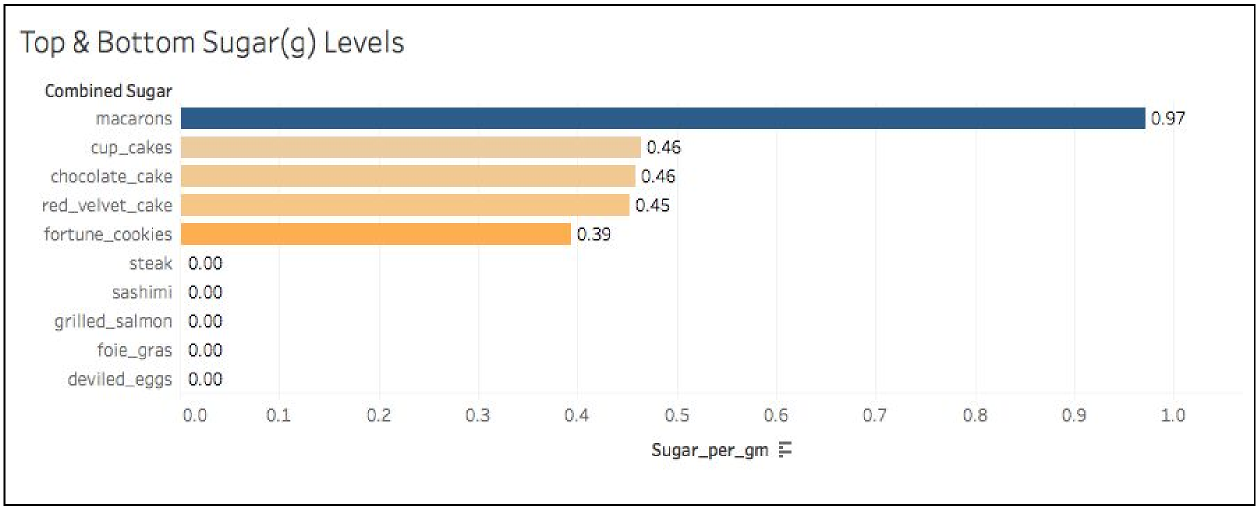
We might want to know which food items have the highest cholesterol content in them and which have the lowest cholesterol. Based on this knowledge, some people might choose if they want to eat a specific food item or not. Many people suffering from heart ailments are advised to reduce their cholesterol intake and hence this visualization created using a horizontal bar chart can be very helpful in such cases.



We can see that deviled eggs and eggs benedict are high on Cholesterol while pho and miso soup are low on cholesterol. So, a health conscious person may avoid deviled eggs and prefer miso soup instead.

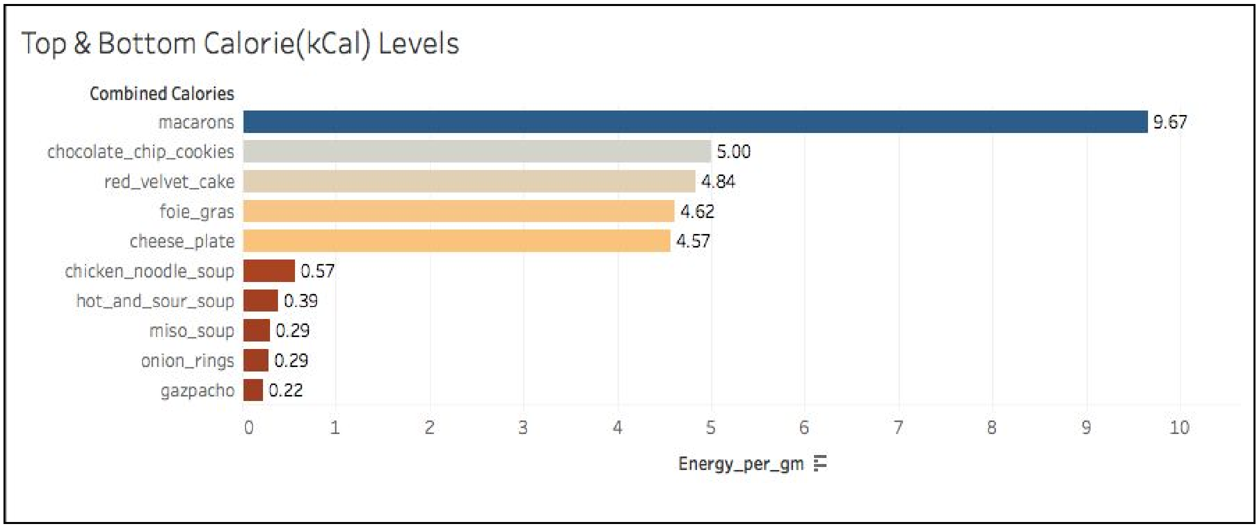
Food Items with highest and lowest Sugar

Most youngsters and females are very conscious of their sugar intake. Sugar is generally thought about as not so good for health and often considered a major cause of weight gain. It would be helpful to know which food items have the highest and lowest sugar levels so that the diet conscious people or people suffering from diabetes can select their food items based on their sugar content. We can see that macarons and cupcakes are high on sugar while steak and sashimi are low on sugar.



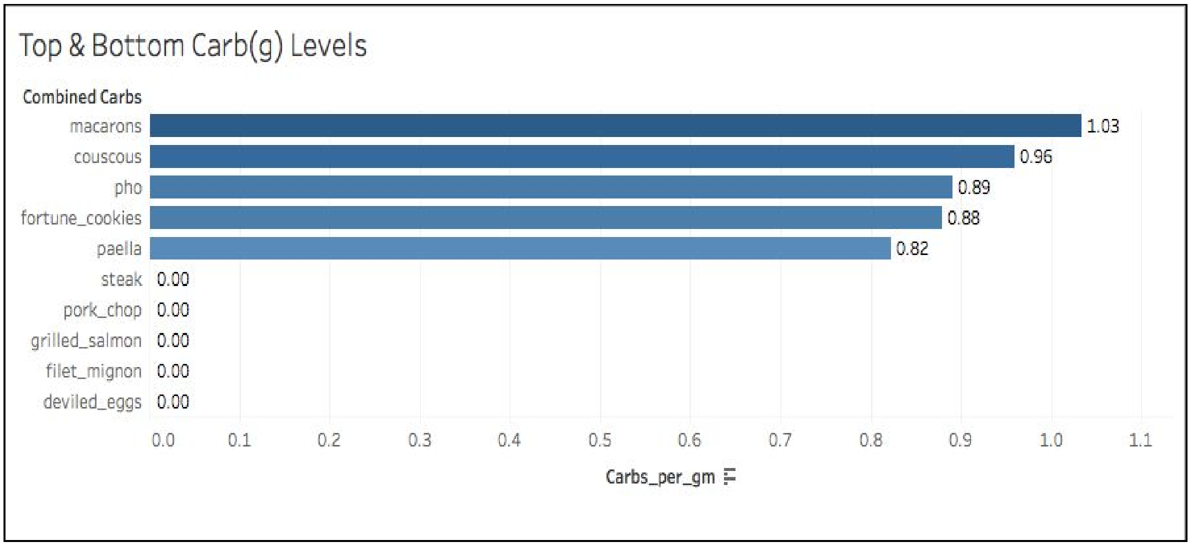
Food Items with highest and lowest Calories

Calorie intake analysis was the most important factor for designing our app and hence it would be extremely helpful to know which food items are loaded with the most and least calorie content. People generally consider calories as the first filter whenever they choose to eat something. We can see that macarons have the highest calories while gazpacho has the lowest calories.



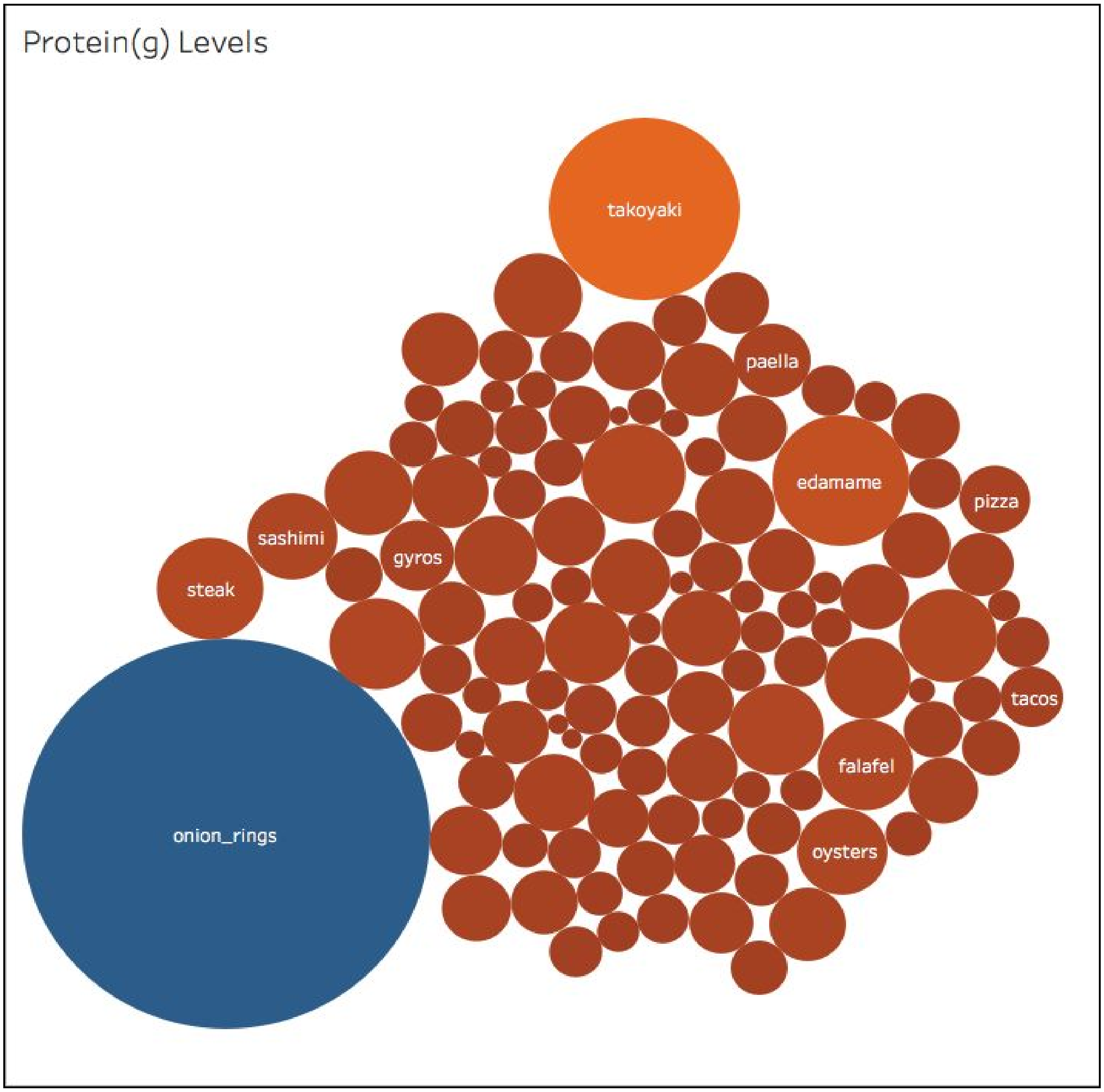
Food Items with highest and lowest Carbohydrates

People suffering from diabetes are often advised to reduce their carbohydrate intake. Eating carbohydrates have the biggest impact on blood sugar and insulin levels. By lowering carbohydrate intake, blood sugars are controlled and insulin levels are minimized. Hence, a diabetic patient may want to know which food items are lower on carbs. We see that macarons and couscous are high on carbs while steak and pork chop are low on carbs.



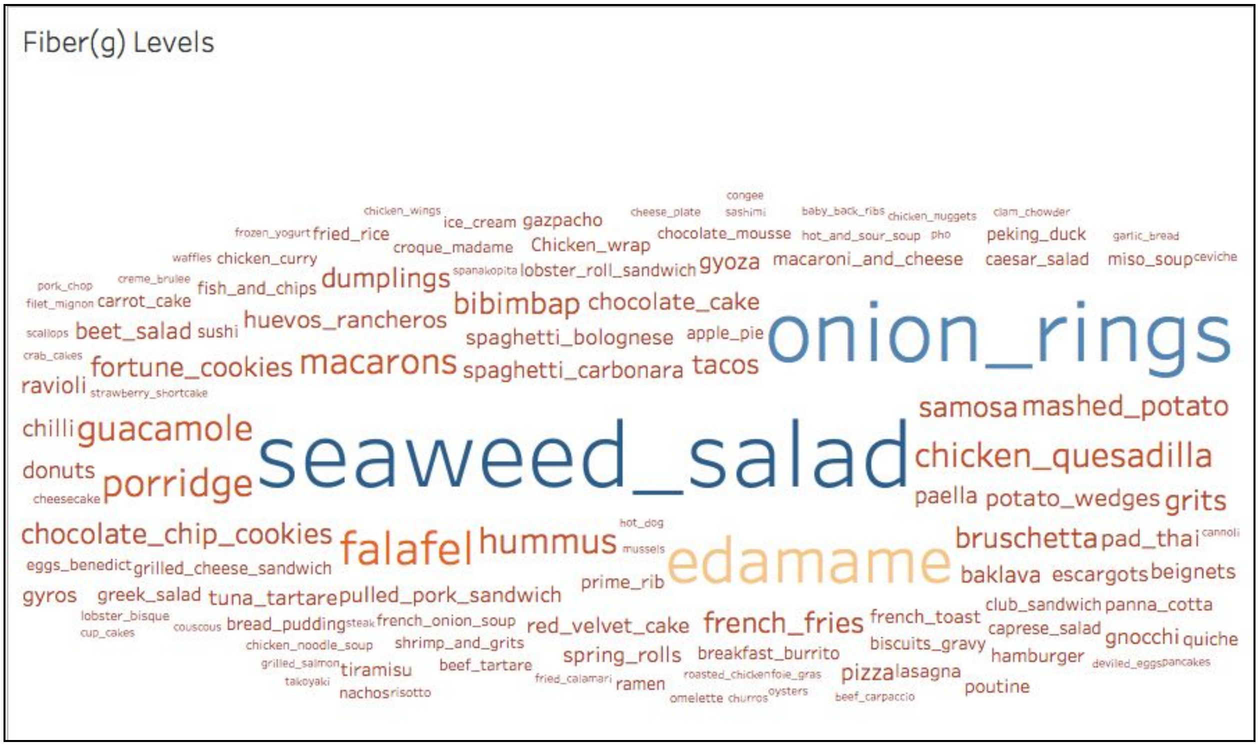
Protein Levels in Food Items

Protein is an essential part of a healthy diet, helps keep muscles strong and can also play a role in metabolism and hunger. Eating a diet high in protein helps burn more calories and hence people who work out regularly prefer to eat food items that are high in proteins. From the visualization below we can see that onion rings have the highest protein content amongst all food items followed by takoyaki while pizza and tacos have lesser protein.



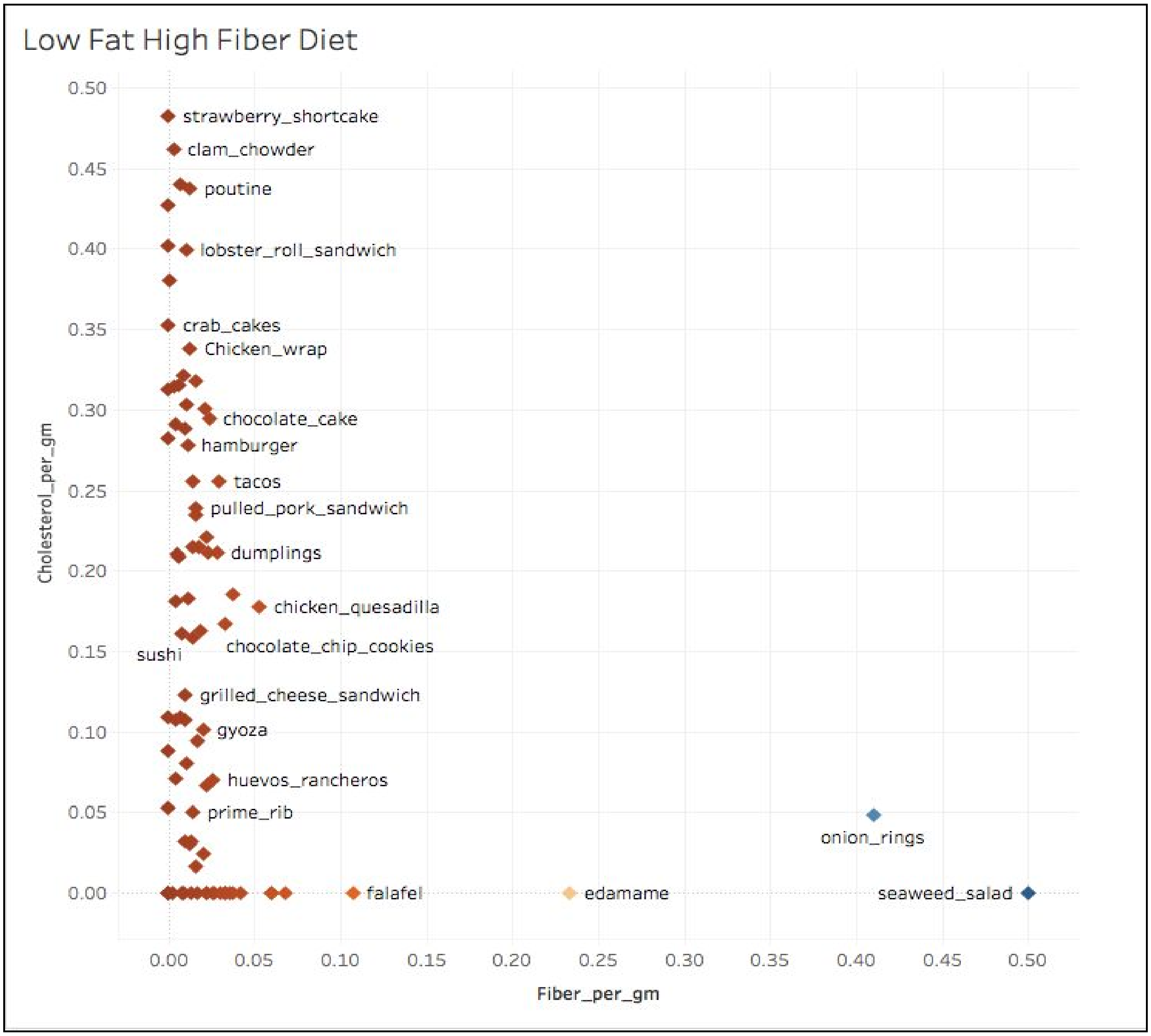
Fiber Levels in Food Items

Foods containing fiber can provide health benefits such as helping to maintain a healthy weight and lowering your risk of diabetes, heart disease. Hence, most people suffering from heart diseases are advised to consume a high fiber diet. Using a word cloud visualization we can see that seaweed salad and onion rings are high on fiber followed by edamame, falafel, and hummus.



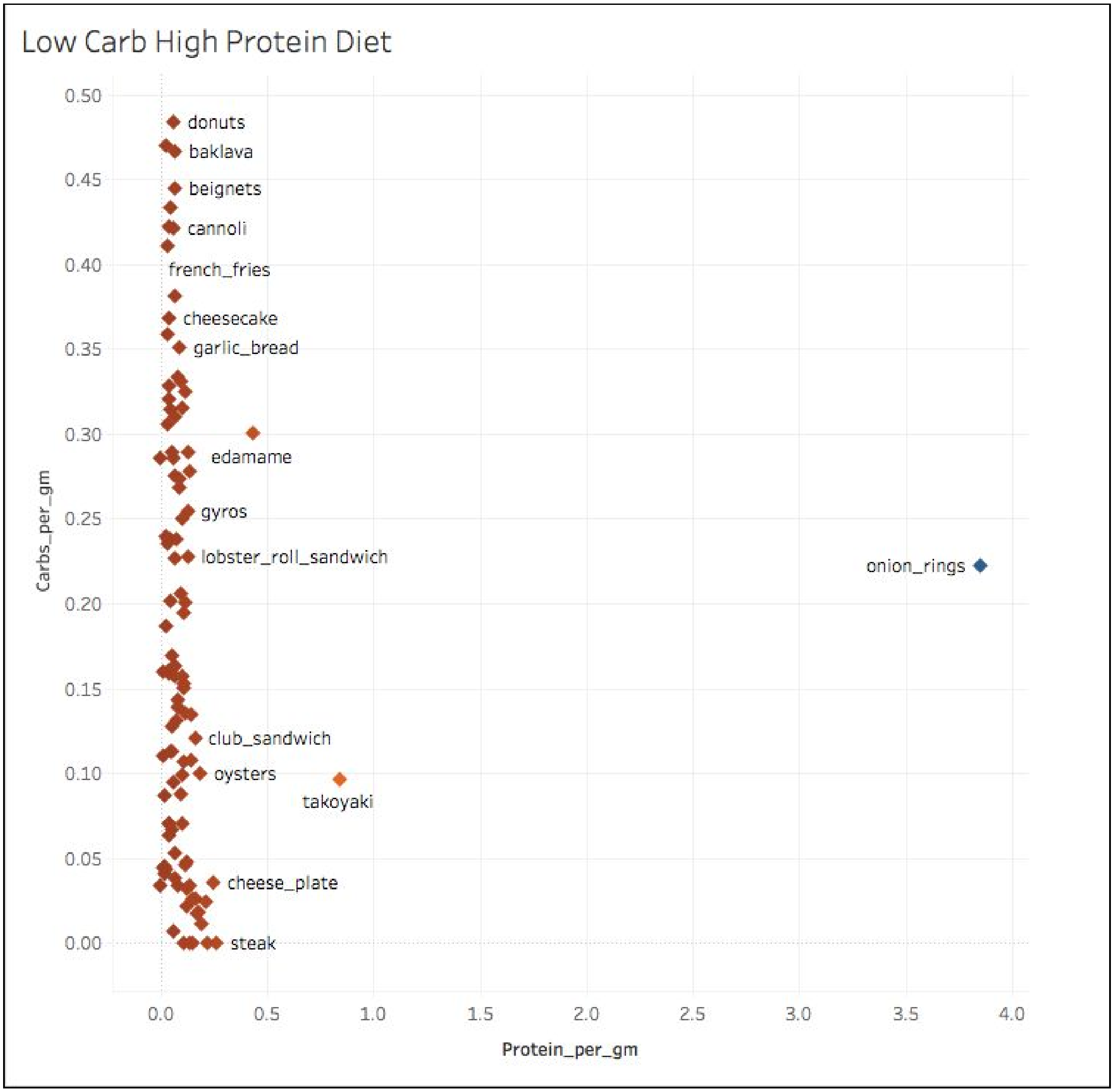
Diet Type: Low Fat High Fiber

Eating a high fiber diet that is low in fat can help maintain overall health. Fiber-rich foods are naturally low in fat. We can look at the visualization below to identify which food items fall under this diet type. Seaweed salad, onion rings, edamame, and falafel can be preferred by people following the low-fat high fiber diet regime.



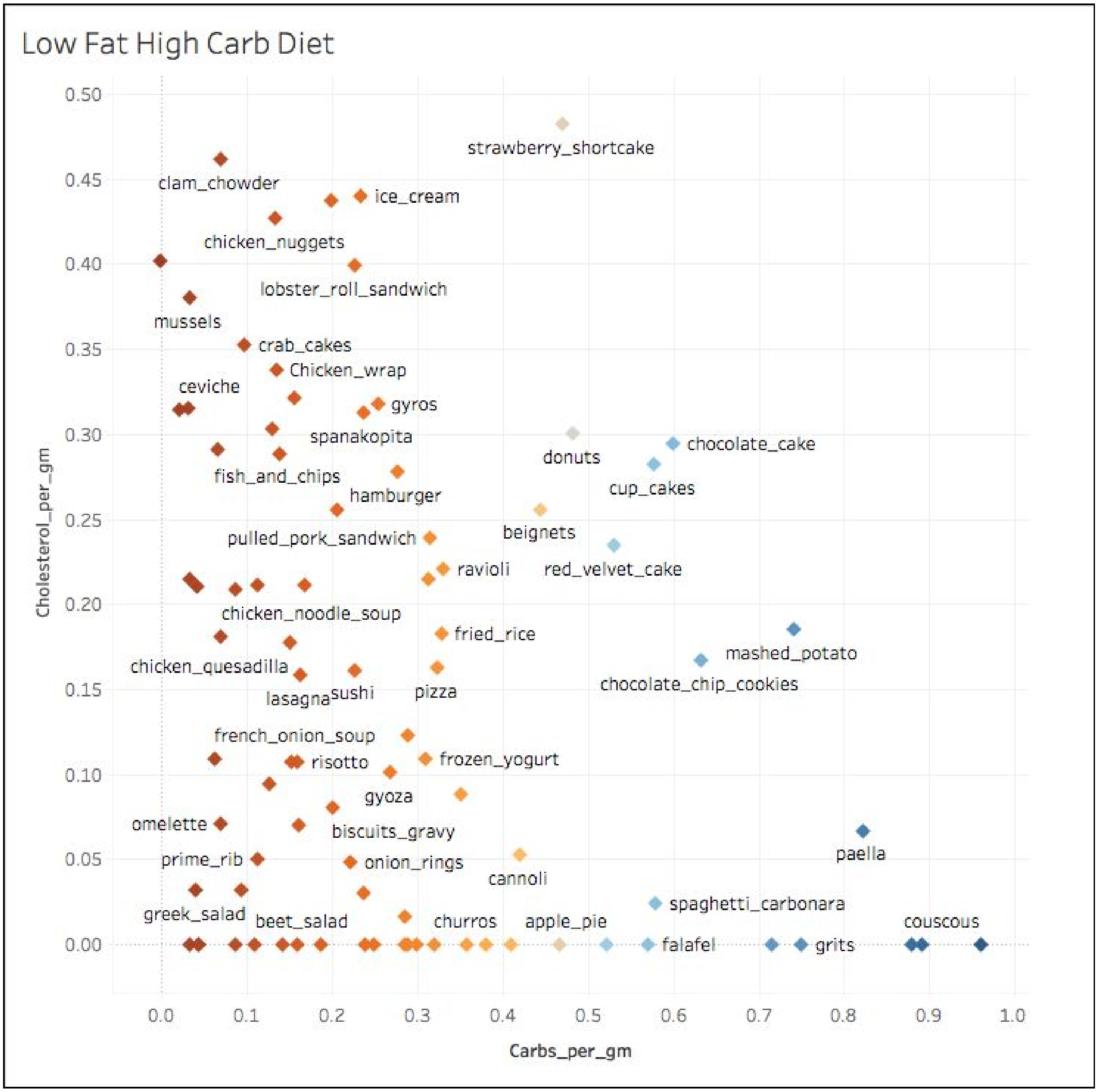
Diet Type: Low Carb High Protein

From the visualization below we can see that there are not many food items that fall into this type of diet, as most of the low carb is also low on protein. Onion ring looks like a good option followed by takoyaki. When we increase our image dataset further, maybe we will find more food items which fall in this diet type.



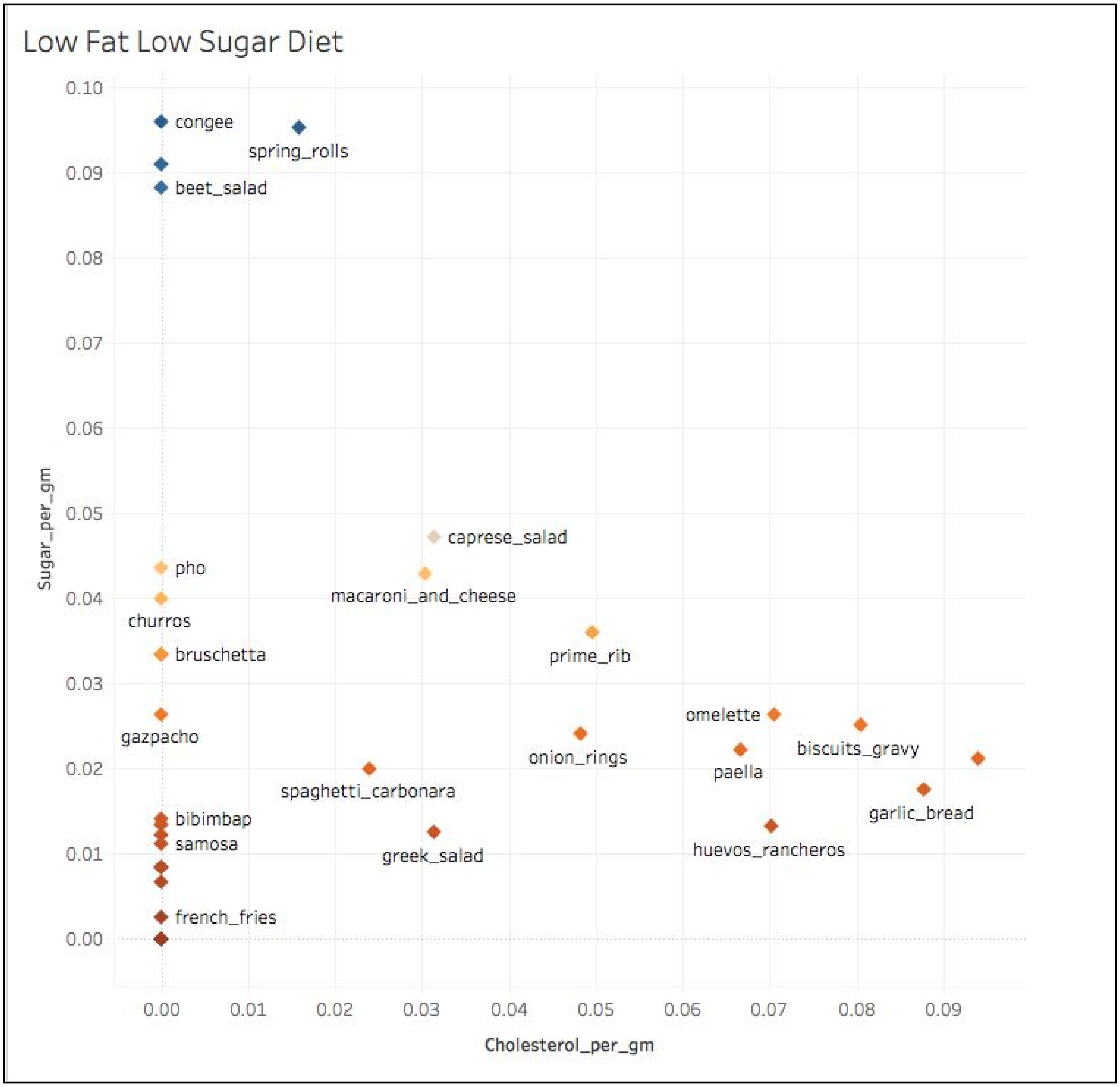
Diet Type: Low Fat High Carb

Eating a high-carbohydrate diet that is low in fat may provide a more beneficial way of burning excess fat. It's entirely possible to lose body fat while eating a high carb low-fat diet as long as more calories are burned than consumed and this diet tends to lead people to eat fewer calories thanks to limited food choices. Hence, some people nowadays follow this type of diet. Couscous, paella, grits, etc are excellent choices for people following the low-fat high fiber diet type.



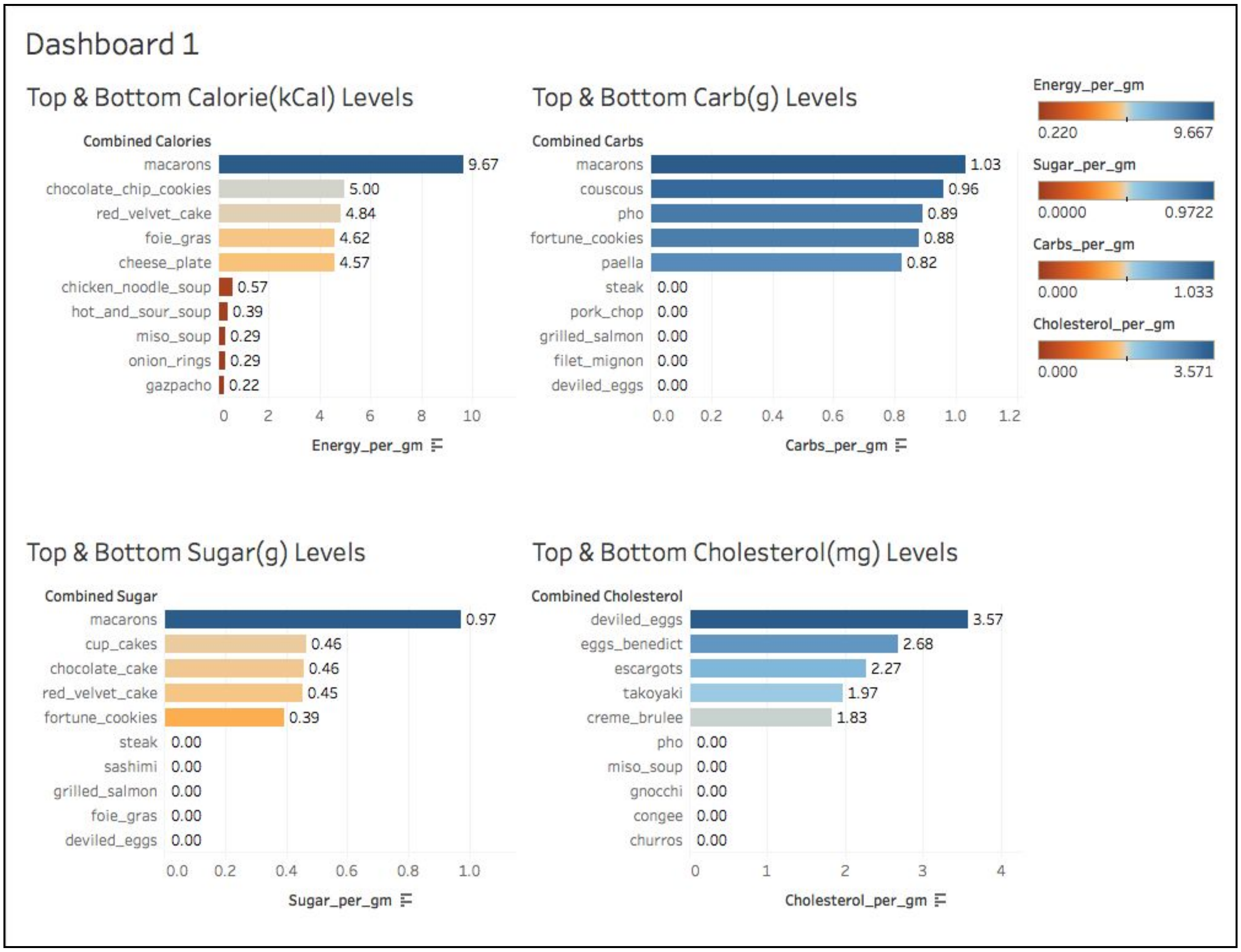
Diet Type: Low Fat Low Sugar

Some people are looking for an overall low-fat low sugar diet with no other specific restrictions on carbs or protein. For such people, we can see from below visualization that french fries, samosa, bibimbap can be good food choices and stay on diet. Greek salad also looks like an acceptable choice.



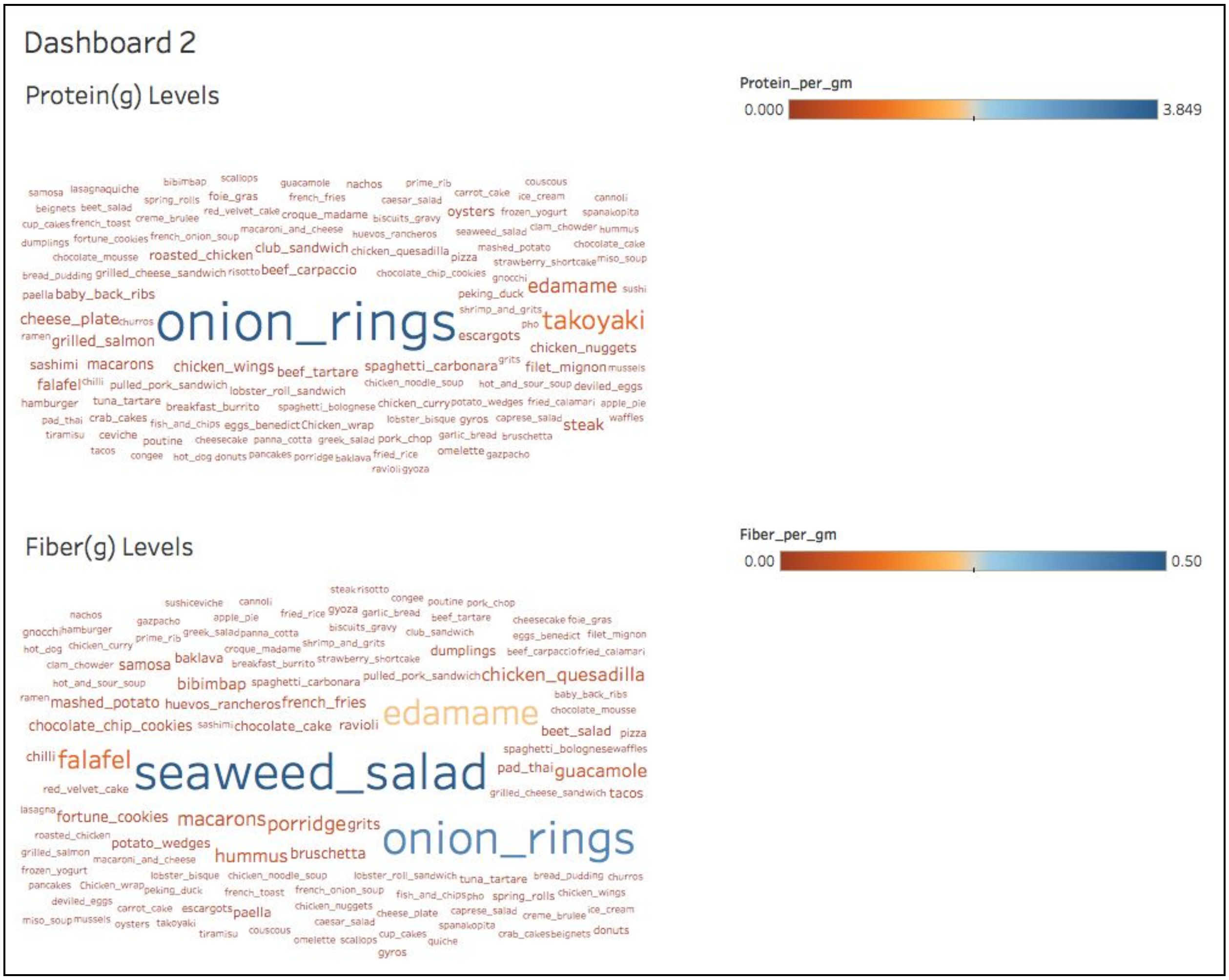
__Dashboard 1 with Highest and Lowest Nutrients__

This dashboard provides at-a-glance views of key indicators - the top and bottom food items for each of the nutrient categories. This is a dynamic visualization and will get updated whenever new data is added to the dataset. Thus this dashboard will always show the top and bottom nutrients based on the currently available data.



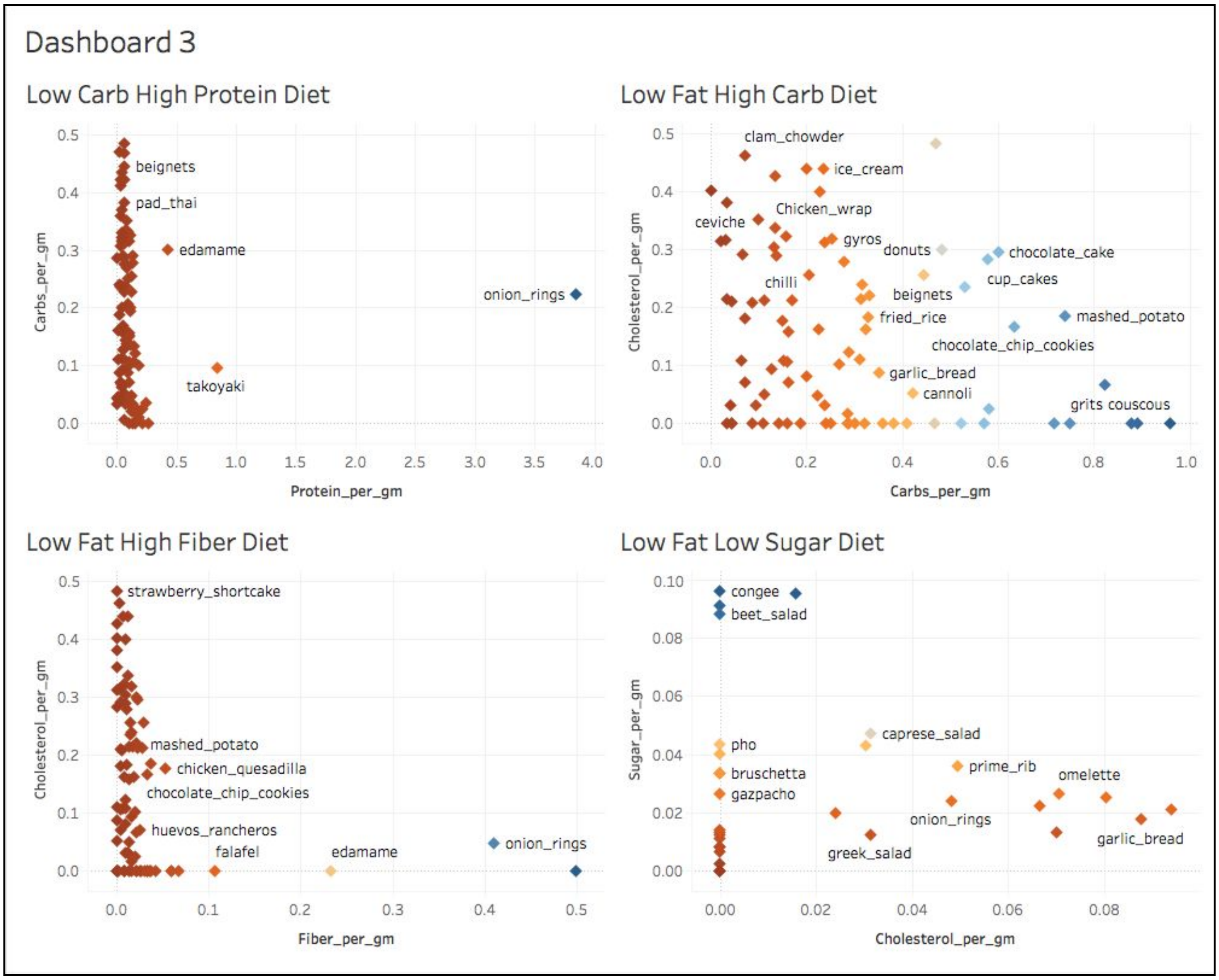
Dashboard 2 with Nutrient Levels

This dashboard provides at-a-glance views of key indicators - food items with the most nutrient content categories for protein and fiber. This is a dynamic visualization and will get updated whenever new data is added to the dataset. Thus this dashboard will always show the top food items based on the currently available data.



Dashboard 3 with Diet Types

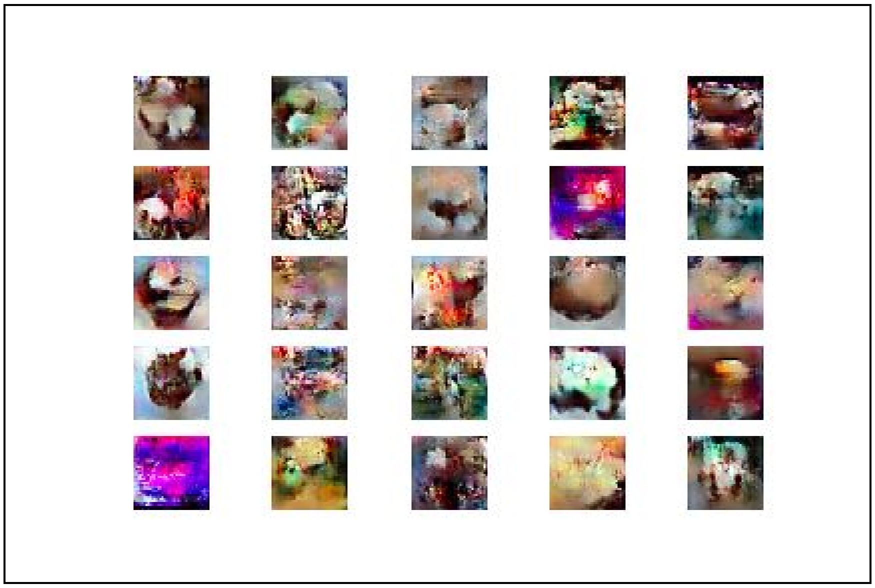
This dashboard provides at-a-glance views of different diet types and associated food items. If a person follows different diets at different times, it will be helpful to look at this dashboard for more information about the food items specific to each diet type. This is a dynamic visualization and will get updated whenever new data is added to the dataset. Thus this dashboard will always show the top food items based on the currently available data.



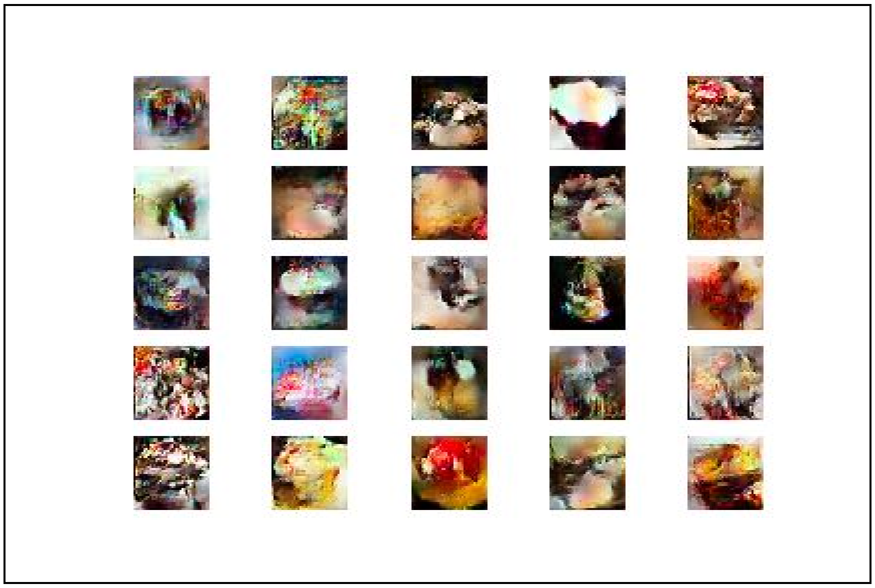
Data Augmentation with GANs

Having a large dataset is crucial for good model performance. As part of our data sourcing process, we have utilized data augmentation using GANs for our own food images to create additional images as GAN's can learn to imitate any distribution of data. With each set of epochs, GAN's creates a number of alike image samples. Looking at the images produced after the first 2000 epochs, you can see that it does not have any real structure, looking at the images after 4000 epochs, the images start to take shape and lastly, the images produced after 10000 epochs are clearer even though a couple are still unrecognizable. Below are the results generated from GANs after 2000 epochs:

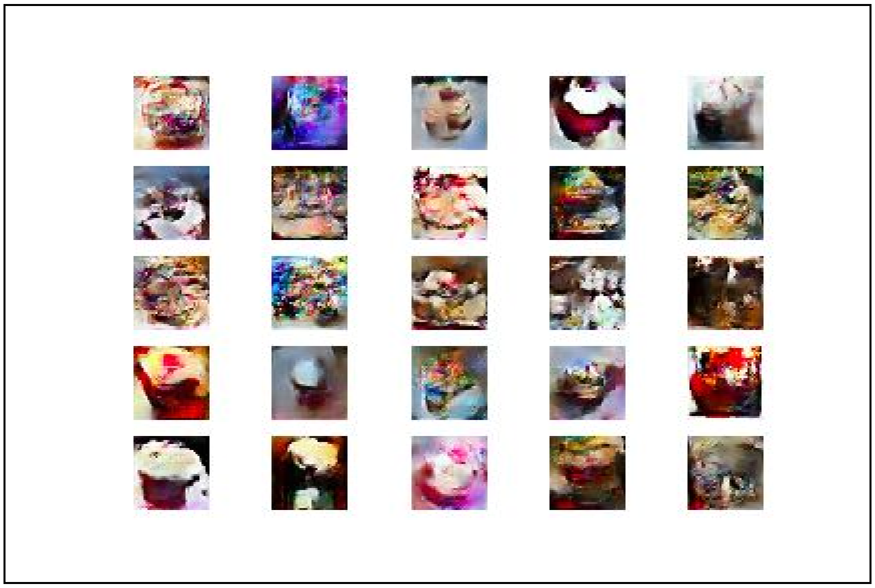
After 4000 epochs:



After 6000 epochs:



After 8000 epochs:



After 10000 epochs:

Thus, we have been able to add new image samples and increased the size of our dataset.

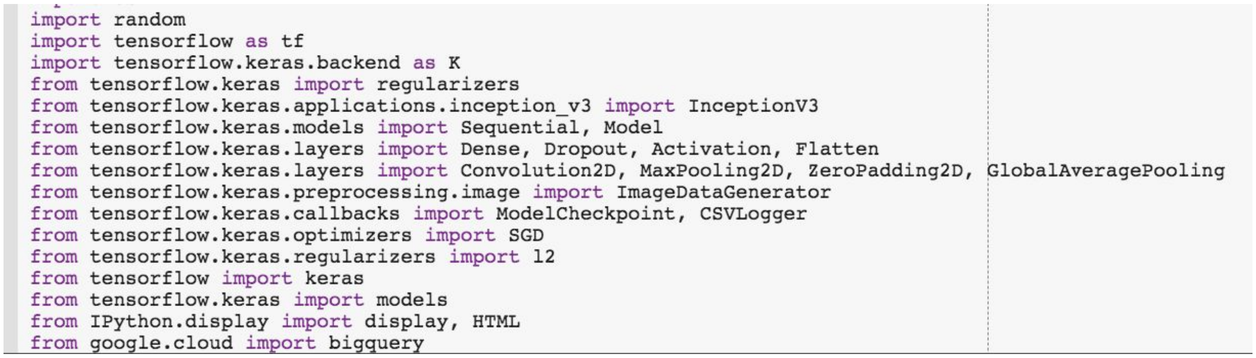
Data Analytics

We used the images saved in our Google Cloud storage and start building our model on Google Compute Engine (VM). Since our dataset is huge, we used 4 GPUs from GCP to run our instance. We used Deep Learning Neural Network based on Tensorflow and Keras for image classification. We will go through the step-by-step approach of what we have done.

Step 1: Importing Libraries and Split dataset

We have imported Keras library and their modules, such as Max Pooling, Sequential, etc. to apply CNN Model. We also imported other Google libraries such as BigQuery for integration purposes.

We have split the Train and test images in 70:30 ratio. Train and test images were copied in separate folders Each directory contained subdirectories with images of different classes.



```
import random
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
from tensorflow.keras import models
from IPython.display import display, HTML
from google.cloud import bigquery
```

***Figure 6**: Libraries and Loaded Packages*

Step 2: Building CNN

We utilized Convolutional Neural Network for our Image classification. CNN is a complex neural network algorithm and multiple hidden layers are used in CNN for image classification. In this product, we have used a pre-trained weights of CNN VGG16 and modify it according to our needs - by adding a new output layer on top of it. VGG16 is a award-winning model and it saves our time to build a model from scratch.

CNN consists of mainly consists of several of following layers

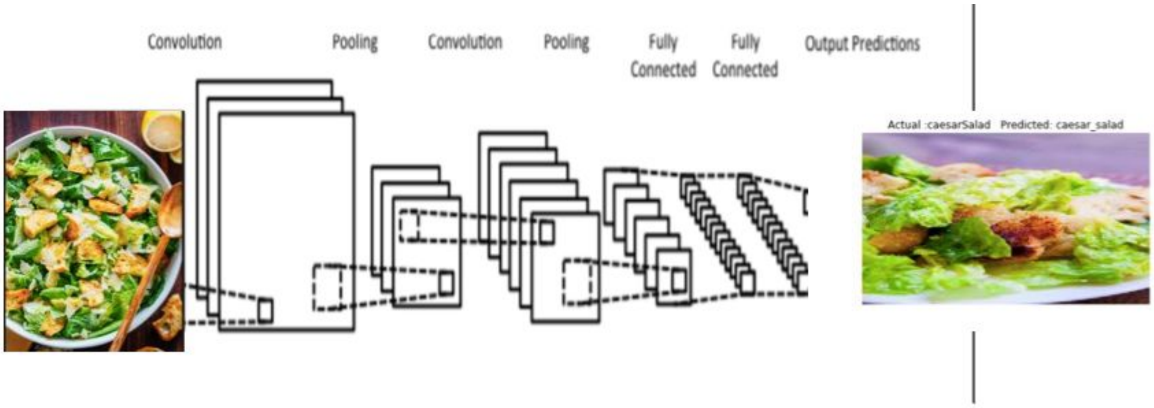


Figure 7: *CNN Basic Architecture*

- **Input layer:** It extracts images and hold raw pixel value of the image. It applies many different filters to it to create a feature map

- **Convolutional layer**: It applies convolution operation to the input and passes the information to next layer. Convolution is a linear operation with things like element wise matrix multiplication and addition

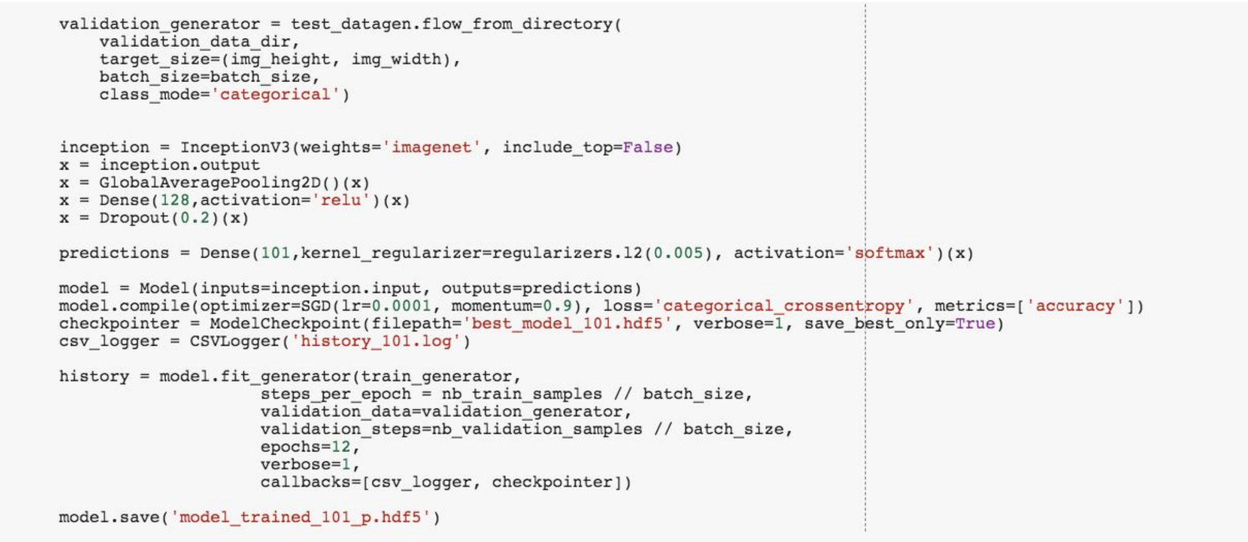
- **Activation Function ReLU**: It is one of the most used activation functions in CNN. Activation function is applied to feature map to increase non linearity in the network. ReLU improves neural network by speeding up the train.

- **Pooling layer:** It applies pooling layer to each feature map. It reduces the number of parameters in our model by a process called down-sampling. It down samples the features like edges. The most used pooling type is called max pooling. It makes it possible to detect objects in an image no matter where they're located. Pooling helps to reduce the number of required parameters and the amount of computation required. It also helps control overfitting.

- **Flatten layer:** This layer will flatten the pooled images into a sequence of one long factor. It inputs the vector into next fully connected layer

- **Fully Connected layer:** The final fully connected layer provides the "voting" of the classes that we're after.

- **Activation Function Softmax**: It calculates probability of every class. In our product we have 120 classes and Softmax assigns the probability of the occurrence of every class. The range of output of Softmax is between 0 and 1 and the sum of all probabilities will be between 0 and 1. We have used it as that last layer in our function.

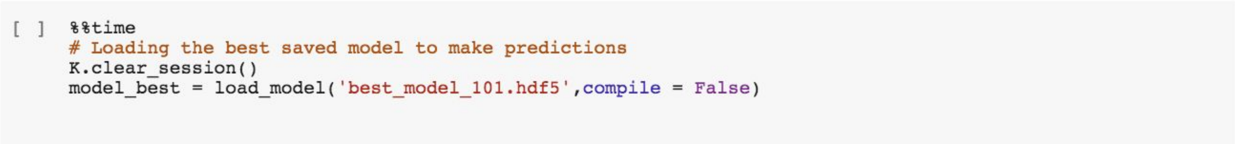


```
validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')


inception = InceptionV3(weights='imagenet', include_top=False)
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(101,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath='best_model_101.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('history_101.log')

history = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=12,
                    verbose=1,
                    callbacks=[csv_logger, checkpointer])

model.save('model_trained_101_p.hdf5')
```

Figure 8: *Actual Model*

Step 3: Training our Model

We trained our model through forward and backward propagation through many epochs. We plugged and played with different hyperparameters like batch size, epochs etc. to find the best accuracy and Loss function. In general, with the increase of epochs, accuracy increased. Training our model on almost 120,000 images took a very long time almost 14-15 hours. Once our model is trained, we will save our model for further prediction on new images.



```
[ ]  %%time
     # Loading the best saved model to make predictions
     K.clear_session()
     model_best = load_model('best_model_101.hdf5',compile = False)
```

Figure 9: *Loading Model*

Step 4: Model Evaluation

After 12 epochs our model accuracy is 74.29% and Loss function is 1.33. We are under the understanding that as we increase our epochs, the model accuracy will be increased. We tried to use other pre trained weights like RESTNET but Inception VGG16 is much faster than others so we went ahead with Inception VGG16.

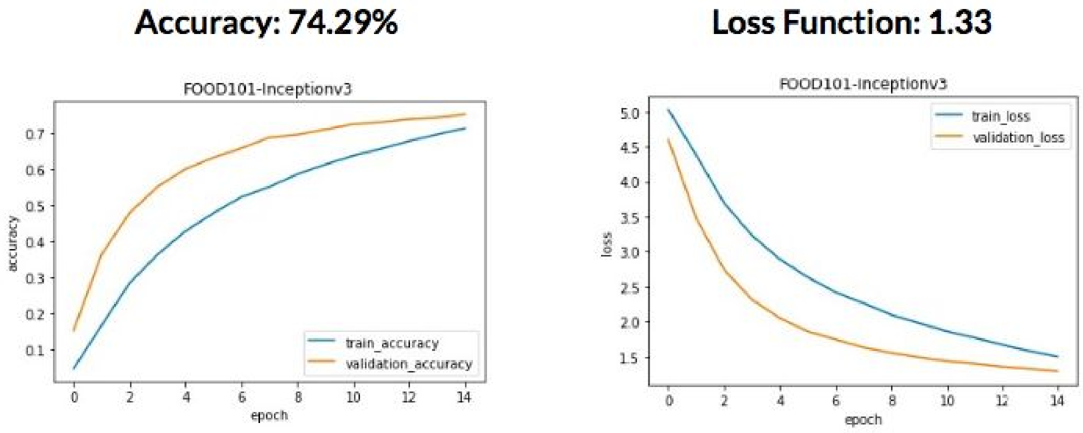
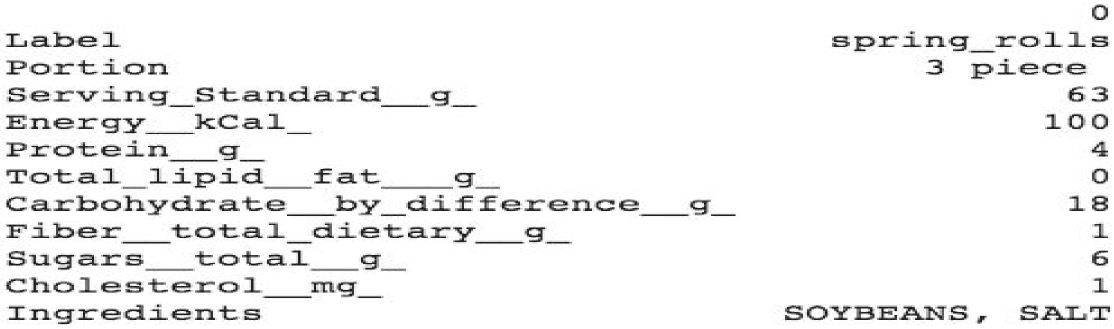


Figure 10: *Accuracy and Loss Function Graph*

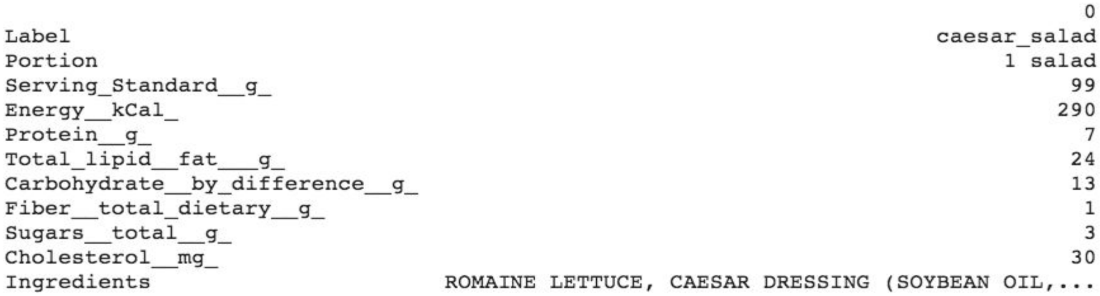
Step 5: Testing

We have downloaded few images from different classed from Google images url. We load the model saved in last step and test on new images. Our model almost predicted 10 images out of 10 correctly. For every image our model showed the actual food item and what our model has predicted, followed by the nutrient information which is fetched based on the predicted value.

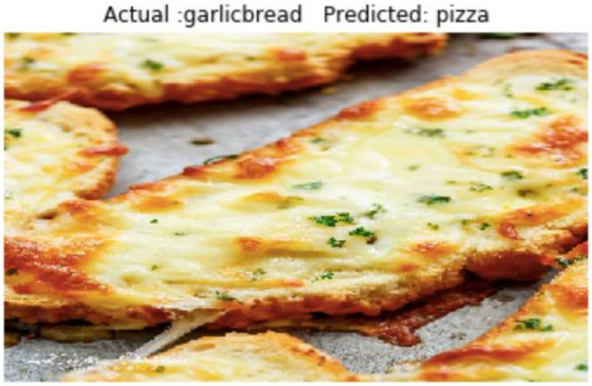
Actual :springrolls Predicted: spring_rolls

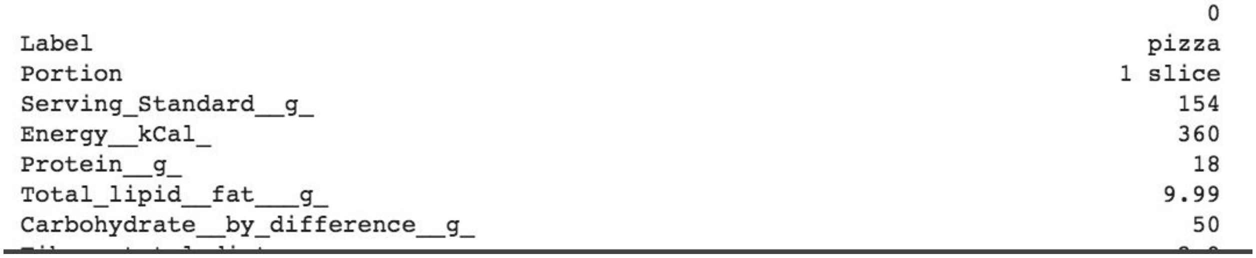

```
                                                    0
Label                                     spring_rolls
Portion                                       3 piece
Serving_Standard__g_                               63
Energy__kCal_                                     100
Protein__g_                                         4
Total_lipid__fat___g_                               0
Carbohydrate__by_difference__g_                    18
Fiber__total_dietary__g_                            1
Sugars__total__g_                                   6
Cholesterol__mg_                                    1
Ingredients                            SOYBEANS, SALT
```

Actual :caesarSalad Predicted: caesar_salad


```
                                                                                  0
Label                                                                   caesar_salad
Portion                                                                      1 salad
Serving_Standard__g_                                                              99
Energy__kCal_                                                                    290
Protein__g_                                                                        7
Total_lipid__fat___g_                                                             24
Carbohydrate__by_difference__g_                                                   13
Fiber__total_dietary__g_                                                           1
Sugars__total__g_                                                                  3
Cholesterol__mg_                                                                  30
Ingredients                    ROMAINE LETTUCE, CAESAR DRESSING (SOYBEAN OIL,...
```

We realized that since we have around 74% accuracy, our model still would be able to predict incorrectly. For instance, our model has predicted garlic bread as pizza. However, we could argue with this because even our eyes might say this is a thin-layer pizza.





```
                                                    0
Label                                           pizza
Portion                                       1 slice
Serving_Standard__g_                              154
Energy__kCal_                                     360
Protein__g_                                        18
Total_lipid__fat___g_                            9.99
Carbohydrate__by_difference__g_                    50
```

Despite the few limitation our model has, we were able to justify as of why this happened. Although, we realize that we can improve our model by training it with more dataset.

Challenges

- The accuracy of the CNN model depends heavily on the amount of training data fed into it. As the size of training data was high, the training process became increasingly more resource hungry. This calls for high performance GPUs and larger RAMs capacities for training multi-layer NNs involving large training data sets.

- Image classification model was taking more time to train before it starts making meaningful predictions.Initially we started with 10 different food item dataset to have clarity about approx time it will take with full dataset and also to decide number of epochs we want to run to achieve acceptable accuracy.

- Data Augmentation using DCGAN to generate more images for New food item heavily depends on compute power and time to train the model. we started to 4000 epochs and tried till 10000 epochs with multiple hyperparameter tuning to generate clear and close proximity images.

Future Enhancements

- Create a MentorDiet mobile App
- Add more images to meet the requirements of a larger audience
- Add Voice Feature to speak out the image label and nutrient information

References

[1] Functional Foods: Key Trends & Developments in Ingredients. (2014, November 21). Retrieved March 29, 2019, from https://www.packagedfacts.com/Functional-Foods-Key-8540424/

[2] DanB, “Food 101,” Kaggle, 03-Jan-2018. [Online]. Available: https://www.kaggle.com/dansbecker/food-101/home.

[3] “Welcome to the USDA Food Composition Databases,” USDA Food Composition Databases. [Online]. Available: https://ndb.nal.usda.gov/ndb.

[4] “Data Catalog,” *Data.gov*. [Online]. Available: https://catalog.data.gov/dataset?tags=calories.

[5] Nutritionix.com. (2019). {{MetaTags.title || 'Nutritionix'}}. [online] Available at: https://www.nutritionix.com/database

[6] Introduction to loading data into BigQuery | BigQuery | Google Cloud. (n.d.). Retrieved April 23, 2019, from https://cloud.google.com/bigquery/docs/loading-data

[7] Choosing a Storage Option | Google Cloud. (n.d.). Retrieved April 23, 2019, from https://cloud.google.com/storage-options/

[8] Storage Options | Compute Engine Documentation | Google Cloud. (n.d.). Retrieved April 23, 2019, from https://cloud.google.com/compute/docs/disks/

[9] Stackdriver - Hybrid Monitoring | Stackdriver | Google Cloud. (n.d.). Retrieved April 23, 2019, from https://cloud.google.com/stackdriver/

[10] Stackdriver Monitoring documentation | Stackdriver Monitoring | Google Cloud. (n.d.). Retrieved April 23, 2019, from https://cloud.google.com/monitoring/docs/