# Final Project - EDA

Etay Nahum - 313163735

```
library(tidymodels)
library(tidyverse)
library(ggplot2)
library(tidytext)
library(rpart)
library(glmnet)
library(reticulate)
library(xgboost)
library(kknn)
library(randomForest)
library(purrr)
```

```
import pandas as pd
import numpy as np
import os
from matplotlib.image import imread
import warnings
```

## Reading and Uniting the Data

```
food_train <- read_csv("food_train.csv")
food_test <- read_csv("food_test.csv")
food_nutrients <- read_csv("food_nutrients.csv")
nutrients <- read_csv("nutrients.csv")
```

- I united the food nutrients and the nutrients datasets, changed the dataset to wide so each nutrient have it's own column so i can unite it with the doof train dataset.
- Unite the train and test datasets with the final nutrients datasets by idx.
- Saved the columns in "cols_with_NA" that have over 80% NA , before changing those NA into 0 in the test and train sets (the assumption is that a nutrients with NA suggesting that there isn't this nutrient in the snack).
- Changing ml to g (there are that same)

```
full_nut_data <- full_join(food_nutrients , nutrients) %>%
  select(idx , amount , name)

full_nut_data <- full_nut_data %>%
  pivot_wider(id_cols = 1 ,
              values_from = amount , names_from = name , values_fn = mean)

index_for_train <- food_train %>%
```

```
  pull(idx)
index_for_test <- food_test %>%
  pull(idx)

final_train <- full_join(food_train , full_nut_data[index_for_train ,])
final_test <- full_join(food_test , full_nut_data[index_for_test ,])

cols_with_NA <- final_train[,((colSums(is.na(final_train)))/nrow(final_train)) > 0.8]

final_train <- final_train %>%
  mutate(across(is.numeric , ~replace_na(.x , 0))) %>%
  mutate(serving_size_unit = ifelse(serving_size_unit == "ml" , "g" , "g"))

final_test <- final_test %>%
  mutate(across(is.numeric , ~replace_na(.x , 0))) %>%
  mutate(serving_size_unit = ifelse(serving_size_unit == "ml" , "g" , "g"))
```

## NA Features

- Checking if the columns with NA have a pattern/significance in each category. for this i calculated for each category the mean percentage that those columns have a positive value and not zero.

```
## # A tibble: 6 x 2
##   category                                perc
##   <chr>                                  <dbl>
## 1 chips_pretzels_snacks                0.00660
## 2 popcorn_peanuts_seeds_related_snacks 0.00480
## 3 cookies_biscuits                     0.00400
## 4 cakes_cupcakes_snack_cakes           0.00311
## 5 chocolate                            0.00102
## 6 candy                                0.000855
```

- As we can see for all the categories we have very small percentage of positives values in those columns,and we don't have a very big difference between the categories, thus we will not use them in the prediction section.

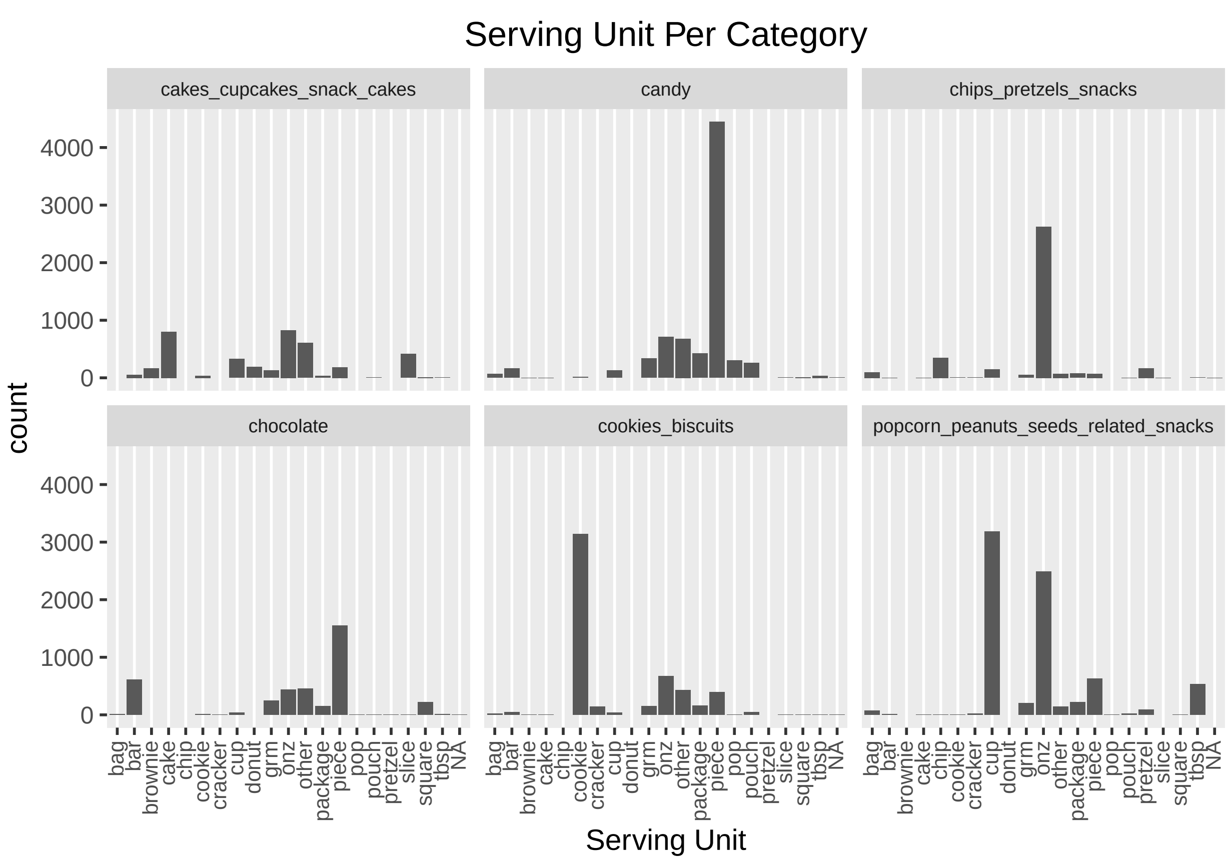
## EDA on Nominal Features

### household_serving_fulltext Feature

- Checking difference in the serving unit for each category with the household_serving_fulltext feature.
- After studying this feature, checking which unit shows up most, and checking errors in the spelling and other variations of spelling for the same unit.

```
strings <- c("onz","cookie","piece","slice","chip" , "cracker" , "cup" ,
             "cake" , "pretzel" ,"pop" ,"square" , "bag","pouch", "package" ,
            "bar" , "brownie" ,  "tbsp" , "donut" ,"piece" , "grm" )

serving_text <- final_train %>%
  select(household_serving_fulltext , category) %>%
  mutate(text_units = gsub('[[:digit:]]+' , '' , household_serving_fulltext)) %>%
```

```r
mutate(text_units = gsub('\\.+' , '' , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"onz") , "onz" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units
                                      ,"cookie|cookeis|cookes|ckooies|cooikes|coookie"),
                           "cookie" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"piece|pcs") , "piece" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"slice") , "slice" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"chip") , "chip" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"cracker") , "cracker" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"cup") , "cup" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"cake") , "cake" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"pretzel|petzels") , "pretzel" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"pop") , "pop" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"square") , "square" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"bag") , "bag" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"pouch") , "pouch" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"package|pkg|pack") , "package" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"bar") , "bar" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"brownie") , "brownie" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"tbsp") , "tbsp" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"donut") , "donut" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units ,"piece") , "piece" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units , "grm") , "grm" , text_units)) %>%
mutate(text_units = ifelse(str_detect(text_units , paste(strings , collapse = "|")) ,
                                      text_units , "other"))
```

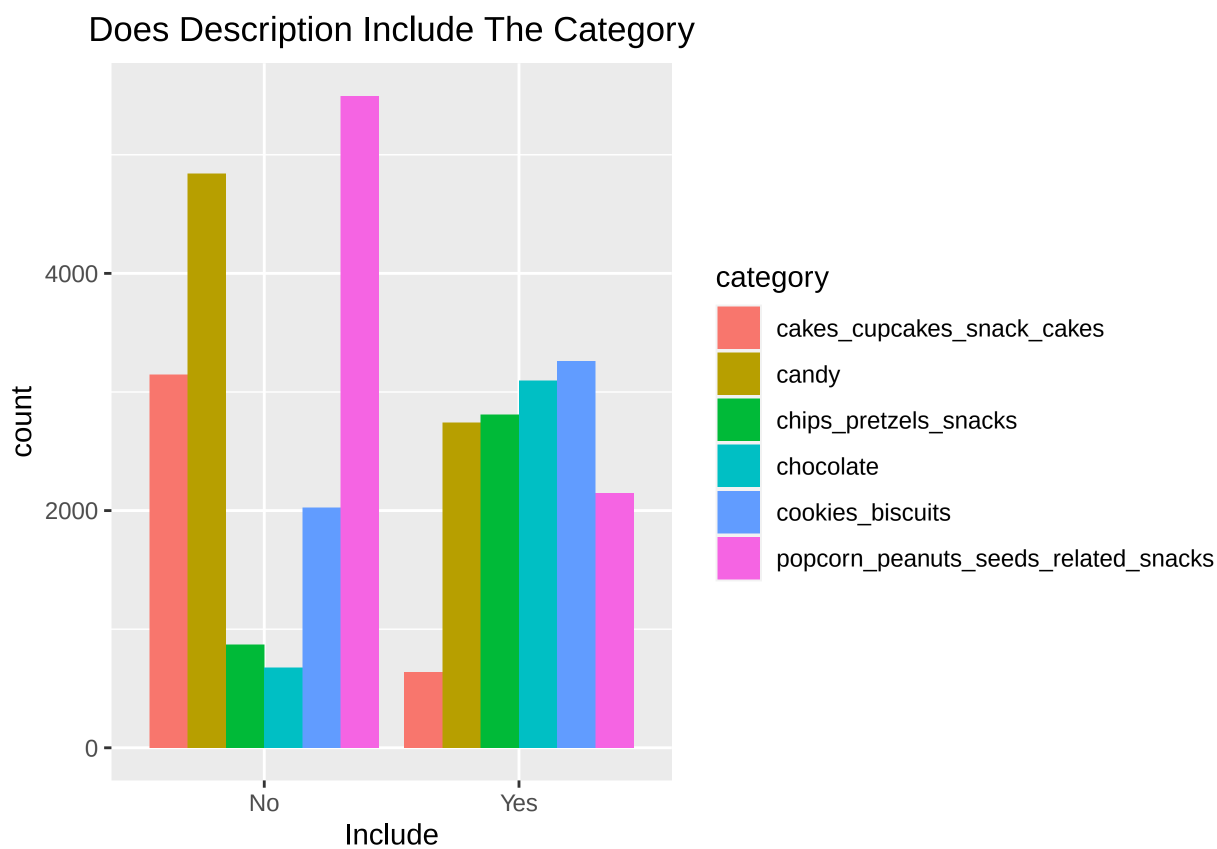


- We can see that cakes_cupcakes_snack_cakes category is pretty diverse, chocolate is mainly "piece" and "bar" , candy is mostly "piece" , cookies_biscuits is mostly "cookie" , chips_pretzels_snacks is mostly "onz", and popcorn_peanuts_seeds_related_snacks is mainly "cup" and "onz". This information will help us in the prediction section.

### Description Feature

- Checking if the description contains the name of the category.

```
desc_func <- function(cate){
  pat <- gsub('\\_+' , '|' , cate)
  new <- final_train %>%
    filter(category == cate) %>%
    select(category , description) %>%
    mutate(include = ifelse(str_detect(description ,pat) , "Yes" , "No")) %>%
    select(include , category)
  return(new)
}

cate_vec <- final_train %>%
  select(category) %>%
  unique() %>%
  pull()

desc_data <- map_dfr(cate_vec , desc_func)
```



- We can see that for chips_pretzels_snacks, chocolate and cookies_biscuits most of the descriptions does contain the name of the category, and for the other three most of them are not. This will further help us in the prediction section.

### Ingredients Feature

- Checking for differences in ingredients for each category by taking the top 15 words in ingredients for each category.

```
func_for_ing <- function(cate){
  top_word <- final_train %>%
  filter(category == cate) %>%
  select(ingredients) %>%
  unnest_tokens(word , ingredients) %>%
  count(word) %>%
  filter(str_detect(word , "[a-z]")) %>%
  arrange(-n) %>%
  slice_head(n = 15) %>%
  add_column(category = rep(cate , 15))
  return(top_word)
}
```

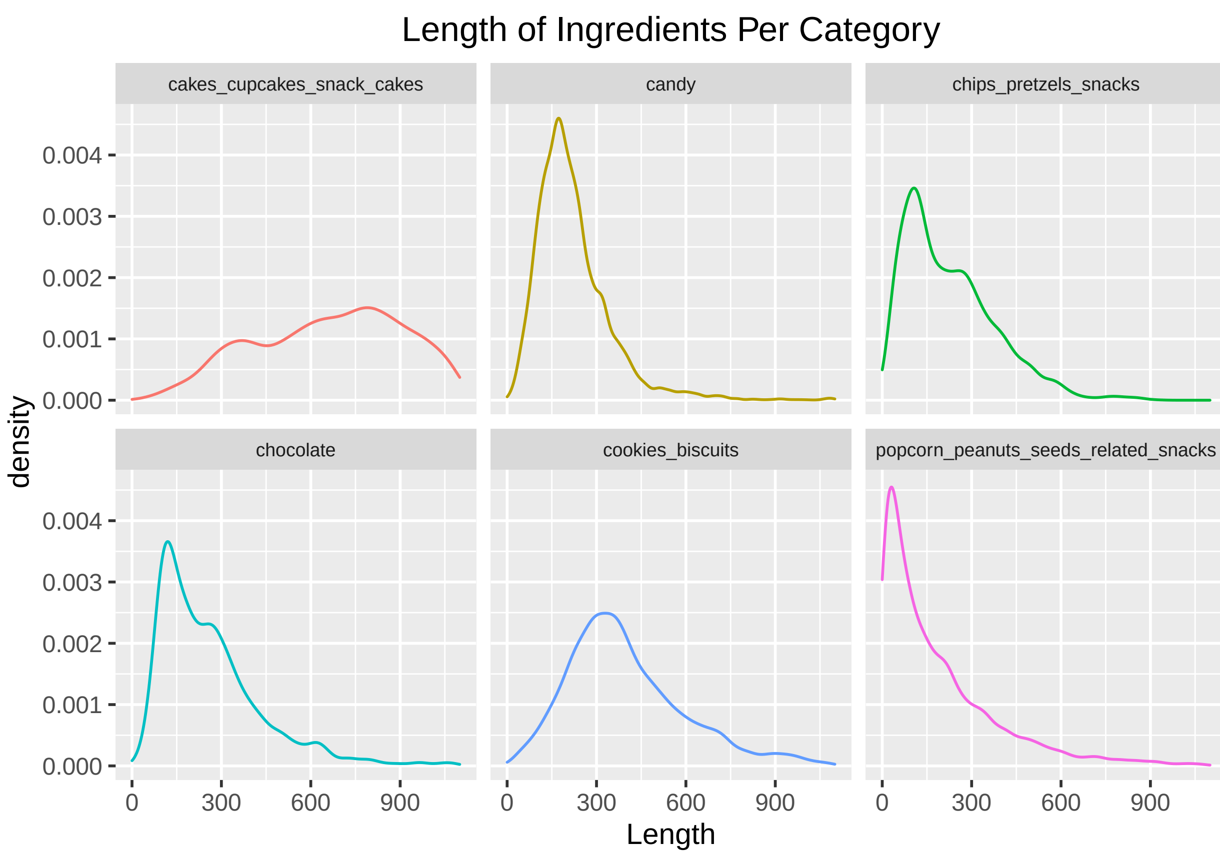
```
ing_df <- map_dfr(cate_vec , func_for_ing)
final_ing_df <- tibble("chocolate" = ing_df[1:15,1] %>% pull() ,
                       "cookies_biscuits" = ing_df[16:30,1] %>% pull() ,
                       "cakes_cupcakes_snack_cakes" = ing_df[31:45,1]%>% pull() ,
                       "candy" = ing_df[46:60 , 1]%>% pull() ,
                       "chips_pretzels_snacks" = ing_df[61:75,1]%>% pull(),
                       "popcorn_peanuts_seeds_related_snacks" = ing_df[76:90,1]%>% pull())

final_ing_df
```

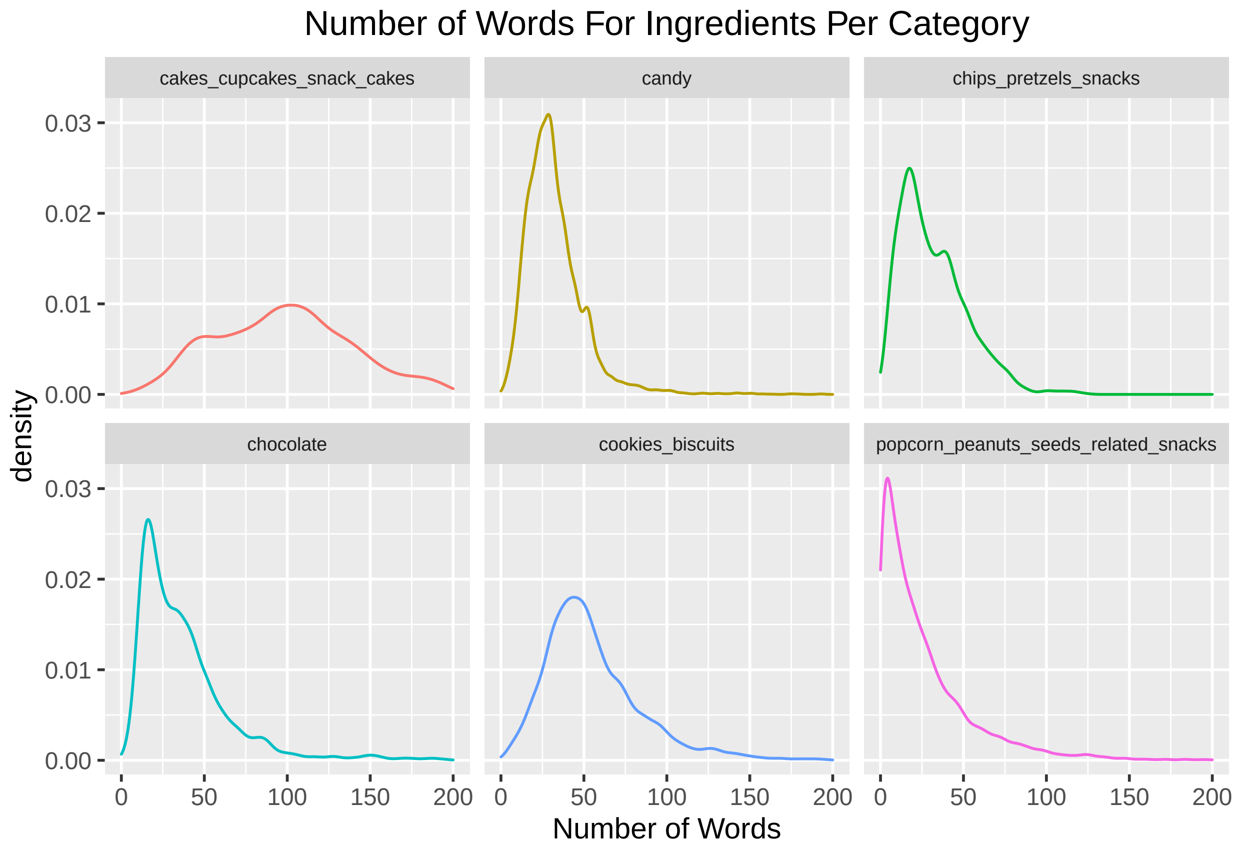
```
## # A tibble: 15 x 6
##    chocolate  cookies_biscuits cakes_cupcakes_snack_cakes candy chips_pretzels_~
##    <chr>      <chr>            <chr>                      <chr> <chr>
##  1 milk       flour            and                        sugar oil
##  2 sugar      sugar            oil                        arti~ salt
##  3 cocoa      oil              flour                      corn  corn
##  4 chocolate  salt             acid                       acid  or
##  5 butter     and              sodium                     syrup powder
##  6 lecithin   wheat            sugar                      yell~ and
##  7 soy        palm             corn                       red   sunflower
##  8 emulsifier acid             wheat                      and   acid
##  9 oil        lecithin         salt                       oil   flour
## 10 vanilla    soy              gum                        blue  of
## 11 natural    corn             palm                       natu~ canola
## 12 salt       natural          milk                       flav~ organic
## 13 and        flavor           starch                     citr~ natural
## 14 flavor     artificial       soy                        milk  sugar
## 15 powder     syrup            water                      flav~ vegetable
## # ... with 1 more variable: popcorn_peanuts_seeds_related_snacks <chr>
```

- Unique (mostly) ingredients:

- chocolate - milk , cocoa , butter , chocolate , emulsifier , vanilla
- cookies_biscuits - wheat , palm , syrup
- cakes_cupcakes_snack_cakes - sodium , gum , starch , water
- candy - yellow , red , blue , citric
- chips_pretzels_snacks - sunflower , canola , organic , vegetable
- popcorn_peanuts_seeds_related_snacks - almonds , sunflower
- Checking the amount of the ingredients of each category by calculating the length and the number of words for each category.

```
length_ing <- final_train %>%
  select(ingredients , category) %>%
  mutate(len_ing = str_length(ingredients)) %>%
  mutate(num_word = str_count(ingredients , "\\w+"))
```






- For both metrics we can see that cakes_cupcakes_snack_cakes category is larger than the rest, followed by cookies_biscuits category, and have a distribution that resembles the normal distribution.
- For the other four, we can see right tail , suggesting we should apply log transformation for the two metrics.

### brand feature

- Checking the top 10 brands and adn filtering only with categories that those brand shows up more than 100 times.

```
brand_data <- final_train %>%
  group_by(category , brand) %>%
  count(brand) %>%
  arrange(-n)

brand_vec <- brand_data %>%
  select(brand) %>%
  pull()
```

```
## Adding missing grouping variables: `category`
```

```
brand_data <- brand_data %>%
  filter(brand %in% brand_vec[1:10]) %>%
  filter(n > 100)
```

```r
brand_data
```

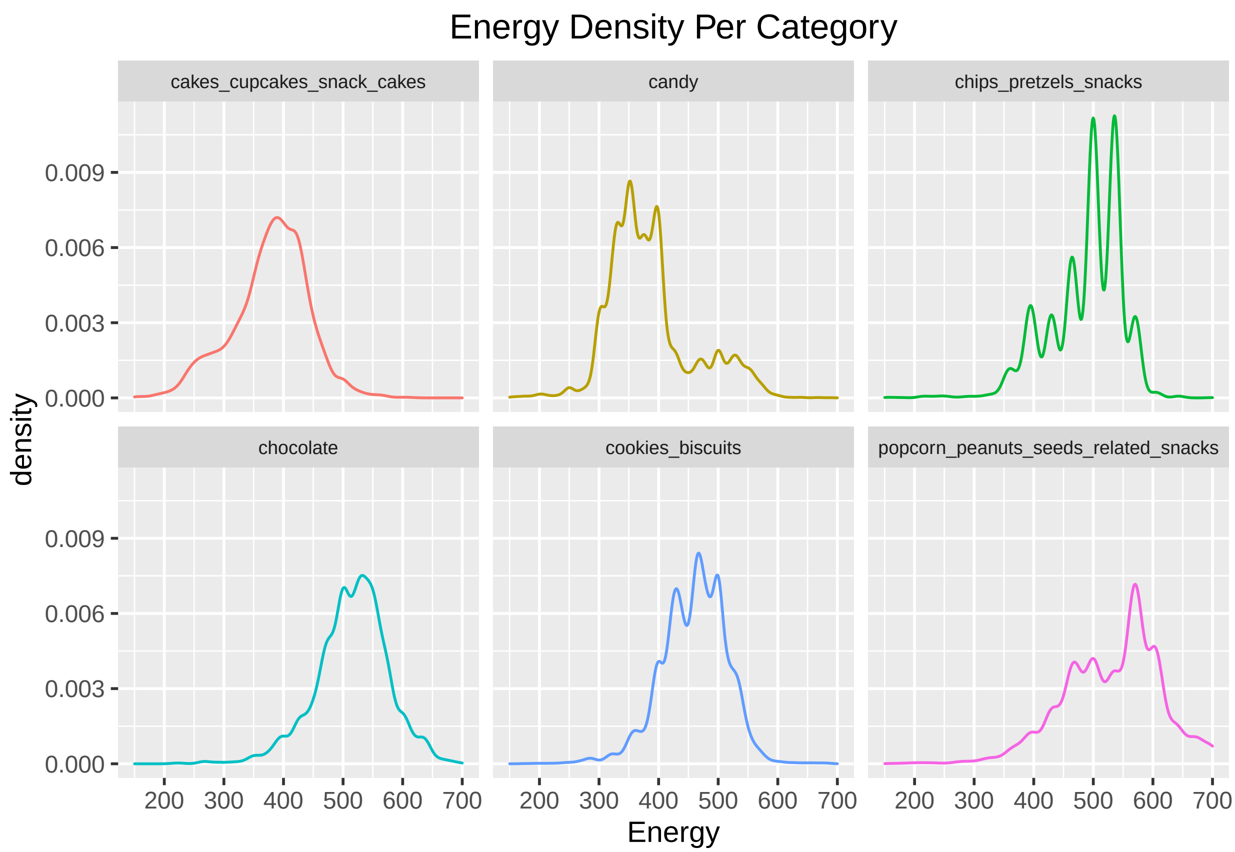
```
## # A tibble: 16 x 3
## # Groups:   category, brand [16]
##    category                             brand                            n
##    <chr>                                <chr>                        <int>
##  1 candy                                ferrara candy company          475
##  2 cakes_cupcakes_snack_cakes           wal-mart stores, inc.          235
##  3 popcorn_peanuts_seeds_related_snacks meijer, inc.                   208
##  4 popcorn_peanuts_seeds_related_snacks target stores                  185
##  5 chocolate                            lindt & sprungli (schweiz) ag   166
##  6 chocolate                            russell stover candies inc.    149
##  7 chips_pretzels_snacks                utz quality foods, inc.        146
##  8 candy                                frankford candy, llc           144
##  9 candy                                not a branded item             141
## 10 cookies_biscuits                     nabisco biscuit company        140
## 11 cookies_biscuits                     wal-mart stores, inc.          139
## 12 cakes_cupcakes_snack_cakes           not a branded item             136
## 13 cakes_cupcakes_snack_cakes           target stores                  125
## 14 cookies_biscuits                     target stores                  114
## 15 popcorn_peanuts_seeds_related_snacks not a branded item             112
## 16 candy                                russell stover candies inc.    110
```

- We can see that there are brands that shows up more for specific category, like the ferrara candy company for candy category and meijer for popcorn.

## EDA on Numeric features

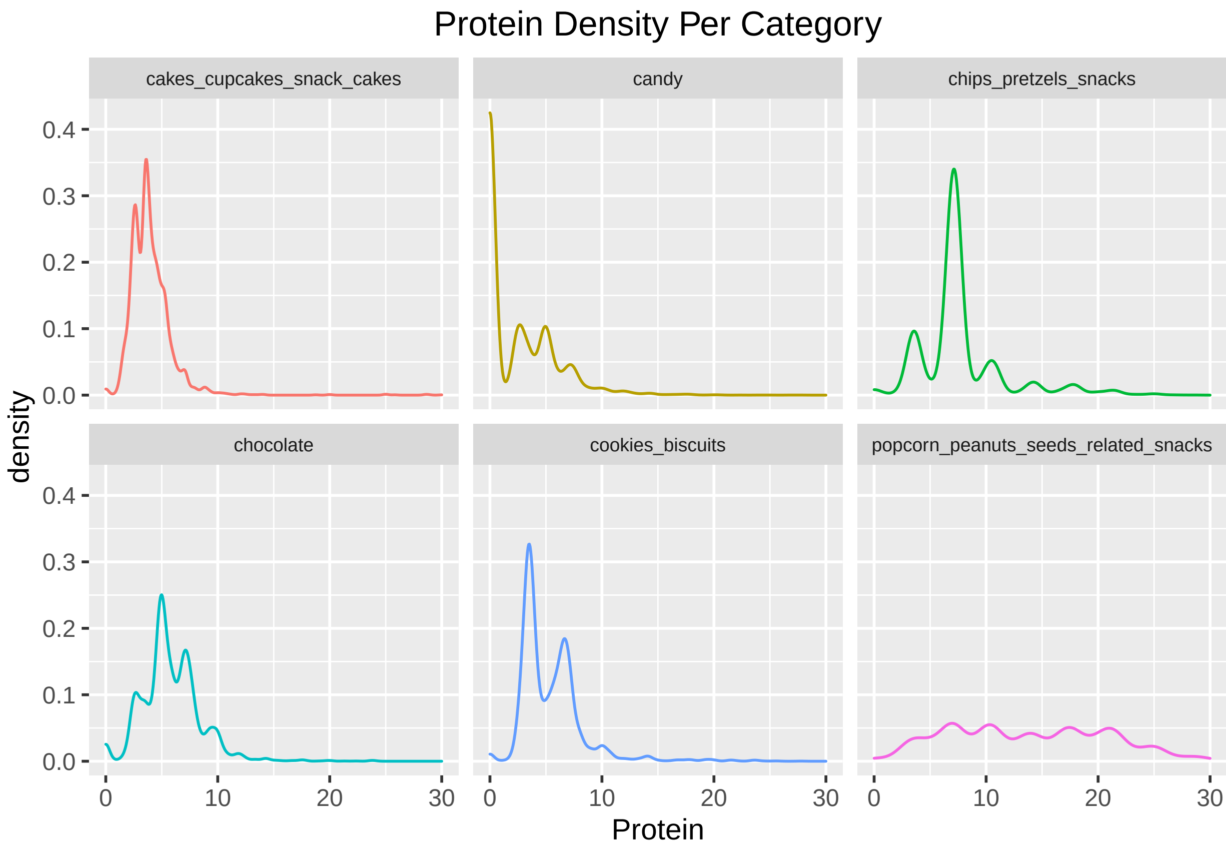
- For this section we will focus on the serving size feature , and on the four most common nutrients we will check when buying a snack - energy, protein, fat saturated and sodium.

**energy**



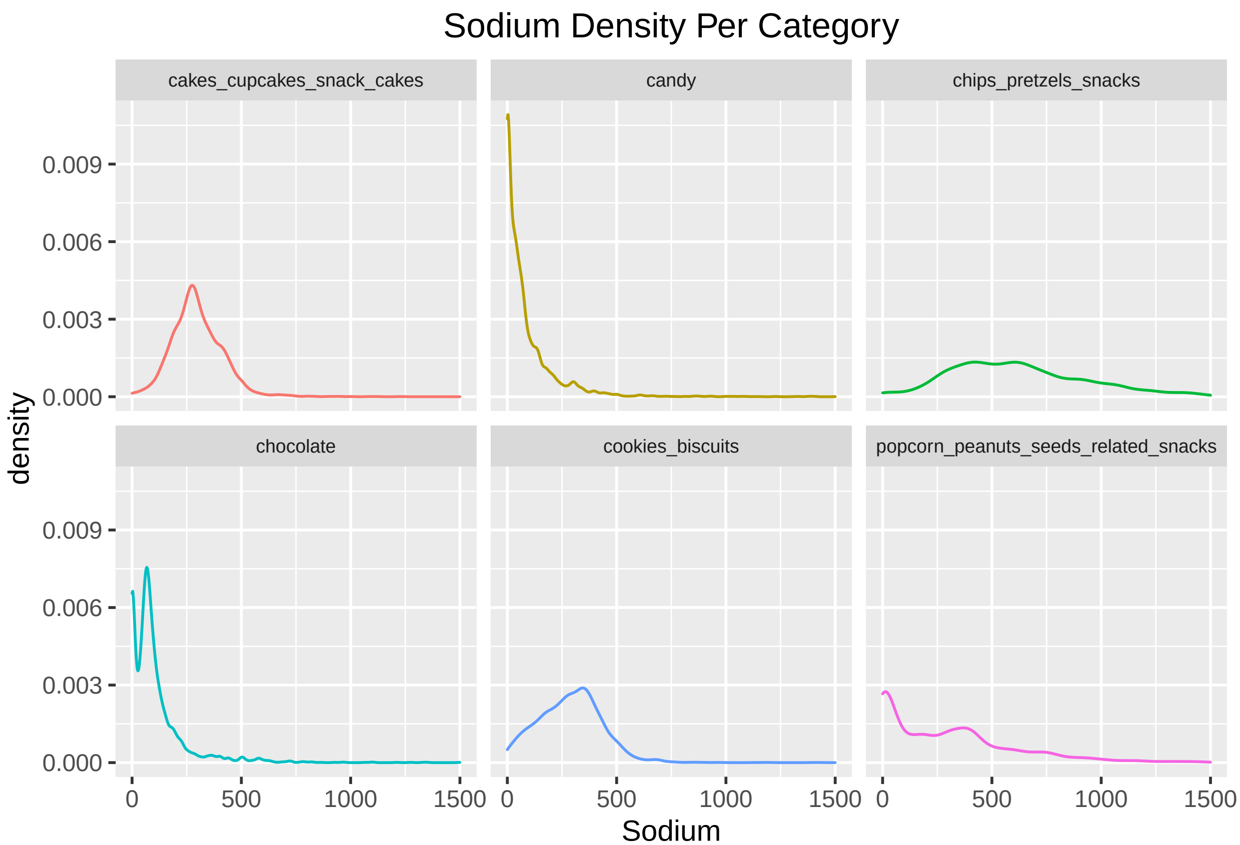

- We can see that popcorn_peanuts_seeds_related_snacks category has the highest energy, followed by chips_pretzels_snacks.
- cakes_cupcakes_snack_cakes and candy about the same, chocolate and cookies_biscuits about the same.

**protein**



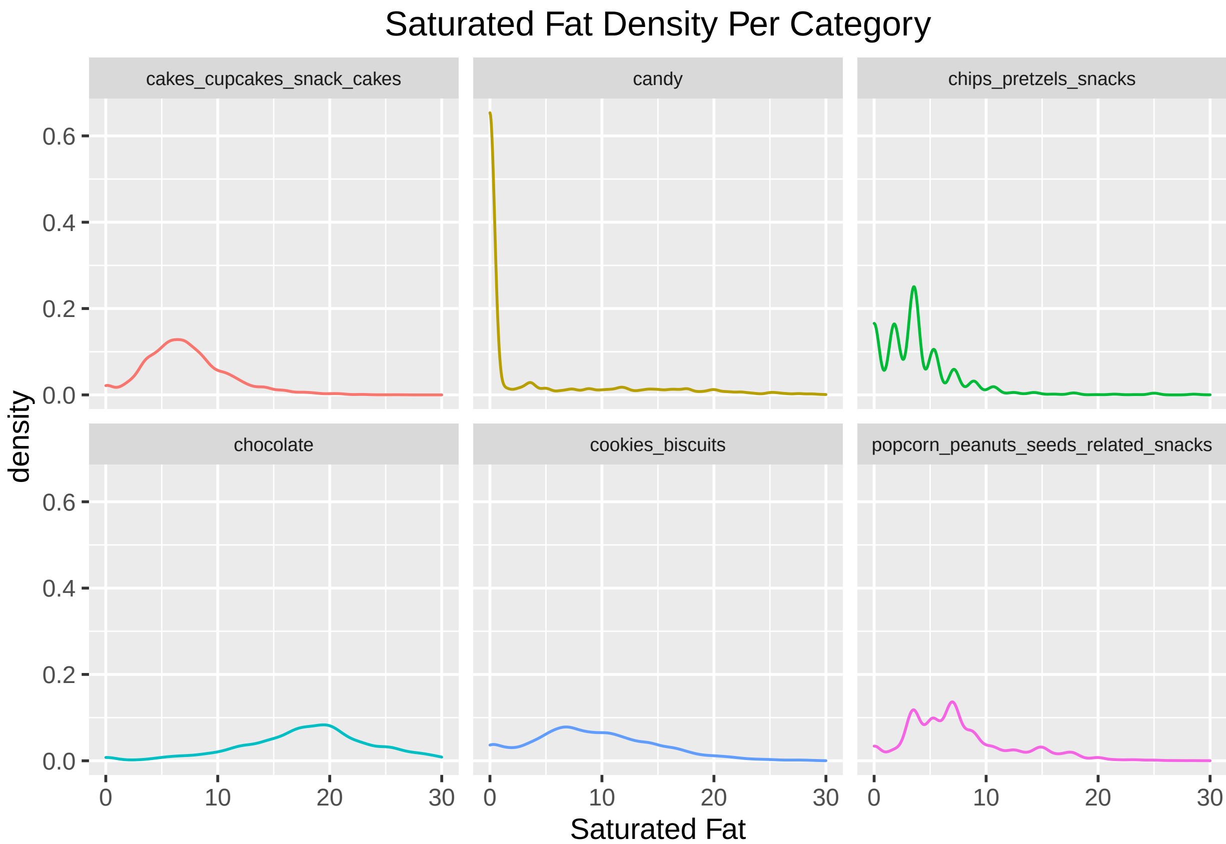

- popcorn_peanuts_seeds_related_snacks has the highest protein , while the other four expect for candy are about the same.

**sodium**



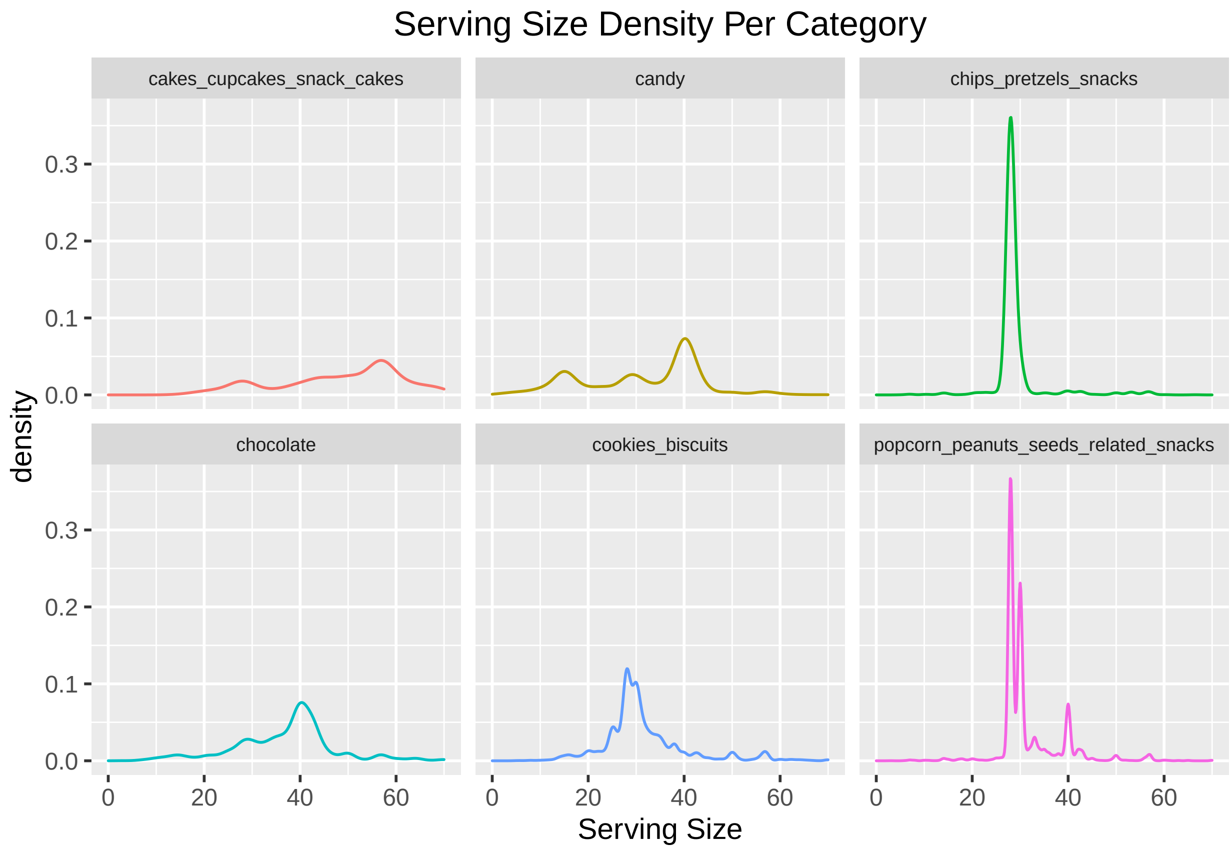

- chips_pretzels_snacks has the highest sodium , and the other four are lower but with different distribution.

**fat, saturated**




- chocolate has the highest fat, candy the lowest and the other four are about the same.

**serving size**




- cakes_cupcakes_snack_cakes has the highest serving size, chips_pretzels_snacks is very concentrated in about the 30 size.

## EDA on Image data

- My goal was to check if there is difference in the colors of the snacks packages, for this i calculated the mean of each rgb channel.
- Note - because most of the pictures have white background, the results are skewed upwards, but my assumption is that the order between the categories remain the same.
- Note - all of the python chunks were written in R markdown, the warning filter in those chunks is there because i had some "futurewarning", basically telling that append is going to be replaced with concat, warnings of that nature.

```
warnings.filterwarnings('ignore')
dir = os.listdir("C:/Users/itay/train")
df_img = pd.DataFrame()
```

- Function that takes the mean of each rgb channel and the index for every pic.

```
warnings.filterwarnings('ignore')
def img_func(img_path):
```

```
  idx = os.path.basename(img_path)
  idx = idx.split(".")[0]
  image = imread(img_path)
  red = image[:,:,0].flatten()
  green = image[:,:,1].flatten()
  blue = image[:,:,2].flatten()
  red = np.mean(red)
  green = np.mean(green)
  blue = np.mean(blue)
  dict = {"idx" : idx , "red" : red , "green" : green , "blue" : blue}
  return(dict)
```

- Looping through all the training pics.

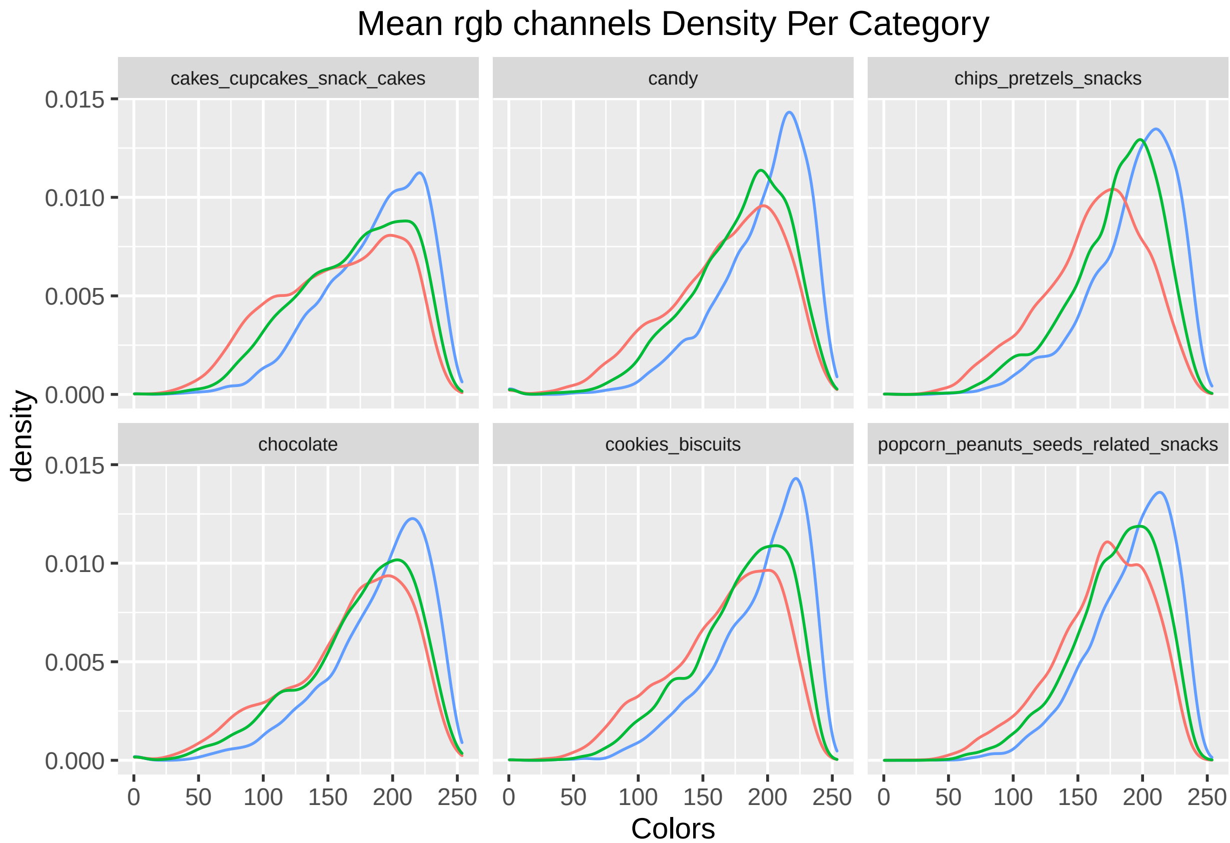
```
warnings.filterwarnings('ignore')
for directory in dir:
  sub_dir = "C:/Users/itay/train/" + directory
  sub_dir_enter = os.listdir(sub_dir)
  for image in sub_dir_enter:
    img = sub_dir + "/" + image
    temp_df = img_func(img)
    df_img = df_img.append(temp_df , ignore_index = True)
```

```
warnings.filterwarnings('ignore')
print(df_img.shape)
```

```
## (31751, 4)
```

- joining the pics DF with the training dataset.

```
data_images <- tibble(py$df_img)

data_images$idx <- as.numeric(data_images$idx)

final_train <- full_join(final_train , data_images , by = "idx")
```




- We can see for all the categories that blue is dominant, other than that there is no additional information that can help us in the prediction.