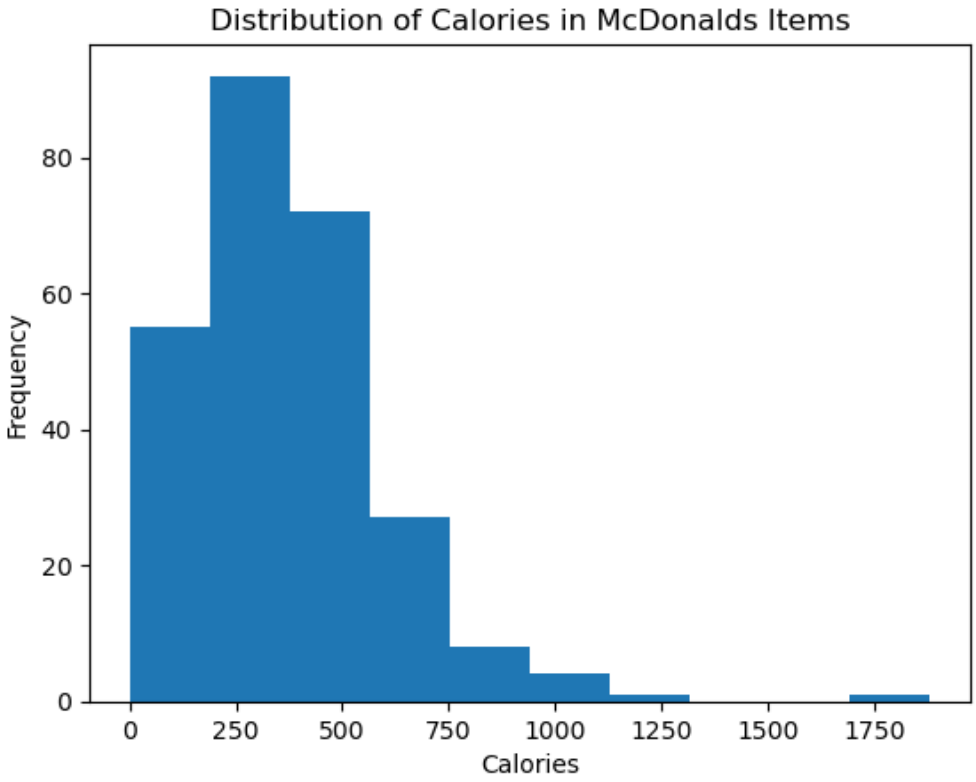
DSC 148: Homework 1

Kevin Wong

January 7, 2025

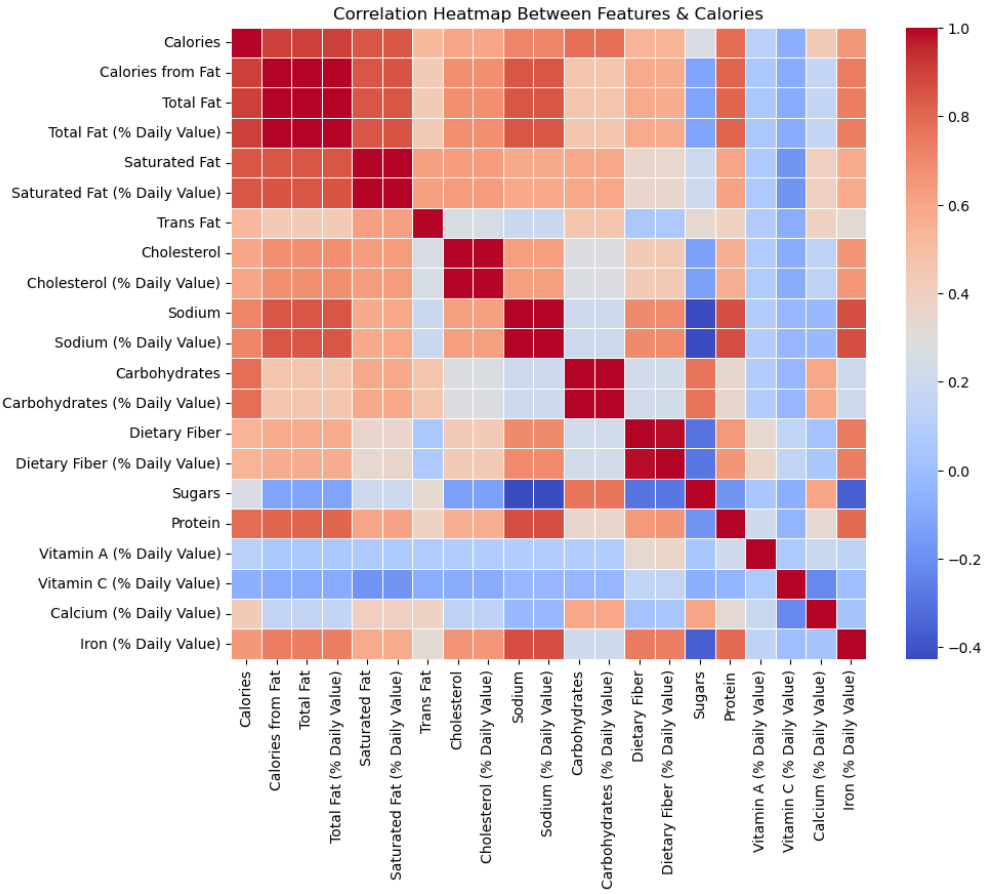
1. Data Exploration (20 points)

(a) Plot the histogram of the Calories. Comment on the datatype of the features.



The datatype of the feature "calories" are integers, which are a quantitative, continuous variable.

(b) Plot the correlation heatmap between features and Calories.



The diagonal elements of the heatmap will always be one because those are the areas where the same features are predicting each other. Since features can completely predict each other, there is no variability, which makes it so that the correlation between two of the exact same feature is 1.

(c) The features with the second and third largest positive correlation with Calories are:

- Second highest: "Calories from Fat"
- Third highest: "Total Fat"

(d) Features with negative correlation with Calories: "Vitamin C (% Daily Value)"

This feature does somewhat meet my expectation, because it is known that food with a lot of Vitamin C, like bell peppers, broccoli, brussel sprouts, and tomatoes, are vegetables that are typically lower in calories than other foods. On the other hand, a lot of very high calorie food, such as fast food, does not contain a lot of Vitamin C, which could explain the negative correlation between "Calories" and "Vitamin C (% Daily Value)"

2. Plotting (20 points)

(a) Scatter plots:

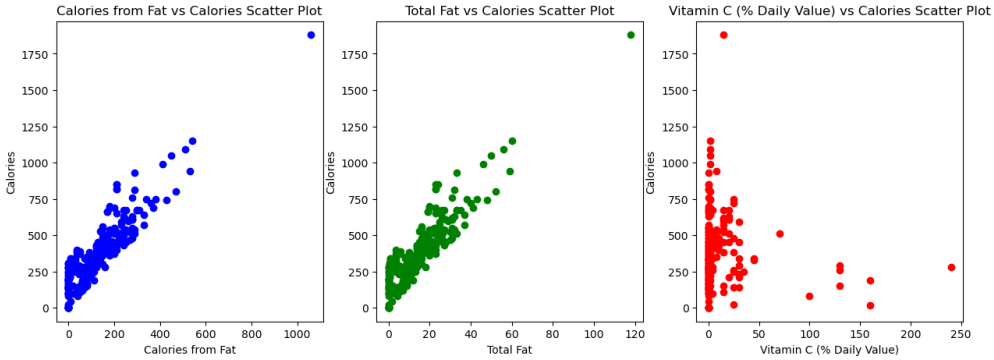


Figure 1: Scatter Plots of Selected Features vs. Calories

(b) Box plots:

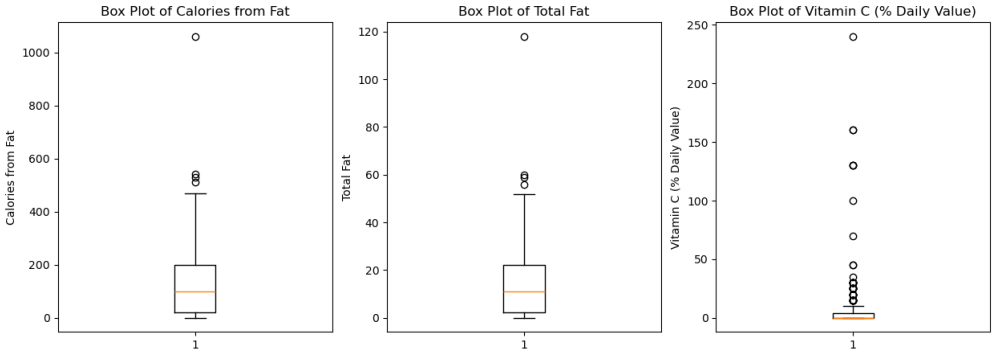


Figure 2: Box Plots of Selected Features

3. Data Pre-processing: Missing Values (20 points)

(a) Report median and standard deviation of all numerical features

Feature	std	median
Calories	240.269886	340.000000
Calories from Fat	127.875914	100.000000
Total Fat	14.205998	11.000000
Total Fat (% Daily Value)	21.885199	17.000000
Saturated Fat	5.321873	5.000000
Saturated Fat (% Daily Value)	26.639209	24.000000
Trans Fat	0.429133	0.000000
Cholesterol	87.269257	35.000000
Cholesterol (% Daily Value)	29.091653	11.000000
Sodium	577.026323	190.000000
Sodium (% Daily Value)	24.034954	8.000000
Carbohydrates	28.252232	44.000000
Carbohydrates (% Daily Value)	9.419544	15.000000
Dietary Fiber	1.567717	1.000000
Dietary Fiber (% Daily Value)	6.307057	5.000000
Sugars	28.679797	17.500000
Protein	11.426146	12.000000
Vitamin A (% Daily Value)	24.366381	8.000000
Vitamin C (% Daily Value)	26.345542	0.000000
Calcium (% Daily Value)	17.019953	20.000000
Iron (% Daily Value)	8.723263	4.000000

(b) Code for replacing outliers:

The code below replaces outliers by identifying them by creating a lower and upper bound per feature by linking the table from the previous question and replacing these "outlier" values that don't within these bounds with NaNs.

```
def replace_outliers(df, median_std_data):
    for column in df.columns:
        median = median_std_data.loc['50%', column]
        std = median_std_data.loc['std', column]
        bounds = [median - 3 * std, median + 3 * std]
        df[column] = df[column].where((df[column] >= bounds[0]) & (df
   [column] <= bounds[1]), np.nan)
    return df

cleaned_data = replace_outliers(numerical_data.copy(),
   median_std_data)
cleaned_data.describe().loc[['std', '50%']]

```

Below are the total number of NaNs for all of the features after running the code above:

Feature	Number of NaNs
Calories	3
Calories from Fat	4
Total Fat	4
Total Fat (% Daily Value)	4
Saturated Fat	0
Saturated Fat (% Daily Value)	0
Trans Fat	9
Cholesterol	5
Cholesterol (% Daily Value)	6
Sodium	5
Sodium (% Daily Value)	5
Carbohydrates	5
Carbohydrates (% Daily Value)	5
Dietary Fiber	8
Dietary Fiber (% Daily Value)	5
Sugars	4
Protein	2
Vitamin A (% Daily Value)	6
Vitamin C (% Daily Value)	7
Calcium (% Daily Value)	0
Iron (% Daily Value)	3

(c) Comparison of statistics: The following code replaces all the NaNs with mean values from the 'mean' table.

```
means = cleaned_data.describe().loc[['mean']]

def replace_nans(df):
    for column in df.columns:
        df[column] = df[column].fillna(means[column].values[0])
    return df


replaced_data = replace_nans(cleaned_data.copy())
replaced_data.describe().loc[["std", "50%"]]

```

Below are the means and standard deviations of the data, which got rid of outliers and replaced them with the means of their features:

Feature	std	median
Calories	210.710032	340.000000
Calories from Fat	105.225212	100.000000
Total Fat	11.692321	11.000000
Total Fat (% Daily Value)	17.983982	17.000000
Saturated Fat	5.321873	5.000000
Saturated Fat (% Daily Value)	26.639209	24.000000
Trans Fat	0.330815	0.000000
Cholesterol	57.319632	35.000000
Cholesterol (% Daily Value)	18.375069	11.000000
Sodium	498.403136	190.000000
Sodium (% Daily Value)	20.754357	8.000000
Carbohydrates	25.225053	44.000000
Carbohydrates (% Daily Value)	8.410548	15.000000
Dietary Fiber	1.301702	1.000000
Dietary Fiber (% Daily Value)	5.596463	5.000000
Sugars	26.252703	17.500000
Protein	10.236827	12.000000
Vitamin A (% Daily Value)	12.334957	8.000000
Vitamin C (% Daily Value)	9.587148	0.000000
Calcium (% Daily Value)	17.019953	20.000000
Iron (% Daily Value)	8.065942	4.000000

When compared to our original data, there are significant decreases in standard deviations in almost all the features, while there are little to no changes in the medians, indicating that when the outliers were changed into the means, this made the data more normalized.

4. Linear Regression (20 points)

(a) Multiple linear regression results:

$$\text{Calories} = \theta_0 + \theta_1 \times [\text{Carbohydrates}] + \theta_2 \times [\text{Protein}] + \theta_3 \times [\text{Total Fat}]$$

where:

- θ_0 = -1.8292919301434836
- θ_1 = 3.97896903
- θ_2 = 4.04524858
- θ_3 = 9.01862206

According to the multiple linear regression results, these are what the following refer to:

θ_0 being approximately -1.83 means that when a food has no carbs, protein, or total fat, a food will have a default starting point of -1.83 calories.

θ_1 being approximately 3.98 means that every carbohydrate contributes 3.98 calories towards a food item.

θ_2 being approximately 4.05 means that every gram of protein contributes 4.05 calories towards a food item.

θ_3 being approximately 9.02 means that every gram of total fat contributes 9.02 calories towards a food item.

(b) Simple linear regression results:

$$\text{Calories} = \theta_0 + \theta_1 \times [\text{Total Fat}]$$

where:

- $\theta_0 = 151.58818902493147$
- $\theta_1 = 15.29651666$

The coefficients are probably very different between the two linear regression models because the simple linear regression model can only rely on one feature to determine it's results. This means that it cannot consider a large portion of possibilities, unlike the previous model, because even if an item had 0 total fat, there was a chance it had protein or carbohydrates, which could be used in it's formula. In the simple linear regression results, it can only rely on total fat, so it needs something as a baseline point when there is no total fat in a food item.

(c) Training and testing results:

- Training MSE: 18.91564
- Testing MSE: 60.19673

The model performed significantly better on the training set than the test set. This could potentially mean there was overfitting, especially if there were certain items that were especially similar. Another reason that this could be is because the testing data mostly contained items from the category 'Smoothies Shakes', which the training data had no information about, leading to predictions that were not as accurate.

5. Logistic Regression (20 points)

(a) Expressing odds of prediction:

First let's establish our weights, and values, where $w = [\theta_1, \theta_2, ..., \theta_n]$ and $x = [x_1, x_2, ..., x_n]$. Additionally, we need to establish the meaning of a logistic regression function, which is in this case, results in us wanting to find the odds to predicting label 1. This results in the following equation:

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where z is the linear combination (dot product) of the weights and features. When placed into the expression to get the odds of an event, we get this, which can be simplfied, as shown below

$$\begin{aligned}
\text{odds} &= \frac{P(\text{label} = 1)}{1 - P(\text{label} = 1)} \\
&= \frac{\sigma(z)}{1 - \sigma(z)} \\
&= \frac{\frac{1}{1+e^{-z}}}{1 - \frac{1}{1+e^{-z}}} \\
&= \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} \\
&= \frac{1}{e^{-z}} \\
&= e^{z} \\
&= e^{w^T x}
\end{aligned}$$

(b) Analysis of odds ratio:

By increasing one feature value x_i by 1 and letting all the features remain the same, there is a slight ratio change. For simplicitie's sake, let's change the value x_1 into $x_1 + 1$. This means these are our weights and values:

$$\begin{aligned}
w &= [\theta_1, \theta_2, ..., \theta_n] \\
x &= [x_1 + 1, x_2, ..., x_n]
\end{aligned}$$

The calculation of the odds remains the same as before, however our final result of the odds is different now. Instead of our final result of $e^{w^T x} = e^{\theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n}$, the result for this increased feature value change is now now $e^{\theta_1 x_1 + \theta_1 + \theta_2 x_2 + ... + \theta_n x_n}$, essentially giving a bias term of θ_1 to the expression.

This new ratio of the new odds of predicting label 1 versus the old odds of predicitng label 1 are now:

$$\begin{aligned}
\text{ratio} &= \frac{e^{\theta_1 x_1 + \theta_1 + \theta_2 x_2 + ... + \theta_n x_n}}{e^{\theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n}} \\
&= e^{(\theta_1 x_1 + \theta_1 + \theta_2 x_2 + ... + \theta_n x_n) - (\theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n)} \\
&= e^{\theta_1}
\end{aligned}$$

Therefore. the ratio of new odds of predicting label 1 to the old odds of predicting label 1 in this case is e^{θ_1}, which when generalized to x_i, would have a ratio of e^{θ_i}. We can infer that raising the feature x_i by 1 results in multiplying the odds of predicting label 1 by e^{θ_i}, with odds decreasing or increasing by a factor of e^{θ_i}.

We can also deduce that since the weights directly correspond to the feature values, the larger a weight, the more influential and therefore stronger a weight is, the larger the change in chance for predicting a label.