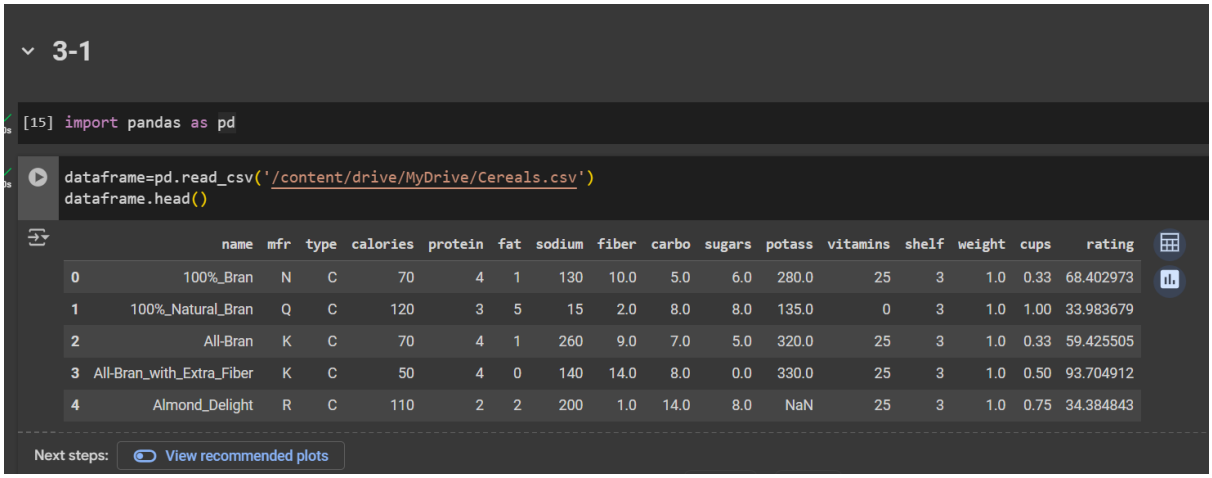
حوری دهش 9821413

.3

.1



3-1

```
[15] import pandas as pd

dataframe=pd.read_csv('/content/drive/MyDrive/Cereals.csv')
dataframe.head()
```

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100%_Bran	N	C	70	4	1	130	10.0	5.0	6.0	280.0	25	3	1.0	0.33	68.402973
1	100%_Natural_Bran	Q	C	120	3	5	15	2.0	8.0	8.0	135.0	0	3	1.0	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5.0	320.0	25	3	1.0	0.33	59.425505
3	All-Bran_with_Extra_Fiber	K	C	50	4	0	140	14.0	8.0	0.0	330.0	25	3	1.0	0.50	93.704912
4	Almond_Delight	R	C	110	2	2	200	1.0	14.0	8.0	NaN	25	3	1.0	0.75	34.384843

Next steps: View recommended plots

متغییرهای عددی:

– vitamins – potass – sugars – carbo – fiber – sodium – fat – protein – calories
rating – cups – weight – shelf

متغیرهای کیفی ترتیبی:

این متغیرها مقادیری دارند که قابل مرتب سازی هستند و ترتیب مشخص دارند:

shelf : این متغیر نشان دهنده مکان قرارگیری محصول در قفسه ها است و می تواند دارای ترتیب مشخصی باشد (مثلا قفسه‌های ۱، ۲، ۳)

متغیرهای کیفی اسمی:

این متغیرها مقادیری دارند که قابل دسته بندی هستند ولی ترتیب مشخص ندارند :

type – mfr – name

.2

3-2

```
numeric_columns = dataframe.select_dtypes(include=['float64', 'int64']).columns

summary_stats = dataframe[numeric_columns].describe().T
summary_stats['median'] = dataframe[numeric_columns].median()
summary_stats = summary_stats[['mean', 'median', 'min', 'max', 'std']]

print(summary_stats)
```

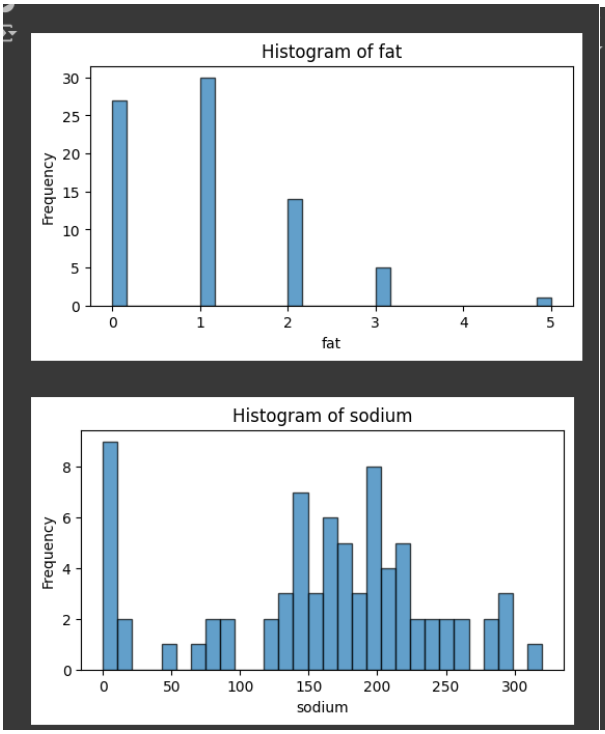
```
                mean      median         min         max         std
calories  106.883117  110.000000   50.000000  160.000000   19.484119
protein     2.545455    3.000000    1.000000    6.000000    1.094790
fat         1.012987    1.000000    0.000000    5.000000    1.006473
sodium    159.675325  180.000000    0.000000  320.000000   83.832295
fiber       2.151948    2.000000    0.000000   14.000000    2.383364
carbo      14.802632   14.500000    5.000000   23.000000    3.907326
sugars      7.026316    7.000000    0.000000   15.000000    4.378656
potass     98.666667   90.000000   15.000000  330.000000   70.410636
vitamins   28.246753   25.000000    0.000000  100.000000   22.342523
shelf       2.207792    2.000000    1.000000    3.000000    0.832524
weight      1.029610    1.000000    0.500000    1.500000    0.150477
cups        0.821039    0.750000    0.250000    1.500000    0.232716
rating     42.665705   40.400208   18.042851   93.704912   14.047289
```

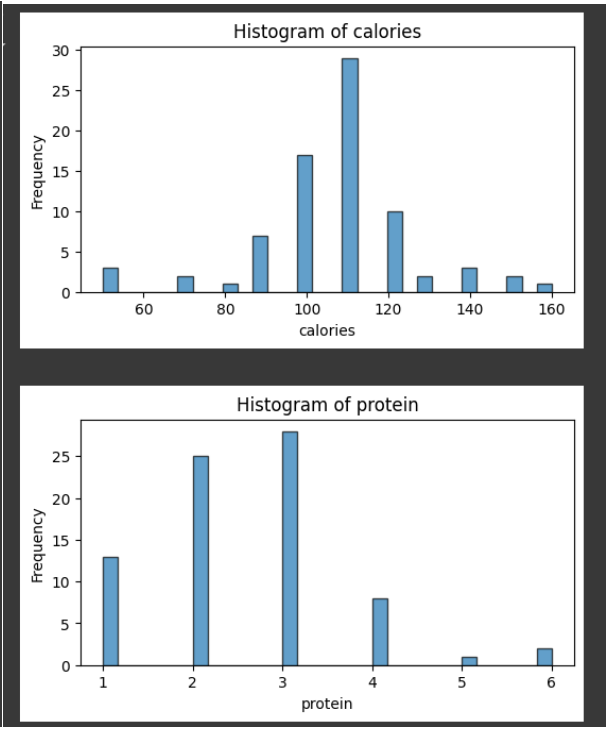
.3

3-3

```
[18] import matplotlib.pyplot as plt
```

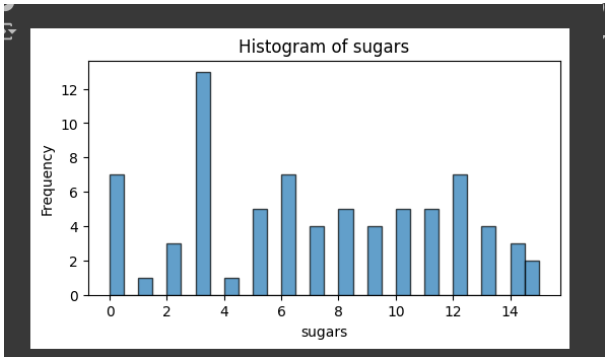
```
for column in numeric_columns:
    plt.figure(figsize=(6, 3))
    plt.hist(dataframe[column].dropna(), bins=30, edgecolor='k', alpha=0.7)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.grid(False)
    plt.show()
    print("\n")
```

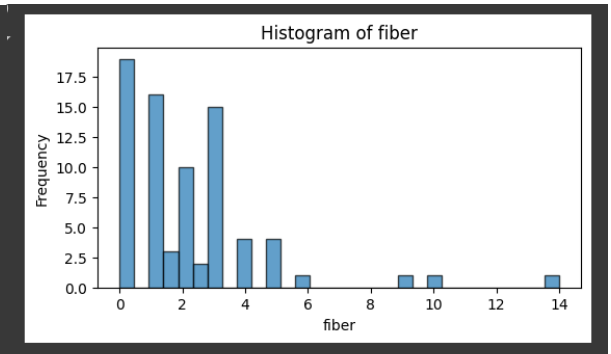

Histogram of fat
Frequency
fat

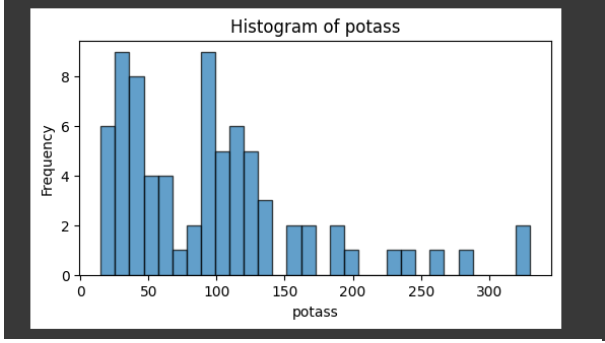

Histogram of calories
Frequency
calories

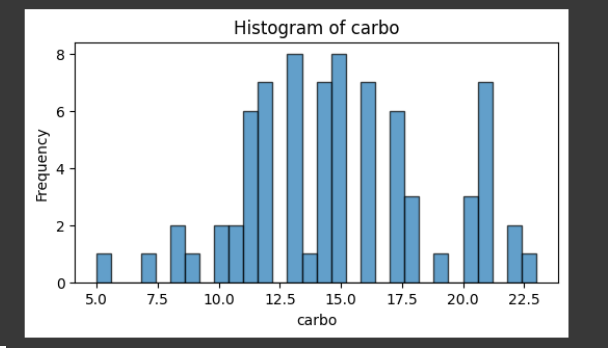
Histogram of sodium
Frequency
sodium

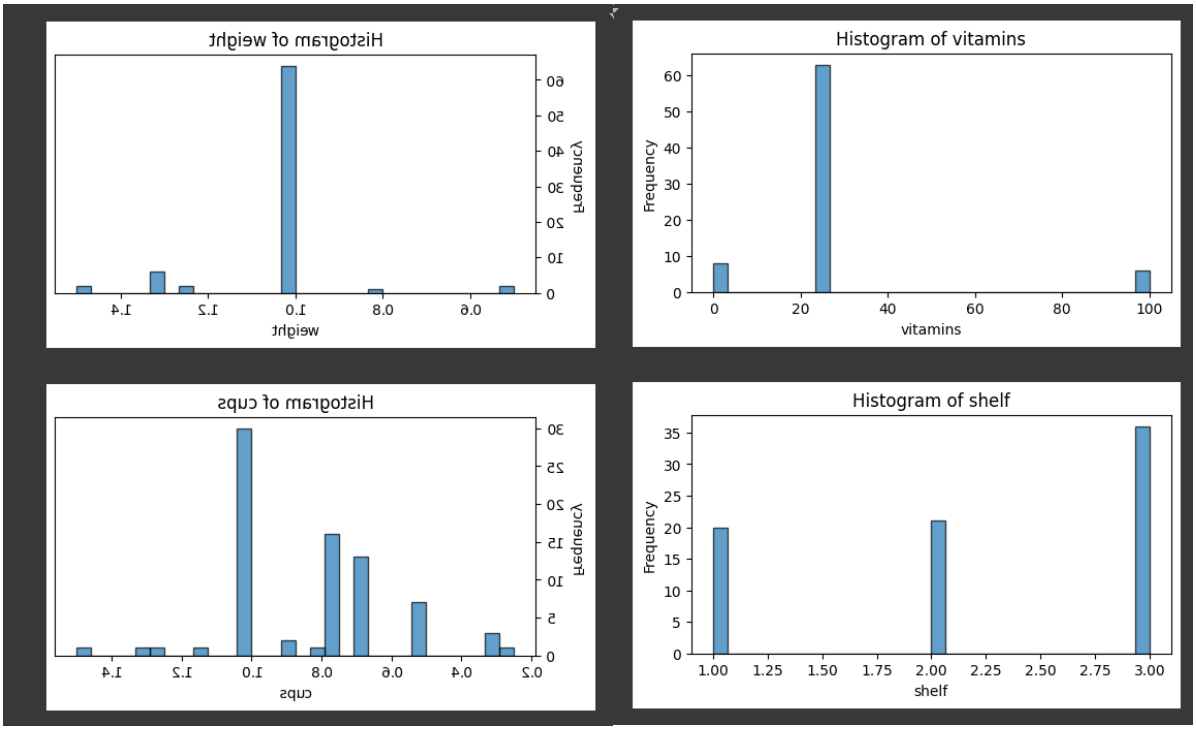
Histogram of protein
Frequency
protein

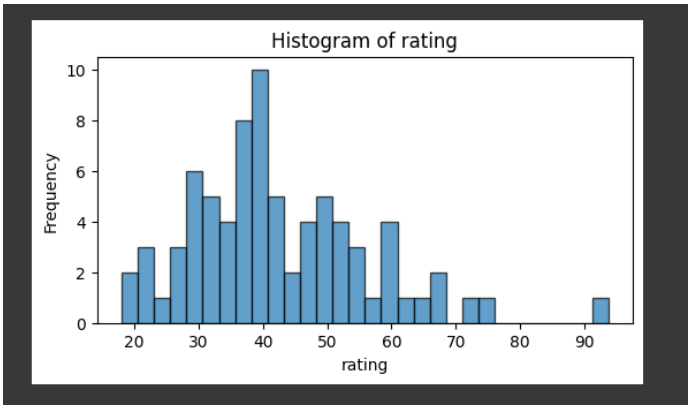

Histogram of sugars
Frequency
sugars


Histogram of fiber
Frequency
fiber


Histogram of potass
Frequency
potass


Histogram of carbo
Frequency
carbo





پرسش 1:

```
[24] std_devs = dataframe[numeric_columns].std()

     highest_variability_variable = std_devs.idxmax()
     highest_variability_value = std_devs.max()

     print(f"Variable with highest variability: {highest_variability_variable}")
     print(f"Standard deviation of this variable: {highest_variability_value}")

     Variable with highest variability: sodium
     Standard deviation of this variable: 83.83229524009317
```

پرسش 2:

```python
skewness = dataframe[numeric_columns].skew()

print("Skewness of numeric variables:")
print(skewness)
```

```
Skewness of numeric variables:
calories   -0.445407
protein     0.745830
fat         1.165989
sodium     -0.575711
fiber       2.431675
carbo       0.112726
sugars      0.044445
potass      1.400355
vitamins    2.463704
shelf      -0.410339
weight      0.309857
cups       -0.104981
rating      0.910240
dtype: float64
```

1. calories : -0.445407 (چوله به چپ)

2. protein : 0.745830 (چوله به راست)

3. fat : 1.165989 (چوله به راست)

4. sodium : -0.575711 (چوله به چپ)

5. fiber : 2.431675 (چوله به راست)

6. carbo : 0.112726 (تقریبا متقارن، کمی چوله به راست)

7. sugars : 0.044445 (تقریبا متقارن)

8. potass : 1.400355 (چوله به راست)

9. vitamins : 2.463704 (چوله به راست)

10. shelf : -0.410339 (چوله به چپ)

11. weight : 0.309857 (تقریبا متقارن، کمی چوله به راست)

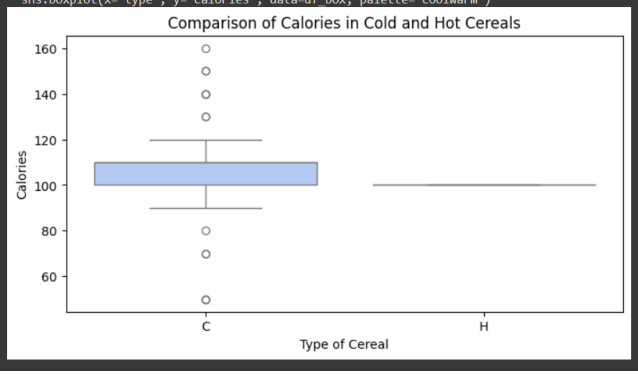
12. cups : -0.104981 (تقریبا متقارن)

13. rating : 0.910240 (چوله به راست)

.4

```
3-4

[26] import seaborn as sns

cold_cereals = dataframe[dataframe['type'] == 'C']
hot_cereals = dataframe[dataframe['type'] == 'H']

df_box = dataframe[['type', 'calories']]

plt.figure(figsize=(8, 4))
sns.boxplot(x='type', y='calories', data=df_box, palette="coolwarm")
plt.title('Comparison of Calories in Cold and Hot Cereals')
plt.xlabel('Type of Cereal')
plt.ylabel('Calories')
plt.show()
```

```
<ipython-input-31-a762fe3c6915>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='type', y='calories', data=df_box, palette="coolwarm")
```



1. غلات سرد (C):

IQR حدود 95 تا 110 کالری است.

میانه مقدار 100 کالری است که با خط درون جعبه مشخص شده است.

نقاط خارج از جعبه نشان‌دهنده نقاط پرت (Outliers) هستند.

خطوط بالا و پایین جعبه (Whiskers) گستره داده هایی که داخل 1.5 برابر IQR از Q1 (چارک اول) و Q3 (چارک سوم) قرار دارند را نشان می دهند.

2. غلات گرم (H):

فقط یک خط نازک مشاهده می شود که نشان دهنده این است که تمامی داده های مربوط به غلات گرم در یک مقدار ثابت قرار دارند.

همین طور به نظر می رسد تمامی داده های مربوط به غلات گرم کالری های بسیار مشابه یا یکسانی داشته باشند.

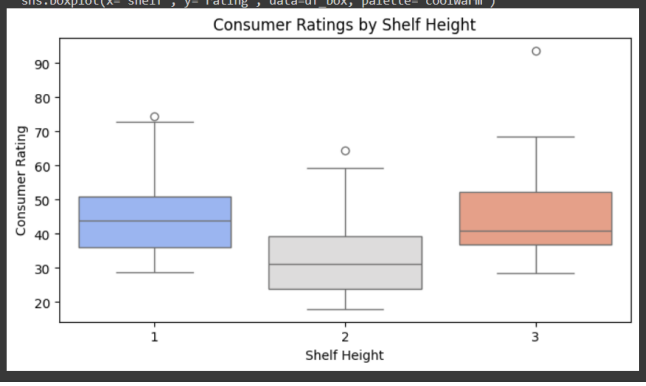
پس:

غلات سرد دارای تنوع بیشتری در مقدار کالری هستند، که این موضوع از نقاط پرت (Outliers) و دامنه گسترده کالری‌ها در این نوع غلات قابل مشاهده است و غلات گرم کالری های بسیار مشابه یا یکسانی دارند که باعث شده نمودار جعبه‌ای تقریبا نامرئی داشته باشند (یعنی تنوع بسیار کم در کالری)

.5

```
3-5

df_box = dataframe[['shelf', 'rating']]

plt.figure(figsize=(8, 4))
sns.boxplot(x='shelf', y='rating', data=df_box, palette="coolwarm")
plt.title('Consumer Ratings by Shelf Height')
plt.xlabel('Shelf Height')
plt.ylabel('Consumer Rating')
plt.show()
```

```
<ipython-input-37-7ccb011b23ba>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='shelf', y='rating', data=df_box, palette="coolwarm")
```



بر اساس نمودارهای جعبه‌ای، تفاوت‌های زیر بین سه طبقه مربوط به ارتفاع قفسه مشاهده میشود:

قفسه 1 دارای میانه بالاتری نسبت به قفسه های 2 و 3 است.

قفسه 1 دارای دامنه تغییرات (IQR) نسبتا بزرگتری است.

قفسه 2 و 3 میانه های مشابهی دارند ولی دارای نقاط پرت متفاوتی هستند.

با توجه به این تفاوت‌ها، برای پیش بینی رتبه بندی مصرف کننده از ارتفاع قفسه، نگهداری تمامی سه طبقه مربوط به ارتفاع قفسه به نظر منطقی می‌آید، زیرا هر کدام از طبقات دارای ویژگی های منحصر به فردی هستند که می تواند در مدل پیش بینی موثر باشد. این تفاوت‌ها در میانه، IQR، و نقاط پرت نشان می دهند که هر طبقه ممکن است اطلاعات مفیدی برای پیش بینی ارائه دهد.

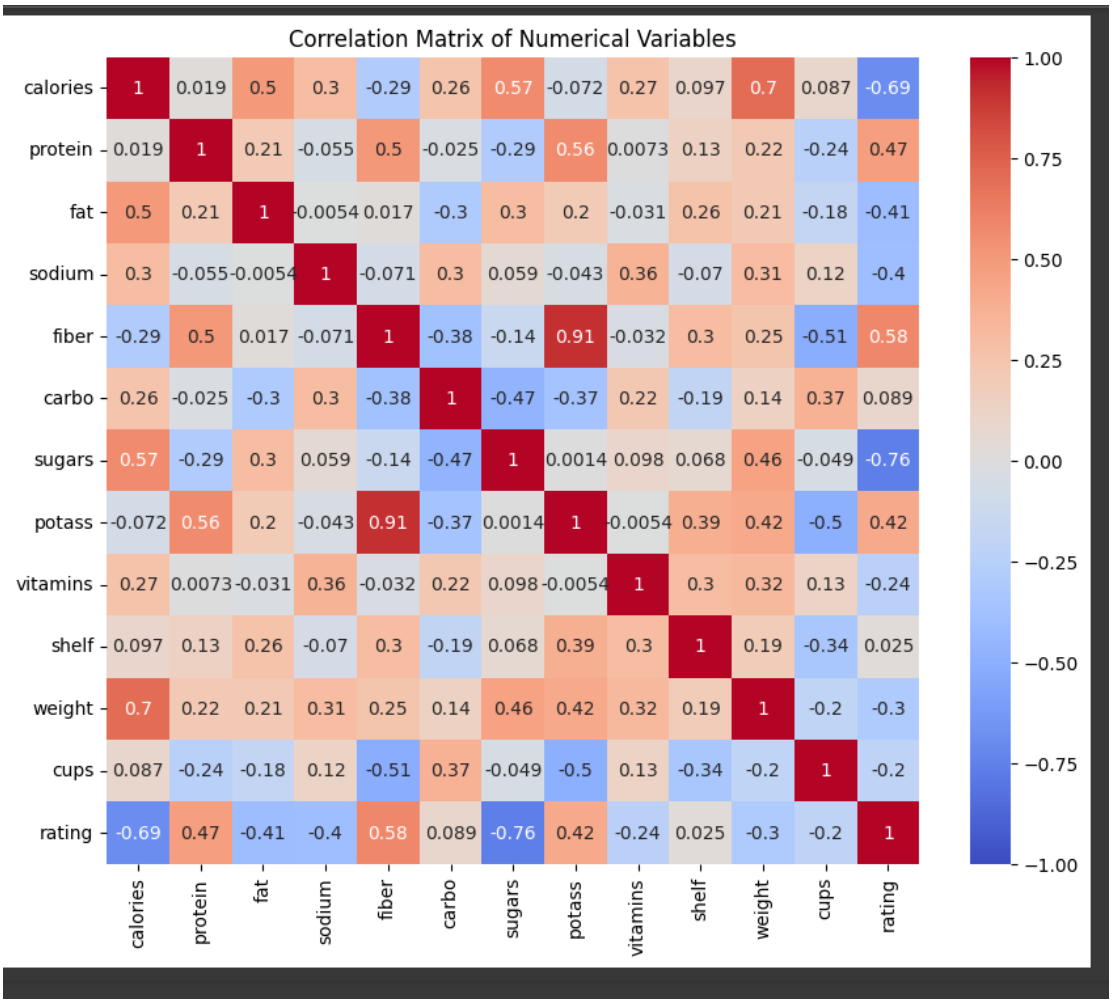
پس لازم است تا تمامی سه طبقه مربوط به ارتفاع قفسه را نگهداری کنیم.

.6

3-6

```
[38] correlation_matrix = dataframe[numeric_columns].corr()
```

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix of Numerical Variables')
plt.show()
```



پرسش 1:

زوج fiber و potass با مقدار همبستگی 0.91

زوج calories و weight با مقدار همبستگی 0.70

زوج sugars و rating با مقدار همبستگی 0.76- (این مقدار منفی است ولی از نظر قدر مطلق بالا است)

پرسش 2:

1- برای کاهش تعداد متغیرها می‌توان از تکنیک های کاهش ابعاد مثل تحلیل مولفه های اصلی (PCA) استفاده کرد. در این تکنیک ها، متغیرهایی که دارای همبستگی بالایی هستند می توانند به یک مولفه جدید ترکیب شوند. به عنوان مثال، از آنجایی که متغیرهای fiber و potass دارای همبستگی بالایی هستند، می‌توان آنها را به یک مولفه جدید ترکیب کرد و تنها یکی از آنها را در تحلیل ها به کار برد.
2- زمانی که دو تا متغییر همبستگی بالایی با هم دارند چه مثبت و چه منفی از بین آنها یکی را نگه داشته و دیگری را حذف می کنیم. ملاک نگه داشتن یکی از این دو متغییر این می باشد که کدام یک از آنها با فیلد هدف یا Target همبستگی بالاتری دارد.

پرسش 3:

نرمال سازی داده ها (به عنوان مثال با استفاده از روش Z-score) مقادیر داده ها را به مقیاسی مشابه تبدیل می کند، اما تاثیری بر روی مقادیر همبستگی ندارد. همبستگی تنها به روابط نسبی بین داده ها بستگی دارد و نرمال سازی این روابط را تغییر نمی دهد. به عبارت دیگر، اگر داده ها نرمال‌سازی شوند، مقادیر همبستگی تغییر نخواهند کرد.

.4

.1

4-1

```
dataframe2=pd.read_csv('/content/drive/MyDrive/diabetes.csv')
dataframe2.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next steps: View recommended plots

```
dataframe2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

.2

4-2

```
missing_values = dataframe2.isnull().sum()
print(missing_values)
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

.3

4-3

```
summary = dataframe2.describe()

min_values = summary.loc['min']
print(min_values)
```

```
Pregnancies                 0.000
Glucose                     0.000
BloodPressure               0.000
SkinThickness               0.000
Insulin                     0.000
BMI                         0.000
DiabetesPedigreeFunction    0.078
Age                        21.000
Outcome                     0.000
Name: min, dtype: float64
```

ستون های Glucose (قندخون)، BloodPressure (فشارخون)، SkinThickness (ضخامت پوست)، Insulin (انسولین)، و BMI (شاخص توده بدنی) دارای مقدار مینیمم صفر هستند. این مقادیر به طور معمول نمی توانند صفر باشند (به عنوان مثال، فشار خون یا قند خون نمی‌تواند صفر باشد). این نشان دهنده وجود داده‌های گمشده یا داده‌های نادرست است که به جای مقادیر واقعی وارد شده‌اند.

مقدار صفر در این ستون ها معمولا به معنی نبود داده یا خطا در جمع آوری داده ها است.

ستون‌های دیگر:

DiabetesPedigreeFunction مقدار مینیمم 0.078 دارد که معقول به نظر می رسد.

Age مقدار مینیمم 21.000 دارد که منطقی است.

Outcome نیز دارای مقدار مینیمم 0.000 است که منطقی است زیرا این ستون احتمالا نشان دهنده برچسب بیماری دیابت (مثلا 0 برای عدم دیابت و 1 برای دیابت) است.

نتیجه گیری درباره داده های گمشده:

مقادیر صفر در ستون هایی که نباید صفر باشند (مثل فشار خون، قند خون، انسولین و ...) احتمالا نمایانگر داده های گمشده هستند.

برای رسیدگی به داده های گمشده، باید مقادیر صفر را به عنوان مقادیر گمشده (NaN) در نظر بگیریم و سپس از تکنیک های مناسب مانند پر کردن با میانگین، میانه، یا استفاده از روش های پیشرفته‌تر برای تخمین داده‌های گمشده استفاده کنیم.

.4

4-4

```
[48] import numpy as np

dataframe2.replace(0, np.nan, inplace=True)

print(dataframe2.head())
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0          6.0    148.0           72.0           35.0      NaN  33.6
1          1.0     85.0           66.0           29.0      NaN  26.6
2          8.0    183.0           64.0            NaN      NaN  23.3
3          1.0     89.0           66.0           23.0     94.0  28.1
4          NaN    137.0           40.0           35.0    168.0  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50      1.0
1                     0.351   31      NaN
2                     0.672   32      1.0
3                     0.167   21      NaN
4                     2.288   33      1.0
```

```
[53] from sklearn.impute import SimpleImputer

imputer_mean = SimpleImputer(strategy='mean')
dataframe_mean = pd.DataFrame(imputer_mean.fit_transform(dataframe2), columns=dataframe2.columns)

print("Using mean strategy:\n", dataframe_mean.head())

imputer_mode = SimpleImputer(strategy='most_frequent')
dataframe_mode = pd.DataFrame(imputer_mode.fit_transform(dataframe2), columns=dataframe2.columns)

print("\nUsing mode strategy:\n", dataframe_mode.head())
```

```
Using mean strategy:
    Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0     6.000000    148.0           72.0       35.00000  155.548223  33.6
1     1.000000     85.0           66.0       29.00000  155.548223  26.6
2     8.000000    183.0           64.0       29.15342  155.548223  23.3
3     1.000000     89.0           66.0       23.00000   94.000000  28.1
4     4.494673    137.0           40.0       35.00000  168.000000  43.1

   DiabetesPedigreeFunction   Age  Outcome
0                     0.627  50.0      1.0
1                     0.351  31.0      1.0
2                     0.672  32.0      1.0
3                     0.167  21.0      1.0
4                     2.288  33.0      1.0

Using mode strategy:
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0           6.0    148.0           72.0           35.0    105.0  33.6
1           1.0     85.0           66.0           29.0    105.0  26.6
2           8.0    183.0           64.0           32.0    105.0  23.3
3           1.0     89.0           66.0           23.0     94.0  28.1
4           1.0    137.0           40.0           35.0    168.0  43.1

   DiabetesPedigreeFunction   Age  Outcome
0                     0.627  50.0      1.0
1                     0.351  31.0      1.0
2                     0.672  32.0      1.0
3                     0.167  21.0      1.0
4                     2.288  33.0      1.0
```