Johnston550TermProject

September 20, 2024

BMI/Heart Disease Analysis

INTRODUCTION

During this analysis I would like to find the correlation of factors that are related to overweight individuals and see if there is a way to predict when a person is headed towards becoming obese. I would also like to locate a formula to escaping that scenario as quickly and effectively as possible for those who have found themselves in it. The dataset I have chosen has numerous columns of factors that can influence a persons ability to gain or lose weight. The target variable within the dataset that I have chosen will be "Physical activity frequency". I have chosen this variable as I believe there are plenty of variables that measure dietary factors but no others that measure exercise or physical activity to a high enough level. The only other variable being "MTRANS Walking" however I made the decision that it would be a less accurate predictor.

I believe that this is an issue that would be deemed as being valuable to be solved due to epidemic of both mental and physical disease that is plaguing the world and especially the United States.

[3]:
```
# Imports neccessary libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

[4]:
```
# loads and displays dataset

df = pd.
 ↪read_csv("estimation_of_obesity_levels_based_on_eating_habits_and_physical_condition.
 ↪csv")

df.head()
```

[4]:

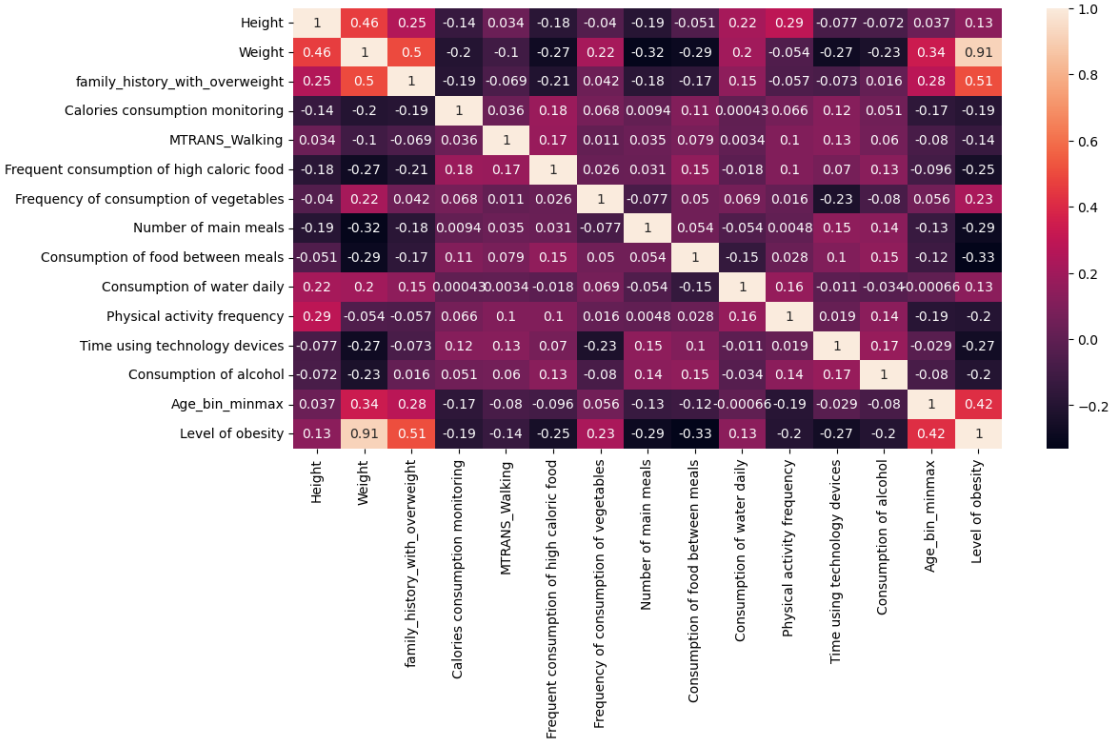
	Height	Weight	family_history_with_overweight	SCC	MTRANS_Walking	\
0	1.62	64.0	1	0	0	
1	1.52	56.0	1	1	0	
2	1.80	77.0	1	0	0	
3	1.80	87.0	0	0	1	
4	1.78	89.8	0	0	0	

```
     FAVC_z  FCVC_minmax     NCP_z  CAEC_minmax  CH2O_minmax  FAF_minmax  \
0  2.766876          0.5  0.404704     0.333333          0.5    0.000000
1  2.766876          1.0  0.404704     0.333333          1.0    1.000000
2  2.766876          0.5  0.404704     0.333333          0.5    0.666667
3  2.766876          1.0  0.404704     0.333333          0.5    0.666667
4  2.766876          0.5  2.164116     0.333333          0.5    0.000000

      TUE_z    CALC_z  Age_bin_minmax  NObeyesdad
0  0.550985  1.439033            0.25           1
1  1.092724  0.516552            0.25           1
2  0.550985  2.472136            0.50           1
3  1.092724  2.472136            0.75           2
4  1.092724  0.516552            0.50           3
```

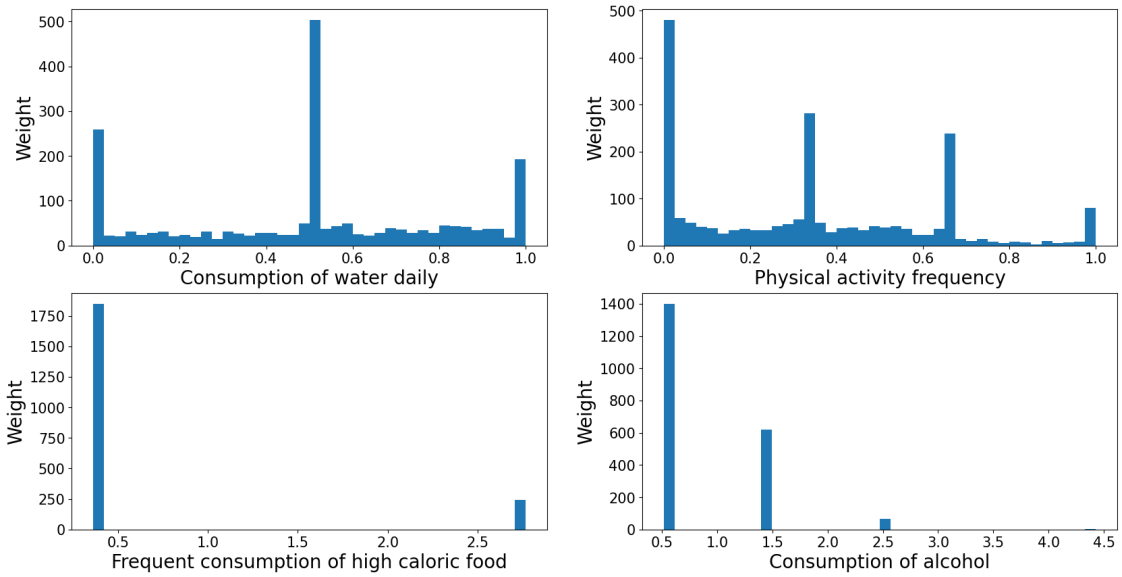
[5]:
```
# renames column names to be easier to read

df.rename(columns={"SCC": "Calories consumption monitoring", "FAVC_z":␣
 ↪"Frequent consumption of high caloric food", "FCVC_minmax": "Frequency of␣
 ↪consumption of vegetables",
                   "NCP_z": "Number of main meals", "CAEC_minmax": "Consumption␣
 ↪of food between meals", "CH2O_minmax": "Consumption of water daily",␣
 ↪"FAF_minmax": "Physical activity frequency",
                    "TUE_z": "Time using technology devices", "CALC_z":␣
 ↪"Consumption of alcohol", "NObeyesdad": "Level of obesity"}, inplace = True)
```

[6]:
```
# Creates a heatmap to analyze the correlation between each variable

plt.figure(figsize = (12,6))
sns.heatmap(df.corr(), annot = True)
```

[6]:
```
<Axes: >
```

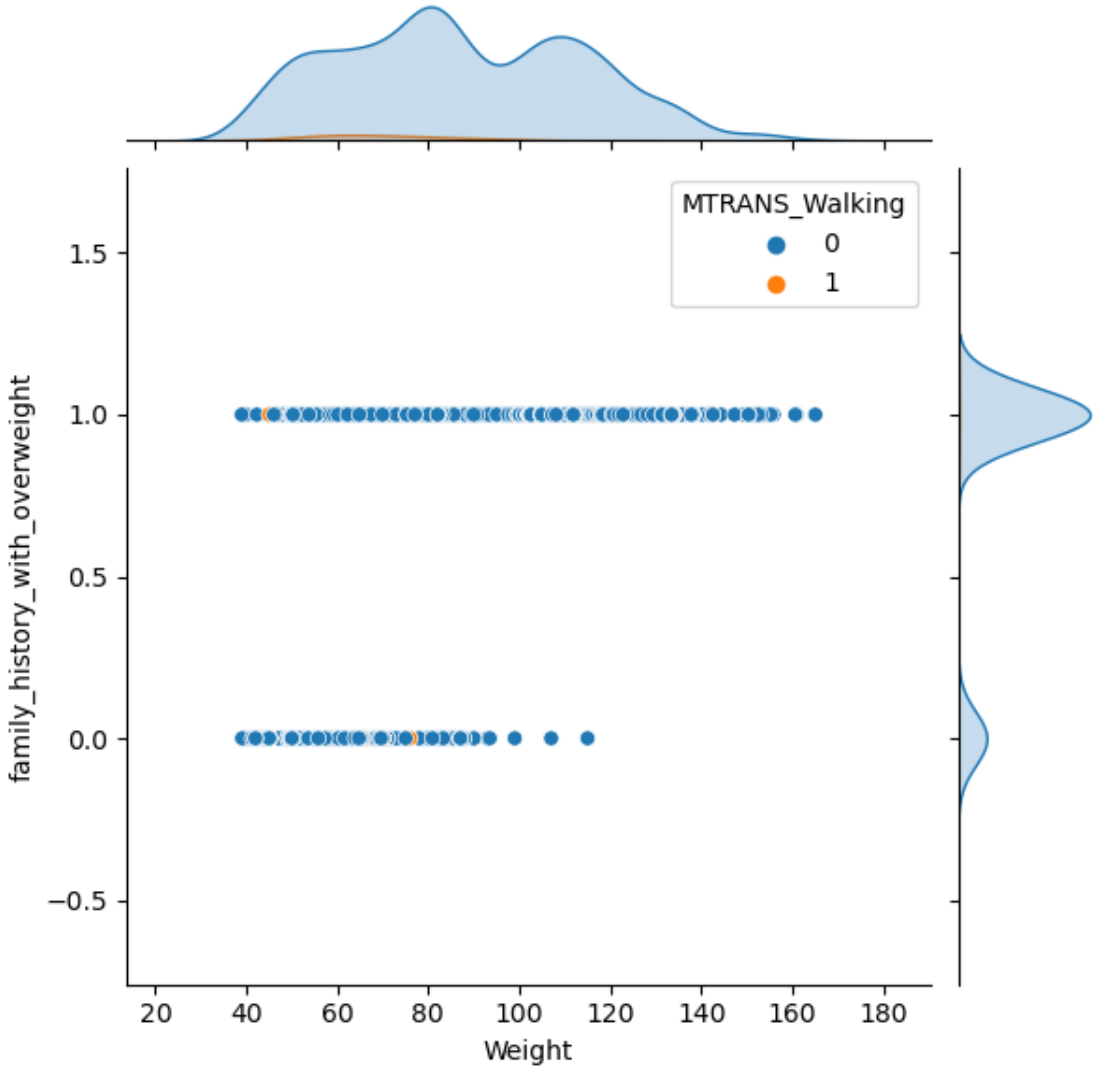


[7]:
```
# Displays the relationship of some of the more important variables to weight


important_variables = ['Consumption of water daily', 'Physical activity␣
 ↪frequency',
                       'Frequent consumption of high caloric food',␣
 ↪'Consumption of alcohol']
xaxes = important_variables
yaxes = ['Weight', 'Weight', 'Weight', 'Weight']
plt.rcParams['figure.figsize'] = (20, 10)
fig, axes = plt.subplots(nrows = 2, ncols = 2)
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(df[important_variables[idx]].dropna(), bins = 40)
    ax.set_xlabel(xaxes[idx], fontsize = 20)
    ax.set_ylabel(yaxes[idx], fontsize = 20)
    ax.tick_params(axis = 'both', labelsize = 15)
plt.show()
```

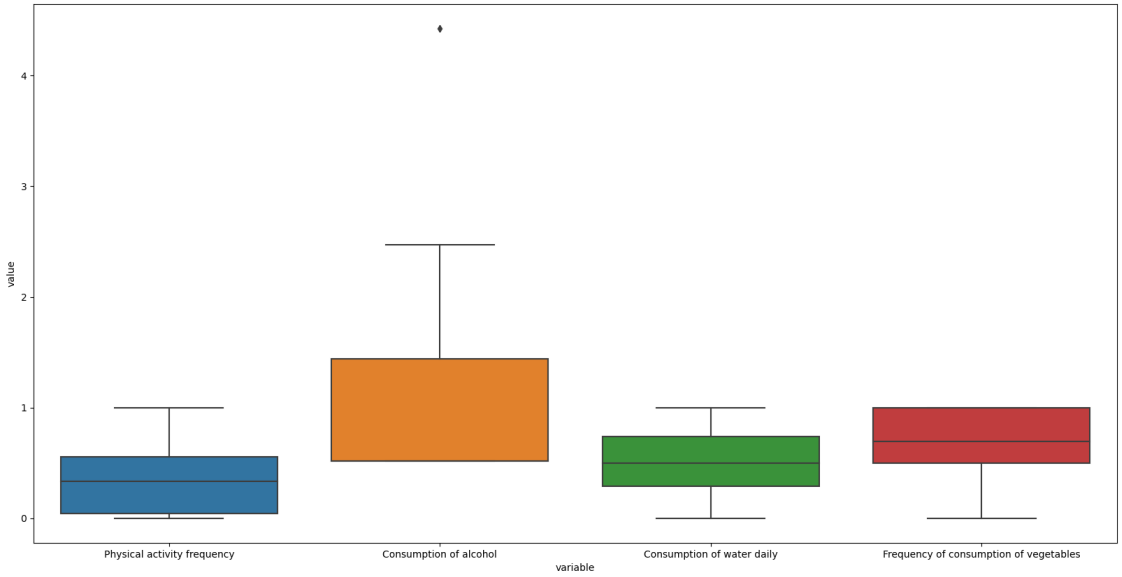


[8]:
```
# Analyzed the weight of those wkth weight issues and those with familial traits, grouped together by those who walk regularly

sns.jointplot(data = df, x = "Weight", y = "family_history_with_overweight", hue = 'MTRANS_Walking')
```

[8]: <seaborn.axisgrid.JointGrid at 0x7f7507582190>



[9]:
```
# Boxplot to locate any outlier values within the dataset related to each␣
 ↪individuals diet

sns_df = pd.DataFrame(data = df, columns = ['Physical activity frequency',␣
 ↪'Consumption of alcohol', 'Consumption of water daily', 'Frequency of␣
 ↪consumption of vegetables'])
sns.boxplot(x = "variable", y = "value", data = pd.melt(sns_df))
plt.show()
```



Overview/Conclusion

So far during my initial analysis of this topic I have noticed that there are certain factors that I would have originally believed to have been strongly correlated with the downward progression of obesity that are not, such as the frequency of physical activity. I have also discovered the vast amount of outliers within this particular dataset that I will need to be careful of. Almost every chart that measured the frequency of an items consumption has not proven to be aligned with my initial hypothesis for that variable. In conclusion, my initial thoughts are that lifestyle is an obviously vital part in preventing obesity but genetics/family history may have a MUCH more integral role to play than I had initially thought.

START OF MILESTONE 2

[12]:
```
# displays the dataset in its current form

df.head()
```

[12]:

	Height	Weight	family_history_with_overweight	Calories consumption monitoring	MTRANS_Walking
0	1.62	64.0	1	0	0
1	1.52	56.0	1	1	0
2	1.80	77.0	1	0	0
3	1.80	87.0	0	0	1
4	1.78	89.8	0		

```
4                                0              0

   Frequent consumption of high caloric food  \
0                                   2.766876
1                                   2.766876
2                                   2.766876
3                                   2.766876
4                                   2.766876

   Frequency of consumption of vegetables  Number of main meals  \
0                                     0.5              0.404704
1                                     1.0              0.404704
2                                     0.5              0.404704
3                                     1.0              0.404704
4                                     0.5              2.164116

   Consumption of food between meals  Consumption of water daily  \
0                           0.333333                         0.5
1                           0.333333                         1.0
2                           0.333333                         0.5
3                           0.333333                         0.5
4                           0.333333                         0.5

   Physical activity frequency  Time using technology devices  \
0                     0.000000                       0.550985
1                     1.000000                       1.092724
2                     0.666667                       0.550985
3                     0.666667                       1.092724
4                     0.000000                       1.092724

   Consumption of alcohol  Age_bin_minmax  Level of obesity
0                1.439033            0.25                 1
1                0.516552            0.25                 1
2                2.472136            0.50                 1
3                2.472136            0.75                 2
4                0.516552            0.50                 3
```

[13]:
```
# displays datas current shape/features

df.shape
```

[13]:
```
(2086, 15)
```

[14]:
```
# Searches for duplicates

df.duplicated()
```

[14]:
```
0       False
1       False
2       False
3       False
4       False
        …
2081    False
2082    False
2083    False
2084    False
2085    False
Length: 2086, dtype: bool
```

[15]:
```
# due to the vast amount of rows within the dataset I will use drop_duplicates␣
↪just to be safe

df = df.drop_duplicates()

df.shape
```

[15]:
```
(2062, 15)
```

[16]:
```
# looks for missing values within the dataset

df.isnull().values.any()
```

[16]:
```
False
```

[17]:
```
# Drops columns that are not useful

df = df.drop(columns = ['MTRANS_Walking','Time using technology devices'])

df.head()
```

[17]:

	Height	Weight	family_history_with_overweight	Calories consumption monitoring	Frequent consumption of high caloric food
0	1.62	64.0	1	0	2.766876
1	1.52	56.0	1	1	2.766876
2	1.80	77.0	1	0	2.766876
3	1.80	87.0	0	0	2.766876
4	1.78	89.8	0	0	2.766876

```
   Frequency of consumption of vegetables  Number of main meals  \
0                                     0.5              0.404704
1                                     1.0              0.404704
2                                     0.5              0.404704
3                                     1.0              0.404704
4                                     0.5              2.164116

   Consumption of food between meals  Consumption of water daily  \
0                           0.333333                         0.5
1                           0.333333                         1.0
2                           0.333333                         0.5
3                           0.333333                         0.5
4                           0.333333                         0.5

   Physical activity frequency  Consumption of alcohol  Age_bin_minmax  \
0                     0.000000                1.439033            0.25
1                     1.000000                0.516552            0.25
2                     0.666667                2.472136            0.50
3                     0.666667                2.472136            0.75
4                     0.000000                0.516552            0.50

   Level of obesity
0                 1
1                 1
2                 1
3                 2
4                 3
```

[18]:
```
# checks datatypes to see if changes are necessary

df.dtypes
```

[18]:
```
Height                                    float64
Weight                                    float64
family_history_with_overweight              int64
Calories consumption monitoring             int64
Frequent consumption of high caloric food  float64
Frequency of consumption of vegetables     float64
Number of main meals                       float64
Consumption of food between meals          float64
Consumption of water daily                 float64
Physical activity frequency                float64
Consumption of alcohol                     float64
Age_bin_minmax                             float64
Level of obesity                            int64
dtype: object
```

[19]:
```
# Renames the columns again to make it seem less messy

df = df.rename(columns = {'Age_bin_minmax' : 'Age_Range',␣
 ↪'family_history_with_overweight' : 'Family_History',
                           'Consumption of alcohol' : 'Alchohol_Consumption',␣
 ↪'Calories consumption monitoring' : 'Tracks_Calories',
                           'Frequent consumption of high caloric food' :␣
 ↪'Junk_Food_Frequency', 'Number of main meals' : 'Average_Meals',
                          'Level of obesity' : 'Obesity_Level', 'Physical␣
 ↪activity frequency' : 'Exercise_Frequency',
                          'Frequency of consumption of vegetables' :␣
 ↪'Vegetable_Consumption', 'Consumption of water daily' : 'Water_Consumption',
                           'Consumption of food between meals' :␣
 ↪'Snack_Consumption'})

df.head()
```

[19]:

	Height	Weight	Family_History	Tracks_Calories	Junk_Food_Frequency	\
0	1.62	64.0	1	0	2.766876	
1	1.52	56.0	1	1	2.766876	
2	1.80	77.0	1	0	2.766876	
3	1.80	87.0	0	0	2.766876	
4	1.78	89.8	0	0	2.766876	

	Vegetable_Consumption	Average_Meals	Snack_Consumption	Water_Consumption	\
0	0.5	0.404704	0.333333	0.5	
1	1.0	0.404704	0.333333	1.0	
2	0.5	0.404704	0.333333	0.5	
3	1.0	0.404704	0.333333	0.5	
4	0.5	2.164116	0.333333	0.5	

	Exercise_Frequency	Alchohol_Consumption	Age_Range	Obesity_Level
0	0.000000	1.439033	0.25	1
1	1.000000	0.516552	0.25	1
2	0.666667	2.472136	0.50	1
3	0.666667	2.472136	0.75	2
4	0.000000	0.516552	0.50	3

[20]:
```
# Re orders the columns for organizational purposes

columns_order = ['Height', 'Weight', 'Age_Range', 'Obesity_Level',␣
 ↪'Tracks_Calories', 'Average_Meals',
                 'Exercise_Frequency', 'Junk_Food_Frequency',␣
 ↪'Water_Consumption', 'Alchohol_Consumption', 'Vegetable_Consumption',
                 'Snack_Consumption']
```

```
df = df.reindex(columns = columns_order)

df.head(10)
```

[20]:

	Height	Weight	Age_Range	Obesity_Level	Tracks_Calories	Average_Meals	\
0	1.62	64.0	0.25	1	0	0.404704	
1	1.52	56.0	0.25	1	1	0.404704	
2	1.80	77.0	0.50	1	0	0.404704	
3	1.80	87.0	0.75	2	0	0.404704	
4	1.78	89.8	0.50	3	0	2.164116	
5	1.62	53.0	1.00	1	0	0.404704	
6	1.50	55.0	0.50	1	0	0.404704	
7	1.64	53.0	0.50	1	0	0.404704	
8	1.78	64.0	0.75	1	0	0.404704	
9	1.72	68.0	0.50	1	0	0.404704	

	Exercise_Frequency	Junk_Food_Frequency	Water_Consumption	\
0	0.000000	2.766876	0.5	
1	1.000000	2.766876	1.0	
2	0.666667	2.766876	0.5	
3	0.666667	2.766876	0.5	
4	0.000000	2.766876	0.5	
5	0.000000	0.361418	0.5	
6	0.333333	0.361418	0.5	
7	1.000000	2.766876	0.5	
8	0.333333	0.361418	0.5	
9	0.333333	0.361418	0.5	

	Alchohol_Consumption	Vegetable_Consumption	Snack_Consumption
0	1.439033	0.5	0.333333
1	0.516552	1.0	0.333333
2	2.472136	0.5	0.333333
3	2.472136	1.0	0.333333
4	0.516552	0.5	0.333333
5	0.516552	0.5	0.333333
6	0.516552	1.0	0.333333
7	0.516552	0.5	0.333333
8	2.472136	1.0	0.333333
9	1.439033	0.5	0.333333

Summary of Data Preperation Phase:

My first step was to drop any duplicate values from the dataset. The shape function total of 2086 and then 2062 before and after using the drop_duplicates function shows that it was successful. I then decided to do the same process for null/NaN values but the isnull().values.any() function displayed that the dataset does not have any. The next step was drop the columns that I did not believe to be useful in the MTRANS_Walking & Time using technology devices columns. The reason that I chose to remove the time using technology column was that I believed it was not

possible to determine if it was a positive or negative & if that data is skewed by a persons job. Someone who works in analytics could be at a computer for 8 hours and then use a fitness app at the gym for another 1-2 hours after work but the differential is not shown. I chose to remove the walking column as I believed that the exercise column was a better version of this column and that having both was not necessary. I then checked the datatypes and was happy with the current state being that there were no str values. The final steps were to rename and reorder the columns/index so that the table is easier to interpret/cleaner. I chose the index order for 'personal factors' - 'frequency columns' - 'consumption columns' I feel comfortable with the current state of the dataset and look forward to building upon it during the next milestones.

[22]:
```
### Milestone 3 - Model Building and Evaluation, Basic Linear Regression Model
```

[46]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

[48]:
```
# Splits the df into test and train sets with the target variable being

X = df.drop('Exercise_Frequency', axis = 1)
y = df['Exercise_Frequency']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

target_model = LinearRegression()
target_model.fit(X_train, y_train)
```

[48]:
```
LinearRegression()
```

[50]:
```
# Displays the relationship of the target variable with the rest of the columns

pd.DataFrame(target_model.coef_, X.columns, columns = ['Coef'])
```

[50]:

	Coef
Height	0.593043
Weight	0.004040
Age_Range	-0.084821
Obesity_Level	-0.071594
Tracks_Calories	0.056632
Average_Meals	0.003714
Junk_Food_Frequency	0.036974
Water_Consumption	0.100805
Alchohol_Consumption	0.072680
Vegetable_Consumption	0.058238
Snack_Consumption	-0.085348

[52]:
```
# Displays the metrics of each set in regards to rmse, absolute error & r2 score
```

```
test_predictions = target_model.predict(X_test)
train_predictions = target_model.predict(X_train)

print('Test Metrics:')
print('R2', metrics.r2_score(y_test, test_predictions))
print('RMSE', metrics.mean_squared_error(y_test, test_predictions, squared =␣
 ↪False))
print('MAE', metrics.mean_absolute_error(y_test, test_predictions))

print('\nTrain Metrics:')
print('R2', metrics.r2_score(y_train, train_predictions))
print('RMSE', metrics.mean_squared_error(y_train, train_predictions, squared =␣
 ↪False))
print('MAE', metrics.mean_absolute_error(y_train, train_predictions))
```

```
Test Metrics:
R2 0.2300899178266015
RMSE 0.24245632046371518
MAE 0.19388668722713734

Train Metrics:
R2 0.2004605432752914
RMSE 0.2562594621754153
MAE 0.20507122238910566
```

```
/home/65c9f9d3-081c-46ec-823e-52cd3305b641/.local/lib/python3.11/site-
packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is
deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean
squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/home/65c9f9d3-081c-46ec-823e-52cd3305b641/.local/lib/python3.11/site-
packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is
deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean
squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Model Review:

For this model I decided to make the target variable Exercise_Frequency as it is the only column that tracks an individuals level of physical activity. This will help the further exploration of this topic to better understand the impact of things such as heart rate and sedimentary lifestyles tend to have on these calculations. This will also help to answer the age old question of what is truly most important, Diet or exercise? When looking at the models results the target variable had the lowest relationship with the Age, Obesity and Snack variables. This allows us to assume that people who do not exercise 'frequently' are more likely to be higher in age, labeled as 'obese' and consume more snacks than others. I believed that understanding the coefficients/relations that this value has with the rest of the columns. The range in values for the R2 score (0.03) indicates that there is a high level of accurate predictability in regards to the independent/target variable. The range in values for the calculated rmse (0.012) and MAE (0.012) indicate an extremely high level

of predictive value. Using these metrics I believe it is safe to interpret the results displayed in the Coef dataframe graphic which displays a negative relationship with the obesity level column.