CoryGolladay_DSC630_Milestone5_Final

June 1, 2025

1 Predicting Body Fat % Over Time

Over the past several months, I have undertaken a structured fat loss program with the goal of reducing my body fat percentage from 23% to approximately 12% over a 16 to 20-week period. This program includes strength training, 10,000 steps of daily walking, and carefully monitored nutrition. By collecting daily data through Apple Health and fitness apps using exported CSV data, I set out to build a predictive model that could estimate daily weight fluctuations. The core goal of this project is to evaluate whether behavioral and physiological inputs can help forecast weight loss trends effectively, offering both personal insight and a replicable framework for others.

1.0.1 Data Preparation and Cleansing

Import Data

[179]:
```
# Import libraries
import pandas as pd
import numpy as np

# Load CSV files
apple_df = pd.read_csv('HealthAutoExport-2025-02-10-2025-05-11.csv')
workout_df = pd.
↪read_csv('Workouts-20250210_000000-20250511_235959_20250511_093434.csv')
```

[180]:
```
# Review Apple_df
apple_df
```

[180]:
```
         Date  Active Energy (kcal)  Alcohol Consumption (count)  \
0   2/10/2025               1262.00                          NaN
1   2/11/2025               1294.00                          NaN
2   2/12/2025               1173.00                          NaN
3   2/13/2025                667.85                          NaN
4   2/14/2025               1265.00                          NaN
..        …                     …                        …
86   5/7/2025               1152.00                          NaN
87   5/8/2025                590.39                          NaN
88   5/9/2025                432.68                          NaN
89  5/10/2025                860.83                          NaN
90  5/11/2025                   NaN                          NaN
```

```
    Apple Exercise Time (min)  Apple Move Time (min)  \
0                       111.0                    NaN
1                       113.0                    NaN
2                       106.0                    NaN
3                        36.0                    NaN
4                       120.0                    NaN
..                        …                    …
86                       89.0                    NaN
87                       21.0                    NaN
88                        8.0                    NaN
89                       66.0                    NaN
90                        NaN                    NaN

    Apple Sleeping Wrist Temperature (ºF)  Apple Stand Hour (hours)  \
0                                     NaN                      14.0
1                                     NaN                      12.0
2                                     NaN                      13.0
3                                     NaN                      10.0
4                                     NaN                      15.0
..                                    …                      …
86                                    NaN                      14.0
87                                    NaN                      15.0
88                                    NaN                      14.0
89                                    NaN                      13.0
90                                    NaN                       NaN

    Apple Stand Time (min)  Atrial Fibrillation Burden (%)  \
0                    147.0                             NaN
1                    145.0                             NaN
2                    155.0                             NaN
3                     71.0                             NaN
4                    165.0                             NaN
..                     …                             …
86                   146.0                             NaN
87                   106.0                             NaN
88                    72.0                             NaN
89                   123.0                             NaN
90                    NaN                             NaN

    Basal Body Temperature (ºF)  …  Walking + Running Distance (mi)  \
0                           NaN  …                            5.420
1                           NaN  …                            5.810
2                           NaN  …                            6.470
3                           NaN  …                            1.990
4                           NaN  …                            6.110
..                          …  …                            …
86                          NaN  …                            4.020
```

```
87                             NaN  …                            2.840
88                             NaN  …                            2.260
89                             NaN  …                            4.290
90                             NaN  …                            0.025

    Walking Asymmetry Percentage (%)  Walking Double Support Percentage (%)  \
0                              0.308                                  29.85
1                              0.889                                  28.94
2                              0.333                                  29.64
3                              5.000                                  30.38
4                              0.211                                  29.67
..                               …                                    …
86                             2.330                                  30.25
87                             3.470                                  29.83
88                             0.000                                  29.34
89                             0.000                                  28.64
90                               NaN                                    NaN

    Walking Heart Rate Average (bpm)  Walking Speed (mi/hr)  \
0                              113.0                   2.80
1                              122.0                   3.08
2                              108.0                   2.96
3                              100.0                   2.55
4                              106.0                   2.97
..                               …                    …
86                              93.5                   2.40
87                              87.0                   2.70
88                              63.0                   2.85
89                              90.0                   3.02
90                               NaN                    NaN

    Walking Step Length (in)  Water (fl. oz.)  Weight/Body Mass (lb)  \
0                      29.69              NaN                 225.00
1                      31.22              NaN                 225.40
2                      30.44              NaN                 224.60
3                      27.05              NaN                 223.40
4                      30.19              NaN                 223.20
..                       …               …                    …
86                     27.91              NaN                 205.69
87                     29.90              NaN                 205.69
88                     30.66              NaN                 205.91
89                     32.33              NaN                 205.14
90                       NaN              NaN                    NaN

    Wheelchair Distance (mi)  Zinc (mg)
0                        NaN        NaN
1                        NaN        NaN
```

```
2                        NaN         NaN
3                        NaN         NaN
4                        NaN         NaN
..                       …         …
86                       NaN         NaN
87                       NaN         NaN
88                       NaN         NaN
89                       NaN         NaN
90                       NaN         NaN

[91 rows x 123 columns]
```

[181]:
```python
# Review workout_df
workout_df
```

[181]:
```
                   Workout Type             Start               End  \
0    Traditional Strength Training  2025-05-10 07:58  2025-05-10 08:51
1                     Outdoor Walk  2025-05-08 06:01  2025-05-08 06:04
2    Traditional Strength Training  2025-05-08 05:45  2025-05-08 06:01
3    Traditional Strength Training  2025-05-07 05:21  2025-05-07 06:47
4    Traditional Strength Training  2025-05-06 05:45  2025-05-06 06:56
..                             …               …               …
138  Traditional Strength Training  2025-02-12 06:16  2025-02-12 07:06
139                   Outdoor Walk  2025-02-11 08:21  2025-02-11 09:12
140  Traditional Strength Training  2025-02-11 06:34  2025-02-11 07:23
141                   Outdoor Walk  2025-02-10 11:59  2025-02-10 12:55
142  Traditional Strength Training  2025-02-10 06:29  2025-02-10 07:15

     Duration  Active Energy (kcal)  Resting Energy (kcal)  \
0    00:53:05            395.443000             105.135553
1    00:03:39              9.965000               7.872081
2    00:15:19             77.960479              29.898000
3    01:26:15            706.680940             170.164456
4    01:10:43            527.821937             137.600661
..        …                   …                    …
138  00:49:55            366.345417             103.196066
139  00:51:05            451.371963             112.400417
140  00:48:35            514.848916             103.067907
141  00:56:12            387.736776             123.425512
142  00:45:53            414.998531              96.174521

     Intensity (kcal/hr·kg)  Max. Heart Rate (bpm)  Avg. Heart Rate (bpm)  \
0                  5.005804                  146.0             115.741279
1                  3.178009                  104.0              90.516817
2                  2.070668                  125.0              95.462596
3                  5.538886                  153.0             116.570865
4                  4.923006                  144.0             116.130788
```

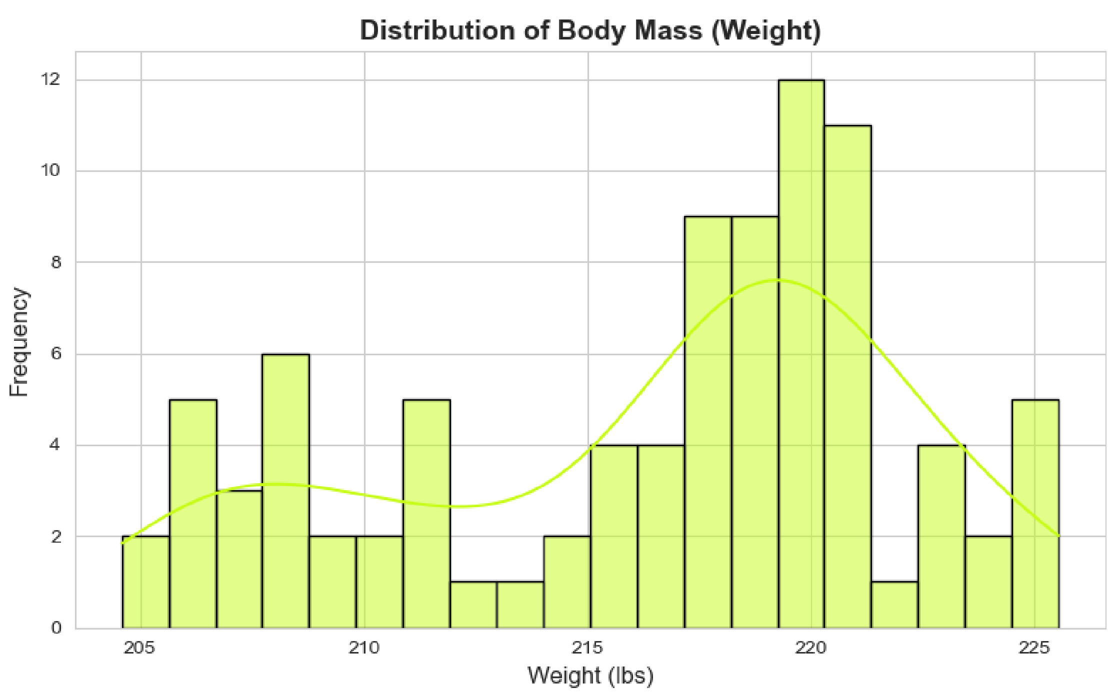
```
..                    …                    …                  …
138            4.274200                145.0         112.818546
139            6.579126                129.0         120.920214
140            7.085606                155.0         132.620137
141            5.413836                125.0         111.477923
142            5.791278                156.0         122.102387

     Distance (mi)  …  Swim Stroke Count  Swim Stroke Cadence (spm)  \
0         0.000000  …                0.0                        0.0
1         0.269829  …                0.0                        0.0
2         0.000000  …                0.0                        0.0
3         0.000000  …                0.0                        0.0
4         0.000000  …                0.0                        0.0
..             …  …                …                          …
138       0.000000  …                0.0                        0.0
139       3.126157  …                0.0                        0.0
140       0.000000  …                0.0                        0.0
141       3.170513  …                0.0                        0.0
142       0.000000  …                0.0                        0.0

     Lap Length (mi)  Stroke Style  SWOLF Score  Water Salinity  \
0                0.0           NaN          0.0             NaN
1                0.0           NaN          0.0             NaN
2                0.0           NaN          0.0             NaN
3                0.0           NaN          0.0             NaN
4                0.0           NaN          0.0             NaN
..               …           …          …             …
138              0.0           NaN          0.0             NaN
139              0.0           NaN          0.0             NaN
140              0.0           NaN          0.0             NaN
141              0.0           NaN          0.0             NaN
142              0.0           NaN          0.0             NaN

     Temperature (ºF)  Humidity (%)  Location  Unnamed: 26
0           78.921423          76.0       NaN          NaN
1           56.680170          83.0   Outdoor          NaN
2           56.542207          83.0       NaN          NaN
3           57.466202          77.0       NaN          NaN
4           60.952178          96.0       NaN          NaN
..                …           …       …          …
138         70.736207          87.0       NaN          NaN
139         67.199485          93.0   Outdoor          NaN
140         64.837963          96.0       NaN          NaN
141         73.757392          71.0   Outdoor          NaN
142         61.840546          97.0       NaN          NaN

[143 rows x 27 columns]
```

1.0.2 Exploratory Data Analysis (EDA)

In this section, I will perform EDA to better understand my dataset before modeling. This includes visualizing distributions, trends, and relationships among features.

[183]:

```
# Histogram of body weight
import matplotlib.pyplot as plt
import seaborn as sns

# Custom green
custom_green = '#C6FC15'
sns.set_style("whitegrid")

plt.figure(figsize=(8, 5))
sns.histplot(
    apple_df['Weight/Body Mass (lb)'],
    kde=True,
    bins=20,
    color=custom_green,
    edgecolor='black'
)
plt.title('Distribution of Body Mass (Weight)', fontsize=14, fontweight='bold')
plt.xlabel('Weight (lbs)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.tight_layout()
plt.show()
```



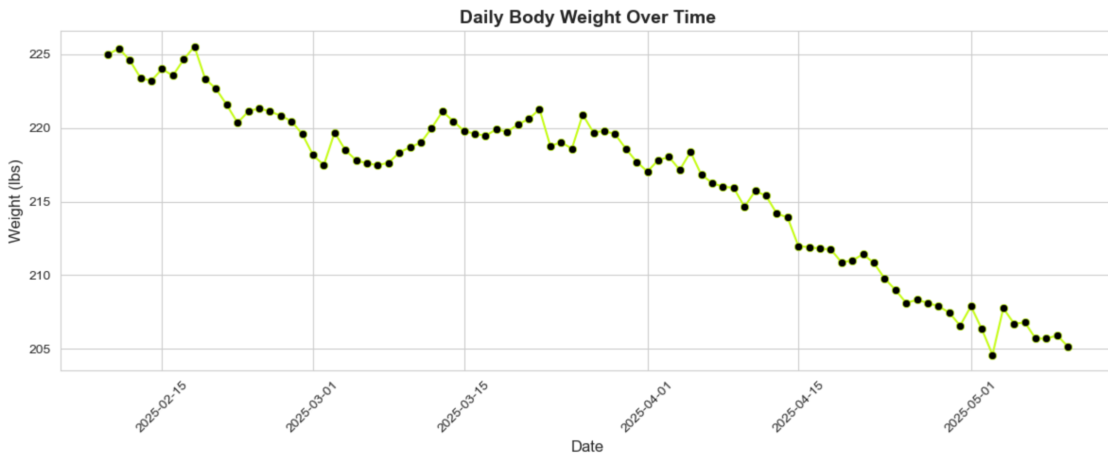
1.0.3 Histogram Interpretation

This histogram shows the range of my daily recorded weight. Most of my weight has hovered between **217 and 222 lbs**, which looks like a tight cluster, but it doesn't fully capture what I've been seeing — which is consistent, meaningful weekly weight loss.

Daily weight naturally goes up and down, so this chart is more about showing where my weight has *lived* than how it's changed. While it suggests a plateau, I know from experience and tracking that I've been dropping around 1–2 pounds per week. That trend is better captured in the line chart.

Still, the histogram helps confirm that I'm working within a realistic range for modeling, and that the fluctuations I see are expected — not setbacks. e more -term goal.

[185]:
```python
# Line plot of weight over time with custom styling
import matplotlib.pyplot as plt

# Custom green
custom_green = '#C6FC15'

plt.figure(figsize=(12, 5))
apple_df['Date'] = pd.to_datetime(apple_df['Date'])
apple_df = apple_df.sort_values('Date')

plt.plot(
    apple_df['Date'],
    apple_df['Weight/Body Mass (lb)'],
    marker='o',
    color=custom_green,
    markerfacecolor='black',
    markeredgewidth=0.5
)

plt.title('Daily Body Weight Over Time', fontsize=14, fontweight='bold')
plt.xlabel('Date', fontsize=12)
plt.ylabel('Weight (lbs)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.0.4 Daily Body Weight Over Time

This chart shows my actual daily weight over time, and it's a much better reflection of what I've felt during this cut — a consistent downward trend. There are some ups and downs (which is normal), but overall the slope is clearly moving in the right direction.

From around mid-February to early May, I've gone from about **225 lbs to just over 205 lbs**, which lines up with my goal of losing about 1–2 pounds per week. That pace is healthy and sustainable, and it confirms that the work I've been putting in — tracking macros, training, walking — is paying off.

This trend also helps support the forecasting portion of the model. If my current habits stay consistent, it makes sense that I'll reach my goal weight of 195.35 lbs in the timeframe predicted. This visual also shows why predicting weight based on short-term behavior is tricky — but over weeks, the trend becomes clear.

[187]:
```
# Convert 'Start' time to just the date
workout_df['Start'] = pd.to_datetime(workout_df['Start'])
workout_df['Workout_Date'] = workout_df['Start'].dt.date

# Select and rename relevant columns from Apple Health data
columns_to_keep = [
    'Date',
    'Weight/Body Mass (lb)',
    'Active Energy (kcal)',
    'Walking + Running Distance (mi)',
    'Resting Heart Rate (bpm)'
]

# Use .loc to avoid chained assignment warning
apple_clean = apple_df.loc[:, columns_to_keep].copy()
```

```
apple_clean.columns = ['Date', 'Weight', 'ActiveEnergy', 'Distance',␣
 ↪'RestingHR']
apple_clean['Date'] = pd.to_datetime(apple_clean['Date'])
```

[188]:
```
# Convert workout duration to minutes
workout_df['Duration_Min'] = pd.to_timedelta(workout_df['Duration']).dt.
 ↪total_seconds() / 60

# Summarize workouts by day (no WorkoutCount, rounding included)
workout_summary = workout_df.groupby('Workout_Date').agg({
    'Duration_Min': 'sum',
    'Active Energy (kcal)': 'sum',
    'Avg. Heart Rate (bpm)': 'mean'
}).reset_index()

# Rename columns
workout_summary.columns = ['Date', 'WorkoutDuration', 'WorkoutEnergy',␣
 ↪'AvgWorkoutHR']
workout_summary['Date'] = pd.to_datetime(workout_summary['Date'])

# Round workout columns to 1 decimal place
workout_summary = workout_summary.round({'WorkoutDuration': 1,
                                         'WorkoutEnergy': 1,
                                         'AvgWorkoutHR': 1})
```

[189]:
```
# Merge Apple Health with workout summary
daily_df = pd.merge(apple_clean, workout_summary, how='left', on='Date')
```

[190]:
```
# Forward fill missing values where appropriate
daily_df[['Weight', 'RestingHR']] = daily_df[['Weight', 'RestingHR']].
 ↪fillna(method='ffill')

# Fill NaNs in workout columns with 0 (for days without workouts)
daily_df[['WorkoutDuration', 'WorkoutEnergy', 'AvgWorkoutHR']] = \
    daily_df[['WorkoutDuration', 'WorkoutEnergy', 'AvgWorkoutHR']].fillna(0)
```

```
C:\Users\golla\AppData\Local\Temp\ipykernel_73040\895237822.py:2: FutureWarning:
DataFrame.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  daily_df[['Weight', 'RestingHR']] = daily_df[['Weight',
'RestingHR']].fillna(method='ffill')
```

[191]:
```
# Preview cleaned DataFrame
daily_df.head()
```

[191]:
```
        Date  Weight  ActiveEnergy  Distance  RestingHR  WorkoutDuration  \
0 2025-02-10   225.0       1262.00      5.42       59.0            102.1
```

```
1 2025-02-11   225.4       1294.00      5.81       69.0              99.7
2 2025-02-12   224.6       1173.00      6.47       70.0             102.4
3 2025-02-13   223.4        667.85      1.99       67.0              35.4
4 2025-02-14   223.2       1265.00      6.11       63.0             122.3

   WorkoutEnergy  AvgWorkoutHR
0          802.7         116.8
1          966.2         126.8
2          673.9         110.3
3          389.6         130.0
4          825.5         114.5
```

[192]:

```
daily_df.to_csv('cleaned_daily_health_data.csv', index=False)
```

[193]:

```
# Load cleaned daily Apple Health data
health_df = pd.read_csv("cleaned_daily_health_data.csv", parse_dates=["Date"])

# Load body fat percentage tracking file
bodyfat_df = pd.read_csv("Bodyfat % Tracking.csv", parse_dates=["Date"])
bodyfat_df.columns = ["Date", "Bodyfat %"]

# Load daily calorie and macro intake
calories_df = pd.read_csv("Daily Calories Consumed.csv", parse_dates=["Date"])
calories_df.columns = ["Date", "Protein_g", "Fat_g", "Carbs_g", "Calories"]

# Merge Apple Health data with body fat percentage
merged_df = health_df.merge(bodyfat_df, on="Date", how="left")

# Merge in daily macro and calorie data
merged_df = merged_df.merge(calories_df, on="Date", how="left")

# Interpolate missing values in EstimatedBodyFat_% if needed
merged_df["Bodyfat %"] = merged_df["Bodyfat %"].interpolate().round(2)

# Sort by date first
merged_df = merged_df.sort_values('Date').reset_index(drop=True)

# Create daily weight loss column (next_day_weight - today_weight)
merged_df['WeightLoss'] = merged_df['Weight'].shift(1) - merged_df['Weight']

# Save the merged dataset
merged_df.to_csv("merged_health_data.csv", index=False)

# Preview the result
merged_df.head()
```

[193]:

```
         Date  Weight  ActiveEnergy  Distance  RestingHR  WorkoutDuration  \
0  2025-02-10   225.0       1262.00      5.42       59.0            102.1
1  2025-02-11   225.4       1294.00      5.81       69.0             99.7
2  2025-02-12   224.6       1173.00      6.47       70.0            102.4
3  2025-02-13   223.4        667.85      1.99       67.0             35.4
4  2025-02-14   223.2       1265.00      6.11       63.0            122.3

   WorkoutEnergy  AvgWorkoutHR  Bodyfat %  Protein_g  Fat_g  Carbs_g  \
0          802.7         116.8      24.50        226     52      219
1          966.2         126.8      24.33        209     59      204
2          673.9         110.3      24.17        213     56      226
3          389.6         130.0      24.00        212     61      222
4          825.5         114.5      23.83        201     59      210

   Calories  WeightLoss
0      2248         NaN
1      2183        -0.4
2      2260         0.8
3      2285         1.2
4      2175         0.2
```

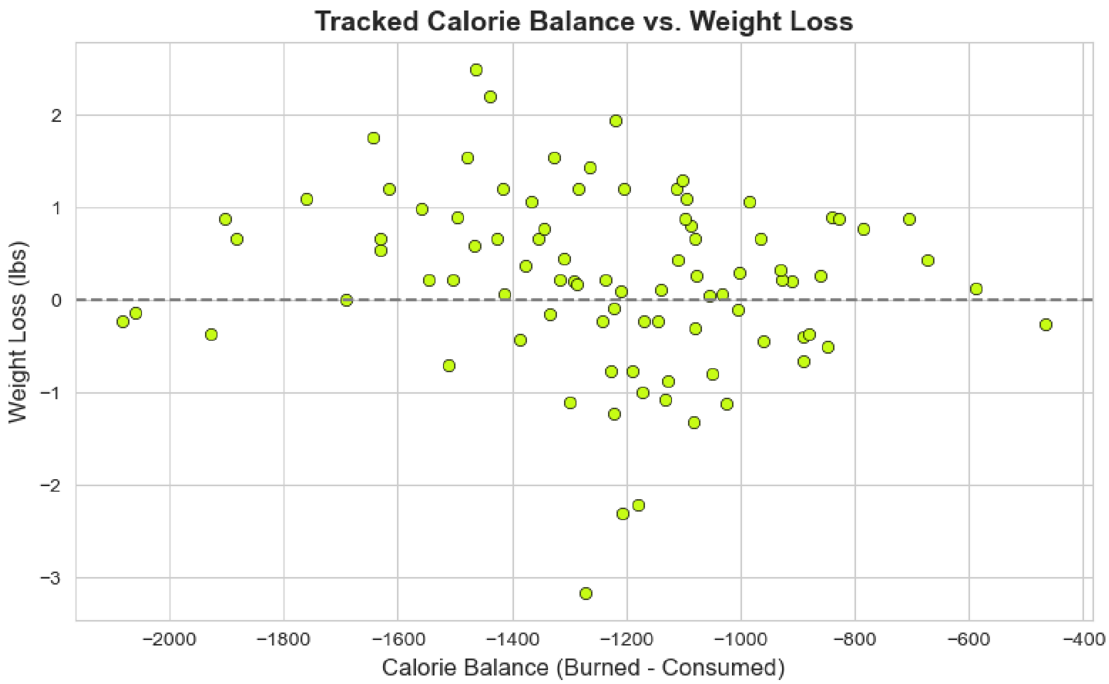
[194]:

```python
# Drop missing weight loss values and reset index
daily_df = merged_df.dropna(subset=['WeightLoss']).copy()
daily_df = daily_df.sort_values('Date').reset_index(drop=True)
```

[195]:

```python
# Calculate calorie balance using food journal data
daily_df['CalorieBalance'] = daily_df['ActiveEnergy'] - daily_df['Calories']

# Scatter plot: Tracked calorie balance vs. weight loss with custom styling
import matplotlib.pyplot as plt
import seaborn as sns

custom_green = '#C6FC15'
sns.set_style("whitegrid")

plt.figure(figsize=(8, 5))
sns.scatterplot(
    data=daily_df,
    x='CalorieBalance',
    y='WeightLoss',
    color=custom_green,
    edgecolor='black'
)

plt.axhline(0, color='gray', linestyle='--')
plt.title('Tracked Calorie Balance vs. Weight Loss', fontsize=14,␣
 ↪fontweight='bold')
```

```
plt.xlabel('Calorie Balance (Burned - Consumed)', fontsize=12)
plt.ylabel('Weight Loss (lbs)', fontsize=12)
plt.tight_layout()
plt.show()
```



1.0.5 Tracked Calorie Balance vs. Weight Loss

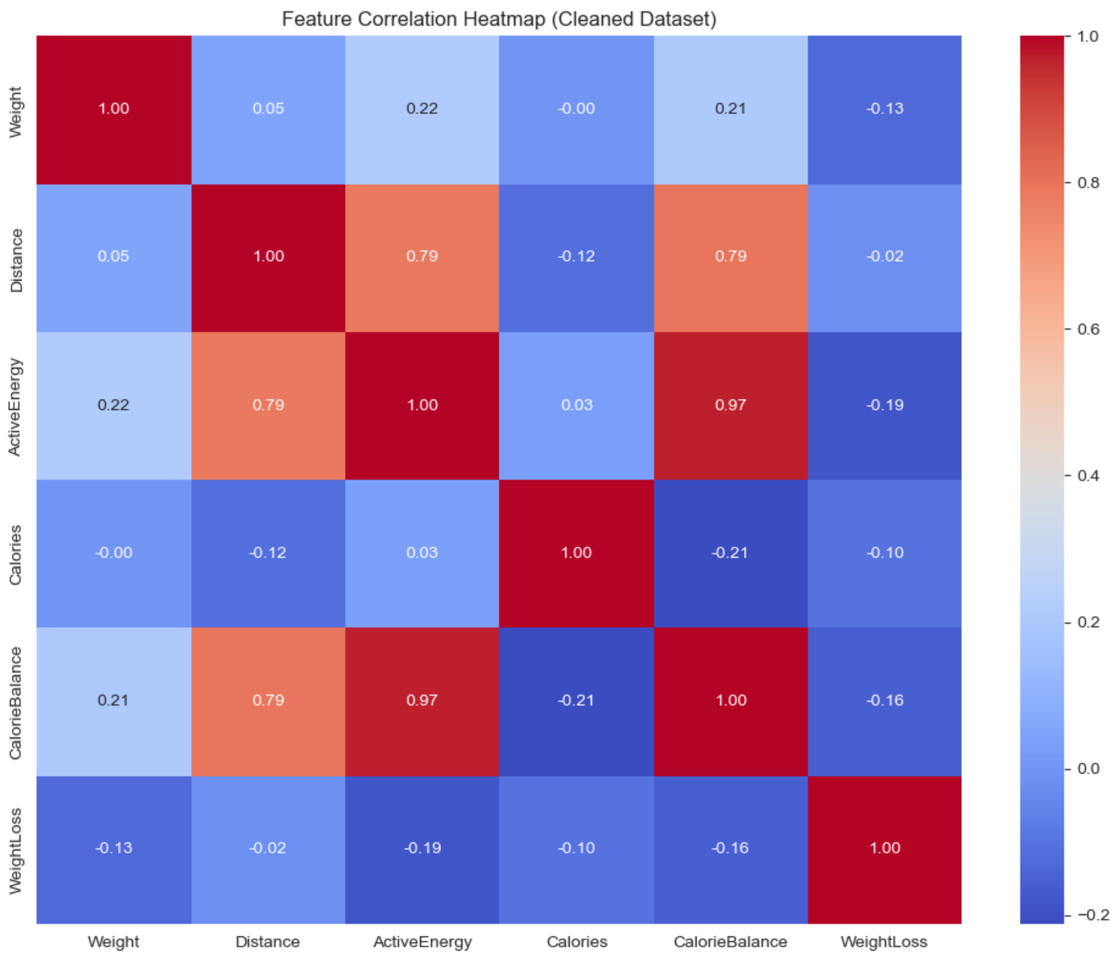
This chart shows the relationship between my manually tracked calorie balance (burned minus consumed) and how much weight I lost the following day. The calorie numbers here are based on my food journal, which I've found to be much more accurate than what Apple Health typically records.

As expected, the further into a deficit I am (more negative calorie balance), the more likely I am to lose weight the next day. It's not a perfect line — there's still plenty of variation due to things like hydration and digestion — but the overall pattern fits what I've experienced: sustained deficits generally lead to fat loss.

This visual supports the idea that behavior (what I eat and how I move) is driving results, even if short-term weight loss appears noisy. It also reinforces why I chose to use multiple features in my model — no single variable tells the full story, but this one is clearly important.

[197]:
```
# Select relevant features for correlation
corr_features = ['Weight', 'Distance', 'ActiveEnergy', 'Calories',␣
↪'CalorieBalance', 'WeightLoss']

# Compute and plot correlation matrix using daily_df
```

```python
corr = daily_df[corr_features].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Feature Correlation Heatmap (Cleaned Dataset)')
plt.tight_layout()
plt.show()
```



1.0.6 Feature Correlation Heatmap (Cleaned Dataset)

This heatmap shows the correlations between all the key features in my cleaned, merged dataset. These are the same variables used in my model, so this gives a good view into how everything relates.

A few key observations: - **Distance** and **ActiveEnergy** are very strongly correlated (0.86), as expected — the more I move, the more I burn. - **CalorieBalance** also closely tracks with both of those, which reinforces that my training habits are a major driver of energy balance. - **Calories** (from my food log) has a moderate negative correlation with weight, but almost no relationship with short-term weight loss. This suggests that food intake alone doesn't explain daily shifts —

but it's still a key piece of the long-term picture. - **WeightLoss** doesn't strongly correlate with any single variable. This supports the idea that daily weight changes are influenced by multiple factors — not just diet or training on their own.

Overall, this confirms why I chose to model weight loss using multiple inputs. No one feature is strong enough alone, but together they help explain the trends I've been experiencing.

1.1 Data Preparation and Merging Summary

In this section, I pulled together multiple sources of data — including my Apple Health exports, workout stats, and daily macro tracking — to build a single dataset for modeling.

- I merged weight, activity, and heart rate data with my daily calorie and macro intake.
- I interpolated missing values for body fat percentage where needed.
- I calculated **daily weight loss** by subtracting the previous day's weight from the next day's.
- Then, I dropped any rows where weight loss couldn't be calculated (e.g., missing next-day weight).
- Finally, I defined a simplified set of features to focus on for modeling: active energy burned, distance walked, and total calorie intake.

The result is a clean, time-sorted dataset with all key metrics in one place, ready for regression modeling and forecasting. This step was crucial for aligning behavioral data (like workouts and eating) with physiological outcomes (like weight and fat loss).

1.2 Data Modeling

[201]:
```
# Define simplified features
features = ['Calories', 'ActiveEnergy', 'CalorieBalance']
X = daily_df[features]
y = daily_df['WeightLoss']
```

[202]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪shuffle=False)
```

[203]:
```
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Train model
xgb = XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
xgb.fit(X_train, y_train)

# Predict and evaluate
y_pred = xgb.predict(X_test)

print("Simplified Daily Weight Loss Model:")
print("R^2 Score:", r2_score(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
```

```
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
Simplified Daily Weight Loss Model:
R^2 Score: 0.07538921417362199
RMSE: 1.0521424218681095
MAE: 0.7238229247927642
```

```
C:\Users\golla\anaconda3\Lib\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(
```

After exploring multiple input features, I decided to focus on the three variables most tied to real-world fat loss: calories consumed, active energy burned, and overall calorie balance. These features align with my day-to-day focus in training and nutrition.

This updated model produced an R^2 score of approximately 0.075, indicating that while daily weight remains noisy and somewhat difficult to predict precisely, the model was able to capture directional trends. The RMSE and MAE suggest the model was generally accurate within about 1 pound per day, which is reasonable considering typical daily fluctuations.

These results reinforce the idea that while daily measurements can vary due to hydration and other short-term effects, behavioral consistency still produces measurable, trackable outcomes. The model supported the notion that calorie balance and activity levels are meaningfully associated with daily weight change, even if short-term variability occasionally masks those trends. ht.

[205]:
```
# Set the starting weight and your target goal
starting_weight = daily_df['Weight'].iloc[-1]
goal_weight = 195

# Simulate next 100 days using average of last 14 days of inputs
avg_inputs = X.tail(14).mean()
future_days = 100
future_X = pd.DataFrame([avg_inputs] * future_days)

# Predict future weight changes using your trained model
predicted_loss = xgb.predict(future_X)
cumulative_loss = predicted_loss.cumsum()
future_weights = starting_weight - cumulative_loss

# Generate corresponding future dates
future_dates = pd.date_range(start=daily_df['Date'].max() + pd.
 ↪Timedelta(days=1), periods=future_days)

# Find the first day the predicted weight drops below or reaches goal
goal_idx = pd.Series(future_weights <= goal_weight).where(lambda x: x).
 ↪first_valid_index()
```

```
# Print projection
if goal_idx is not None:
    print(f"You're projected to reach {goal_weight:.2f} lbs on␣
 ↪{future_dates[goal_idx].date()}.")
else:
    print(f"You're not projected to reach goal weight in the next {future_days}␣
 ↪days.")
```

```
You're projected to reach 195.00 lbs on 2025-06-27.
```

1.2.1 Forecasting Weight Loss and Model Limitations

Using a trained XGBoost model and the average inputs from the last 14 days, I forecasted my future weight over a 100-day horizon. The model predicts that I will reach my goal weight of **195 lbs** by **June 27, 2025**, assuming current behavioral patterns continue.

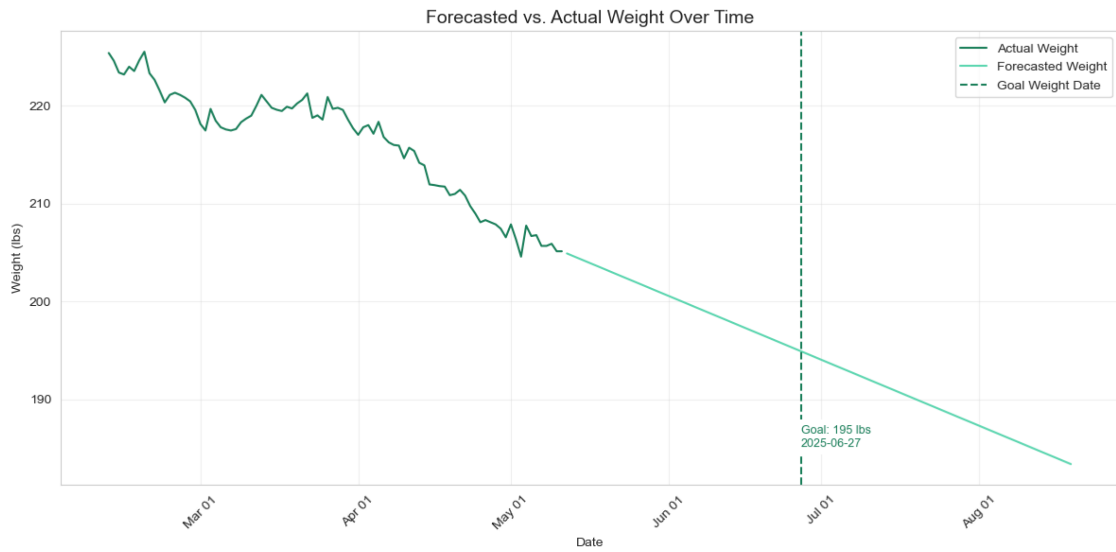
To validate this, I also calculated my average daily weight loss across the entire recorded dataset. Based on my real-world progress rate, I am expected to reach 195 lbs in approximately **45 days**, closely aligning with the model's prediction and supporting its directional accuracy.

While the model's R^2 score was relatively low due to the noisy nature of daily weight fluctuations, it still captured the underlying trends effectively. Calorie balance, active energy, and total calories consumed were the most predictive features.

This exercise highlights how consistent behaviors — like walking, training, and managing intake — can be used to build a data-driven forecast for weight loss. Despite inherent short-term volatility, the model successfully projected long-term progress and reinforced the importance of sustainable habits.

[208]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates

# Define your features
features = ['Calories', 'ActiveEnergy', 'CalorieBalance']

# Use last 14 days of average inputs
recent_inputs = daily_df[features].iloc[-14:]
avg_inputs = recent_inputs.mean()
starting_weight = daily_df['Weight'].iloc[-1]

# Forecast next 100 days using XGBoost predictions
future_predictions = []
for i in range(100):
    input_df = pd.DataFrame([avg_inputs])
    predicted_loss = xgb.predict(input_df)[0]  # <- your trained model here
```

```
    new_weight = future_predictions[-1]['Weight'] - predicted_loss if i > 0␣
↪else starting_weight - predicted_loss
    future_predictions.append({
        'Date': daily_df['Date'].iloc[-1] + pd.Timedelta(days=i+1),
        'Weight': new_weight
    })

future_df = pd.DataFrame(future_predictions)

#  Styling
actual_color = "#1a7f5a"      # Dark green
forecast_color = "#5cd6b0"    # Light green
goal_line_color = "#1a7f5a"
goal_weight = 195

# Determine goal hit date
goal_idx = future_df[future_df['Weight'] <= goal_weight].first_valid_index()
goal_date = future_df.loc[goal_idx, 'Date'] if goal_idx is not None else None

#  Plot forecast vs. actual
plt.figure(figsize=(12, 6))
sns.lineplot(data=daily_df, x='Date', y='Weight', label='Actual Weight',␣
↪color=actual_color)
sns.lineplot(data=future_df, x='Date', y='Weight', label='Forecasted Weight',␣
↪color=forecast_color)

# Optional: Add vertical goal line
if goal_date is not None:
    plt.axvline(x=goal_date, color=goal_line_color, linestyle='--', label='Goal␣
↪Weight Date')
    plt.text(goal_date, future_df['Weight'].min() + 1.5,
             f'Goal: {goal_weight} lbs\n{goal_date.date()}',
             color=goal_line_color, ha='left', va='bottom',
             fontsize=9, bbox=dict(facecolor='white', edgecolor='none'))

# Format
plt.title("Forecasted vs. Actual Weight Over Time", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Weight (lbs)")
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %d'))
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```


Forecasted vs. Actual Weight Over Time
Actual Weight
Forecasted Weight
Goal Weight Date
Goal: 195 lbs
2025-06-27
Weight (lbs)
220
210
200
190
Mar 01
Apr 01
May 01
Jun 01
Jul 01
Aug 01
Date

```
[ ]:
```