DX 602 Final Project

Introduction

In this project, you will practice the skills that you have learned throughout this module with a heavy focus on building models. Most of the problems and questions are open ended compared to your previous homeworks, and you will be asked to explain your choices. Most of them will have a particular type of solution implied, but it is up to you to figure out the details based on what you have learned in this module.

Instructions

Each problem asks you to perform build models, run a computation, or otherwise perform some analysis of the data, and usually answer some questions about the results. Make sure that your question answers are well supported by your analysis and explanations; simply stating an answer without support will earn minimal points.

Notebook cells for code and text have been added for your convenience, but feel free to add additional cells.

Example Code

You may find it helpful to refer to this GitHub repository of Jupyter notebooks for example code.

- https://github.com/bu-cds-omds/dx601-examples
- https://github.com/bu-cds-omds/dx602-examples

Any calculations demonstrated in code examples or videos may be found in these notebooks, and you are allowed to copy this example code in your homework answers.

Submission

To submit your homework, take the following steps.

1. Save and commit this notebook.
2. Push your changes to GitHub.
3. Confirm that your changes are visible in GitHub.
4. Delete the codespace to avoid wasting your free quota.

This project will be entirely manually graded. However, we may rerun some or all of your code to confirm that it works as described.

Late Policy

The normal homework late policy for OMDS does not apply to this project. Boston University requires final grades to be submitted within 72 hours of class instruction ending, so we cannot accommodate 5 days of late submissions.

However, we have delayed the due date of this project to be substantially later than necessary given its scope, and given you more days for submission with full credit than you would have had days for submission with partial credit under the homework late policy. Finally, the deadlines for DX 601 and DX 602 were coordinated to be a week apart while giving ample time for both of their projects.

Shared Imports

For this project, you are forbidden to use modules that were not loaded in this template. While other modules are handy in practice, modules that trivialize these problems interfere with our assessment of your own knowledge and skills.

If you believe a module covered in the course material (not live sessions) is missing, please check with your learning facilitator.

In [27]:
```
import math
import sys
```

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats
import sklearn.linear_model

# YOU MAY IMPORT MORE SKLEARN MODULES BASED ON YOUR CHOICES BELOW.
```

Problems

Problem 1 (5 points)

Pick one of the following data sets to analyze in this project. Load the data set, and show a random sample of 10 rows.

- Titanic disaster (PMLB copy)
- Wine Quality (PMLB - red subset only)
- Body Fat (PMLB)

The PMLB copies of the data are generally cleaner and recommended for this project, but the other links are provided to give you more context. To load the data from the PMLB Github repository, navigate to the `.tsv.gz` file in GitHub and copy the link from the "Raw" button.

If the dataset has missing data, you should drop the rows with missing data before proceeding. If the data set you choose has more than ten columns, you may limit later analysis that is requested per column to just the first ten columns.

In [29]:
```
# YOUR CODE HERE
bf = pd.read_csv('https://github.com/EpistasisLab/pmlb/raw/refs/heads/master
bf.sample(10) #show random sample of 10 rows
```

Out[29]:

	Density	Age	Weight	Height	Neck	Chest	Abdomen	Hip	T
138	1.0481	40.0	168.25	71.25	34.299999	98.300003	88.500000	98.300003	58.09
89	1.0666	48.0	176.00	73.00	36.700001	96.699997	86.500000	98.300003	60.40
134	1.0435	41.0	168.25	69.50	36.500000	98.400002	87.199997	98.400002	56.00
48	1.0678	45.0	135.75	68.50	32.799999	92.300003	83.400002	90.400002	52.00
123	1.0623	47.0	151.50	66.75	36.900002	94.000000	86.099998	95.199997	58.09
236	1.0424	62.0	191.50	72.25	40.599998	104.000000	98.199997	101.099998	59.29
146	1.0550	24.0	208.50	72.75	39.200001	102.000000	99.099998	110.099998	71.19
158	1.0704	30.0	136.50	68.75	35.900002	88.699997	76.599998	89.800003	50.09
190	1.0728	41.0	153.00	69.25	36.400002	91.400002	80.599998	92.300003	54.29
147	1.0322	25.0	206.50	69.75	40.900002	110.900002	100.500000	106.199997	68.40

I chose to work with the Body Fat dataset using the PMLB copy.

Problem 2 (10 points)

List all of the columns and describe them in your own words.

In [30]:
```
# YOUR CODE HERE
bf.columns #lists names of columns
```

Out[30]:
```
Index(['Density', 'Age', 'Weight', 'Height', 'Neck', 'Chest', 'Abdomen', 'H
ip',
       'Thigh', 'Knee', 'Ankle', 'Biceps', 'Forearm', 'Wrist', 'target'],
      dtype='object')
```

Column/Feature	Description
Density	Measured in g/cm^3 (gram per cubic centimenter). Estimated using underwater weighing to compute body volume as the difference between body weight measured in the air and body weight measured underwater. [1]
Age	Age of the individual, measured in years.
Weight	Weight of the individual, measured in lbs (pounds).
Height	Height of the indivudal, measured in inches.
Neck	Circumference of the individual's neck, measured in centimeters.
Chest	Circumference of the individual's chest, measured in centimeters.
Abdomen	Abdomen 2 [2] circumference, measured in centimeters.
Hip	Circumference of the individual's hip, measured in centimeters.
Thigh	Circumference of the individual's thigh, measured in centimeters.
Knee	Circumference of the individual's knee, measured in centimeters.
Ankle	Circumference of the individual's ankle, measured in centimeters.
Biceps	Circumference of the individual's biceps, measured in centimeters.
Forearm	Circumference of the individual's forearm, measured in centimeters.
Wrist	Circumference of the individual's wrist, measured in centimeters.
Target	Estimated percentage of body fat of the individual

[1] Source referenced:
https://wiki.socr.umich.edu/index.php/SOCR_Data_BMI_Regression

[2] Abdomen 2 circumference is measured "laterally, at the level of the iliac crests, and anteriorly, at the umbilicus" *Source Referenced: (Benhke, Wilmore 1974)*.

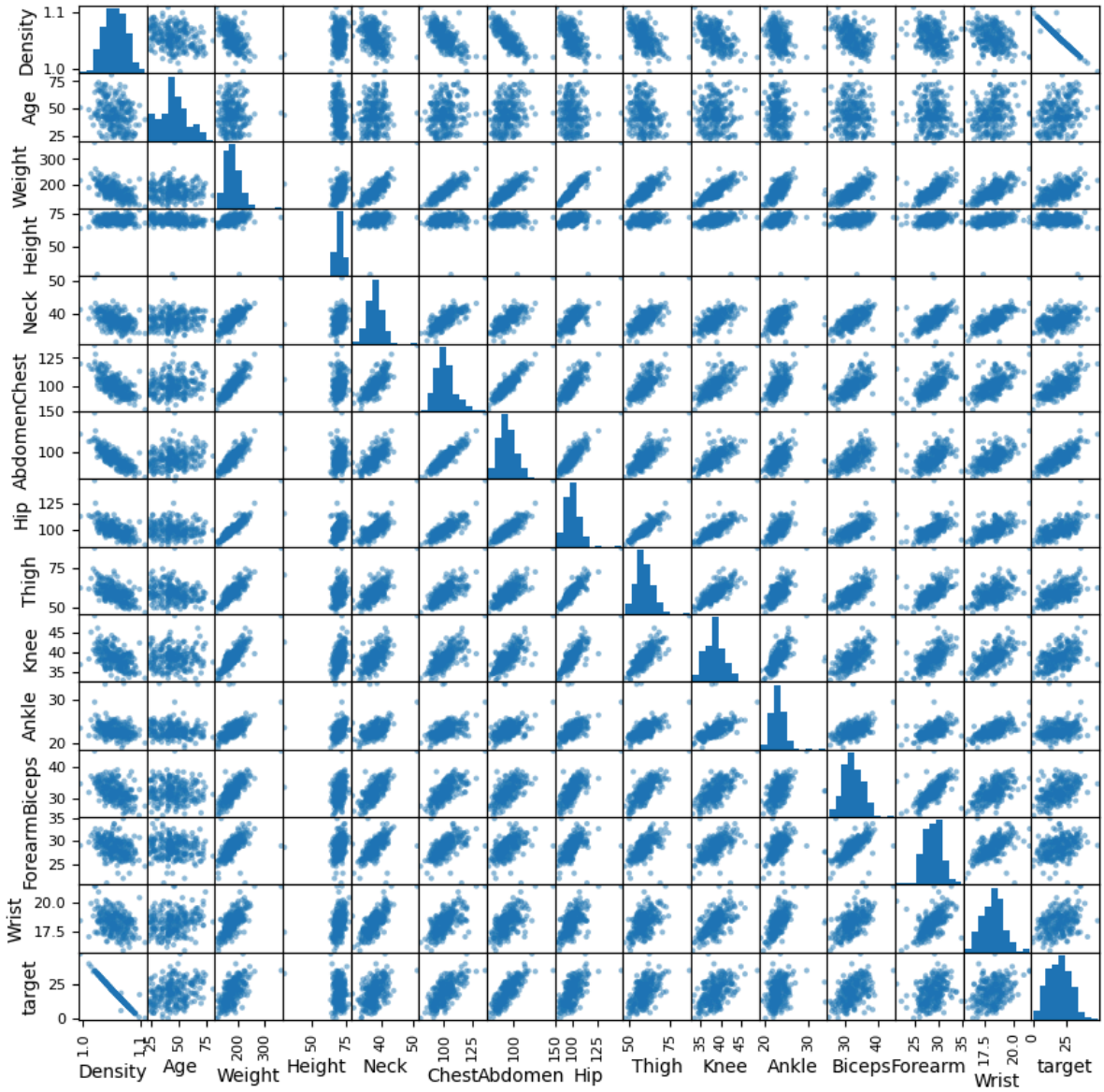
Problem 3 (50 points)

Perform an exploratory analysis of the data set. After your exploratory analysis, pick 3 individual charts that you the think were particularly interesting. Repeat those charts separately from your original analysis, and after each of those charts, explain what you thought was noteworthy.

Exploratory analysis of the Body Fat dataset:

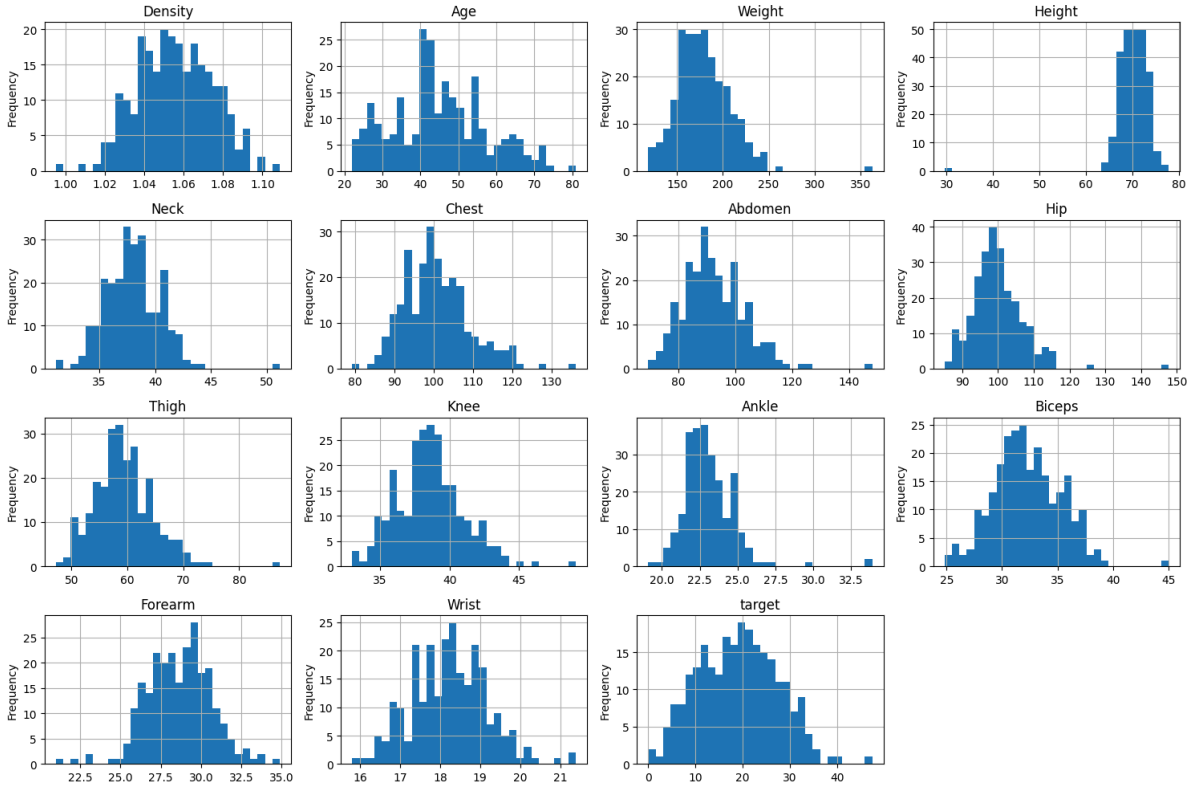
Scatter matrix of the entire dataset

In [31]:
```
# YOUR CODE HERE
#scatter matrix of the entire dataset:
bf_scatter = pd.plotting.scatter_matrix(bf, figsize=(10, 10))
```



Histogram of the entire dataset:

In [32]:
```
axes = bf.hist(figsize=(15,10), bins=30) #histogram of dataset
for ax in axes.flatten(): #.flatten() converts to 1D array, easier for itera
    ax.set_title(ax.get_title())
    ax.set_ylabel('Frequency')
plt.tight_layout()
plt.show()
```



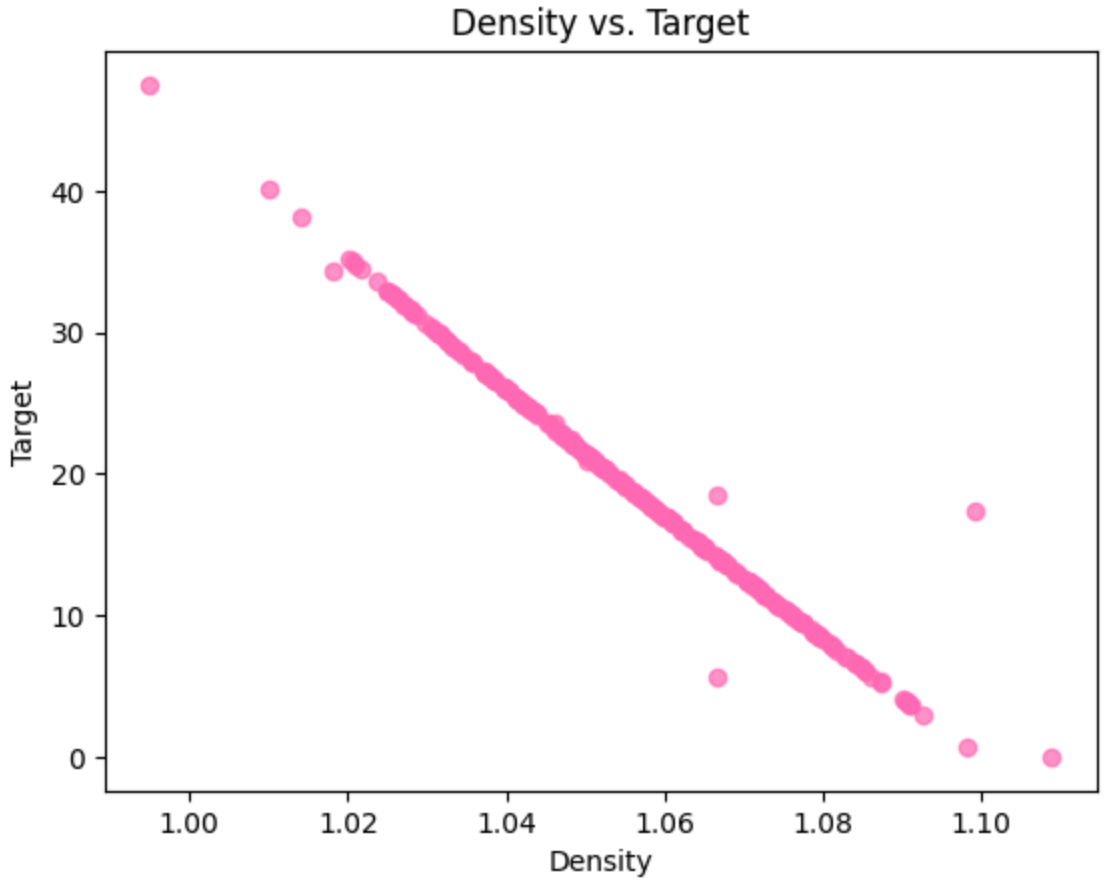
In [ ]:

Source referenced: https://matplotlib.org/stable/api/axes_api.html

Upon viewing the scatter matrix and histograms, I decided to further explore the relationships between density and the target feature, weight and hip measurements, and chest and abdomen measurement. I seleced these reltaionships based on how correlated they seemed, as these three pairs seemed to display nearly linear correlations.

Density and Target Plot:

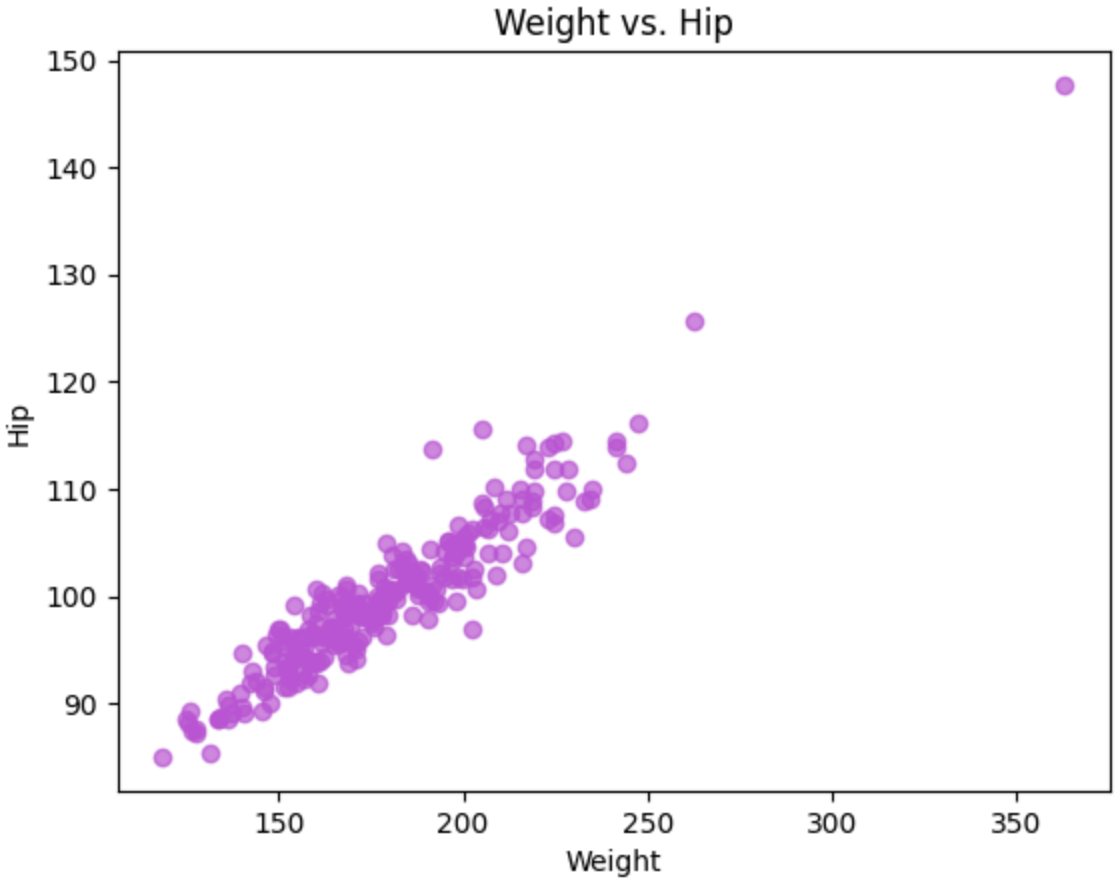
In [33]:
```
density = bf['Density']
target = bf['target']
plt.scatter(density, target, alpha=0.7, color="hotpink")
plt.xlabel("Density")
plt.ylabel("Target")
plt.title("Density vs. Target")
plt.show()
```



> The plot shows a strong linear relationship between density and the target feature. As density increases, the target feature also decreases, with a few outliers to this relationship. This indicates that density plays a key role in predicting body mass percentage.

Weight and Hip Plot:

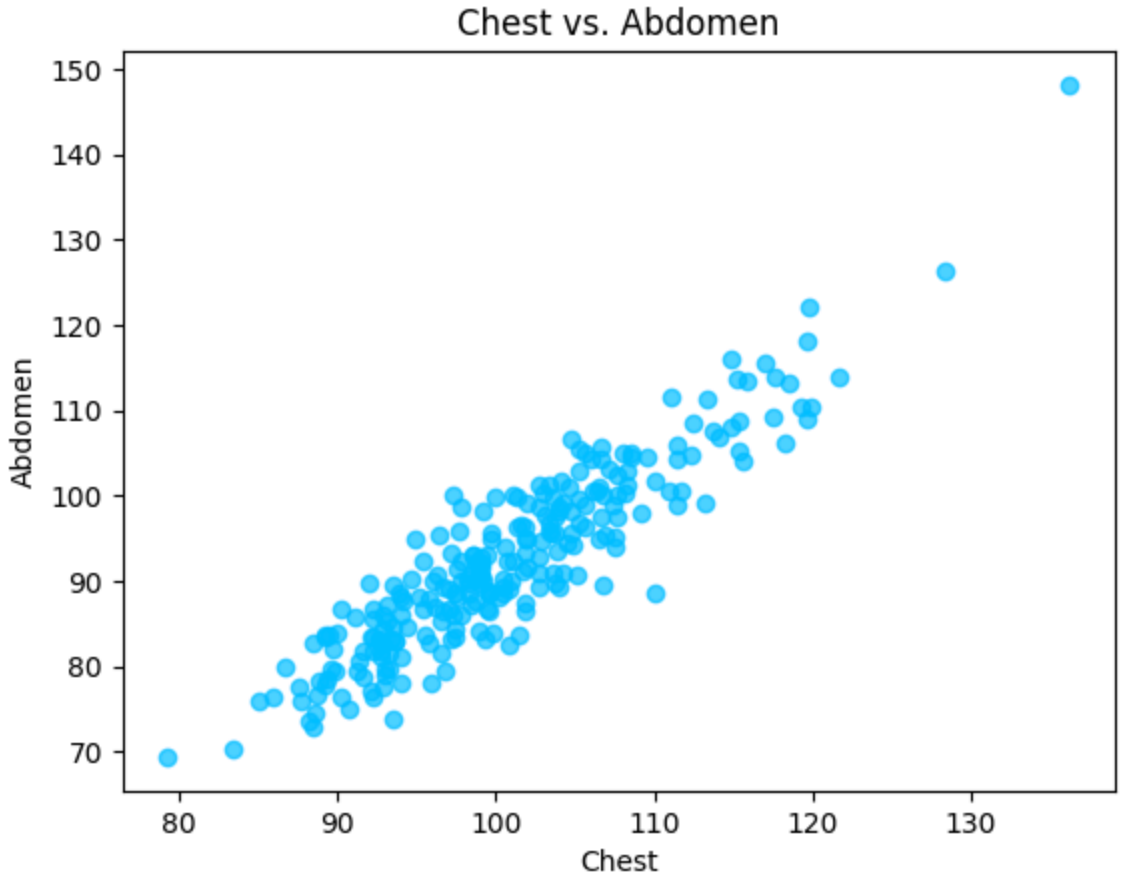
In [34]:
```
weight = bf['Weight']
hip = bf['Hip']
plt.scatter(weight, hip, alpha=0.7, color="mediumorchid")
plt.xlabel("Weight")
plt.ylabel("Hip")
plt.title("Weight vs. Hip")
plt.show()
```



> This plot shows a storng positive correlation between weight and hip measurements. This relationship is more spread out than the one displayed between density and the target variable. This plot indicates that a higher weight will most likely be associated with a larger hip measurement.

Chest and Abdomen Plot:

In [35]:
```
chest = bf['Chest']
abdomen = bf['Abdomen']
plt.scatter(chest, abdomen, alpha=0.7, color="deepskyblue")
plt.xlabel("Chest")
plt.ylabel("Abdomen")
plt.title("Chest vs. Abdomen")
plt.show()
```



> This plot displays a strong positive linear relationship between chest and abdomen measurements. As chest measurements increase, so do those of the abdomen, with some outliers.

Problem 4 (5 points)

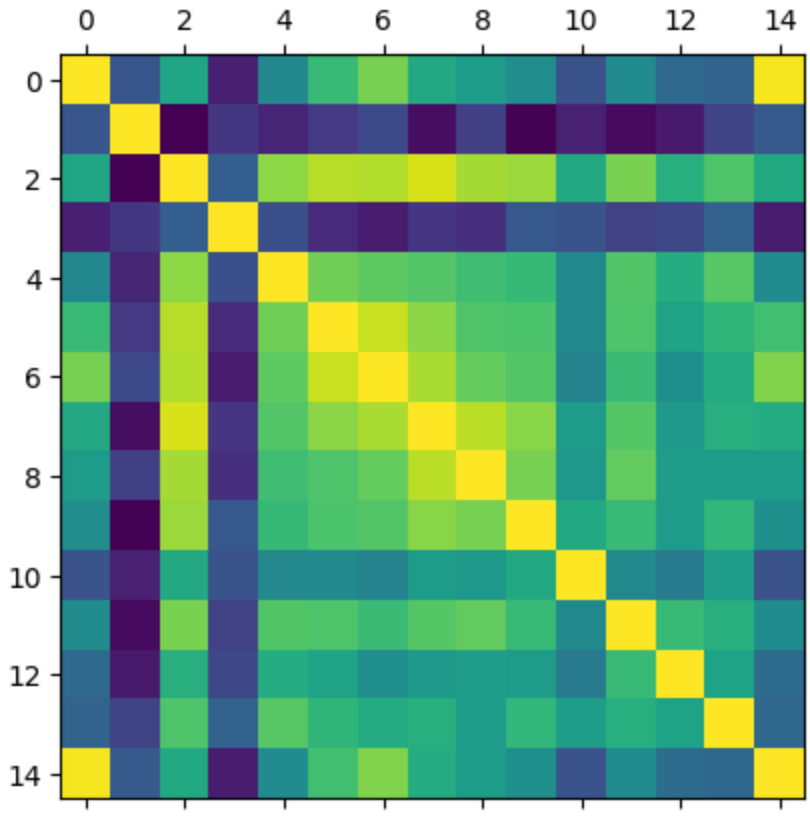
Plot the correlation matrix of the numeric columns in the data set. Which pair of different columns were highlighted as the most correlated?

In [36]:
```
# YOUR CODE HERE
corr_m = bf.corr(numeric_only=True).abs() #correlation matrix of the dataset
#remove diagonal (all 1s)
m = np.equal(*np.indices(corr_m.shape))
c_matrix = corr_m.mask(m)
c_matrix.max()
```

Out[36]:
```
Density    0.987782
Age        0.291458
Weight     0.940884
Height     0.322065
Neck       0.830716
Chest      0.915828
Abdomen    0.915828
Hip        0.940884
Thigh      0.896410
Knee       0.853167
Ankle      0.613685
Biceps     0.800416
Forearm    0.678255
Wrist      0.744826
target     0.987782
dtype: float64
```

In [37]:
```
#plot correlation matrix
plt.matshow(corr_m)
```

Out[37]:
```
<matplotlib.image.AxesImage at 0x7c0544fddca0>
```



> The plot shows the correlation between the features. The features are labelled numerically as they appear in the dataset, with '0' = 'Density', '1' = 'Age', '2' = 'Weight', and so on.

The correlation matrix indicates that there is high correlation between the following pairs:

- Density and Target (0.987782)
- Weight and Hip (0.940884)
- Chest and Abdomen (0.915828)

Problem 5 (10 points)

Pick three different regression model classes to try in problem 6 from the scikit-learn documentation. For each class, provide a link to the scikit-learn documentation, and a link to another web page describing how that kind of model works. The second link should not be from scikit-learn, but Wikipedia is acceptable. You do not need to understand the methods at this time, but it is good to be comfortable researching them.

Note: if you chose a classification dataset (e.g. Titanic), treat this as a 0-1 regression problem where 1 corresponds to membership in the target class and 0 corresponds to not being in the class. Do the same for all the following regression problems.

Regression Model 1: Linear Regression

- Link to scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Link to supplemental web page: https://www.datacamp.com/tutorial/sklearn-linear-regression

Regression Model 2: Random Forest Regressor

- Link to scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- Link to supplemental web page: https://www.geeksforgeeks.org/random-forest-regression-in-python/

Regression Model 3: Ridge Regression

- Link to scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
- Link to supplemental web page: https://www.geeksforgeeks.org/ml-ridge-regressor-using-sklearn/

Problem 6 (50 points)

Build three different regression models using the entire data set. Plot the actual target vs the predicted values for each in one chart. Compute the L2 and L1 losses for each of them. You may use any regression class provided provided by scikit-learn, and you may reuse one class as long as you change its parameters enough to see different results.

Regression Model 1: Linear Regression

In [38]:
```
# YOUR CODE HERE
from sklearn.linear_model import LinearRegression
features = bf.drop(columns=['target']) #contains all features
target = bf['target'] #contains only the target
#t
lr_model = LinearRegression()
lr_model.fit(features, target)
tgt_lr = lr_model.predict(features)
```

In [39]:
```
#Calculate L2 loss (Mean Squared Error or MSE)
from sklearn.metrics import mean_squared_error, mean_absolute_error
L2_lr = mean_squared_error(target, tgt_lr)
#Calculate L1 loss (Mean Absolute Error)
L1_lr = mean_absolute_error(target, tgt_lr)
print(f"The L2 loss using Linear Regression is: {L2_lr}")
print(f"The L1 loss using Linear Regression is: {L1_lr}")
```

```
The L2 loss using Linear Regression is: 1.5272041202364655
The L1 loss using Linear Regression is: 0.48019622407529433
```

Regression Model 2: Random Forest Regressor

In [40]:
```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(random_state=42, max_depth=5)
rf_model.fit(features, target)
tgt_rf = rf_model.predict(features)
```

In [41]:
```
#Calculate L2 loss (Mean Squared Error or MSE)
L2_rf = mean_squared_error(target, tgt_rf)
#Calculate L1 loss (Mean Absolute Error)
L1_rf = mean_absolute_error(target, tgt_rf)
print(f"The L2 loss using Random Forest Regressor is: {L2_rf}")
print(f"The L1 loss using Random Forest Regressor is: {L1_rf}")
```

```
The L2 loss using Random Forest Regressor is: 0.33796344458791405
The L1 loss using Random Forest Regressor is: 0.22372848195602946
```

Regression Model 3: Ridge Regression

In [42]:
```
from sklearn.linear_model import Ridge
r_model = Ridge()
r_model.fit(features, target)
tgt_r = r_model.predict(features)
```

In [43]:
```
#Calculate L2 loss (Mean Squared Error or MSE)
L2_r = mean_squared_error(target, tgt_r)
#Calculate L1 loss (Mean Absolute Error)
L1_r = mean_absolute_error(target, tgt_r)
print(f"The L2 loss using Ridge Regression is: {L2_r}")
print(f"The L1 loss using Ridge Regression is: {L1_r}")
```

```
The L2 loss using Ridge Regression is: 16.7706267555516
The L1 loss using Ridge Regression is: 3.3670432335064477
```

> Largest L2 loss: Ridge Regression

> Largest L1 loss: Ridge Regression

Plot of all models:

In [44]:
```
#plot actual target vs. predicted values
#Linear Regression (LR):
plt.scatter(target, tgt_lr, color="lime", alpha=0.7, label='Linear Regressic
#Random Forest Regression (RFR):
plt.scatter(target, tgt_rf, color="turquoise", alpha=0.7, label="Random Fore
#Ridge Regression:
plt.scatter(target, tgt_r, color="gold", alpha=0.7, label="Ridge Regression"
#Ideal prediction line:
plt.plot([target.min(), target.max()], [target.min(), target.max()], '--', c
#Label plot
plt.xlabel("Actual Target Values")
plt.ylabel("Predicted Target Values")
plt.title("Actual vs. Predicted Body Fat Percentage")
plt.grid(True)
plt.legend()
plt.show()
```

Problem 7 (30 points)

Use 5-fold cross-validation to repeat building the same three kinds of regression models. Compare the L2 losses predicted by cross-validation against the L2 losses training against the whole data set. (The difference is likely from overfitting in the latter.)

Linear Regression: 5-fold cross-validation

In [45]:
```
# YOUR CODE HERE
from sklearn.model_selection import train_test_split, cross_validate
#5-fold cross validation
train_features, test_features, train_target, test_target = train_test_split(
#linear regression model
lr = LinearRegression()
lr_cv = cross_validate(lr, train_features, train_target) #cross-validate
#take mean of 'test_score' to find L2 loss
lr_L2_cv = lr_cv['test_score'].mean()
print(f"5-Fold CV L2 Loss (Linear Regression): {lr_L2_cv}")
print(f"Full Data L2 Loss (Linear Regression): {L2_lr}")
```

```
5-Fold CV L2 Loss (Linear Regression): 0.9657764823286226
Full Data L2 Loss (Linear Regression): 1.5272041202364655
```

> Larger L2 loss using the whole dataset for Linear Regression.

Random Forest Regressor: 5-fold cross-validation

In [46]:
```
#random forest regressor model
rf = RandomForestRegressor(random_state=42, max_depth=5)
rf_cv = cross_validate(rf, train_features, train_target) #cross-validate
#take mean of 'test_score' to find L2 loss
rf_L2_cv = rf_cv['test_score'].mean()
print(f"5-Fold CV L2 Loss (Random Forest Regressor): {rf_L2_cv}")
print(f"Full Data L2 Loss (Random Forest Regressor): {L2_rf}")
```

```
5-Fold CV L2 Loss (Random Forest Regressor): 0.9645227892547978
Full Data L2 Loss (Random Forest Regressor): 0.33796344458791405
```

> Larger L2 loss using 5-fold cross validation for Random Forest Regressor.

Ridge Regression: 5-fold cross-validation

In [47]:
```
#ridge regression model
r = Ridge()
r_cv = cross_validate(r, train_features, train_target) #cross-validate
#take mean of 'test_score' to find L2 loss
r_L2_cv = r_cv['test_score'].mean()
print(f"5-Fold CV L2 Loss (Ridge Regression): {r_L2_cv}")
print(f"Full Data L2 Loss (Ridge Regression): {L2_r}")
```

```
5-Fold CV L2 Loss (Ridge Regression): 0.6949836962503431
Full Data L2 Loss (Ridge Regression): 16.7706267555516
```

> Larger L2 loss using the whole dataset for Ridge Regression.

Sources referenced: AI chat cited in problem 11, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html , https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Problem 8 (25 points)

Build three different regression models as in problem 6, but preprocess the data so that each column has mean zero and standard deviation one first. For full credit, use a scikit-learn pipeline for each model. For each model, compare the L2 losses -- which of them performed differently from your results in problem 6?

(This process will be covered in week 13.)

Regression Model 1: Linear Regression

In [48]:
```
# YOUR CODE HERE
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
#make pipeline with scaler and lin reg
lr_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lin_reg', LinearRegression())
])
#fit pipeline to data:
lr_pipeline.fit(features, target)
lr_pipe_pred = lr_pipeline.predict(features)
L2_lr_pipe = mean_squared_error(target, lr_pipe_pred)
print(f"L2 Loss (Linear Regression, preprocessed): {L2_lr_pipe}")
print(f"L2 Loss (Linear Regression): {L2_lr}")
```

```
L2 Loss (Linear Regression, preprocessed): 1.5272041202364754
L2 Loss (Linear Regression): 1.5272041202364655
```

> Performed the same for Linear Regression

Regression Model 2: Random Forest Regressor

In [49]:
```
rf_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('random_forest', RandomForestRegressor(random_state=42, max_depth=5))
])
#fit pipeline to data:
rf_pipeline.fit(features, target)
```

```
rf_pipe_pred = rf_pipeline.predict(features)
L2_rf_pipe = mean_squared_error(target, rf_pipe_pred)
print(f"L2 Loss (Random Forest Regressor, preprocessed): {L2_rf_pipe}")
print(f"L2 Loss (Random Forest Regressor): {L2_rf}")
```

```
L2 Loss (Random Forest Regressor, preprocessed): 0.33789282042250723
L2 Loss (Random Forest Regressor): 0.33796344458791405
```

> Performed slightly different than results from p6.

Regression Model 3: Ridge Regression

In [50]:
```
r_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('ridget', Ridge())
])
#fit pipeline to data:
r_pipeline.fit(features, target)
r_pipe_pred = r_pipeline.predict(features)
L2_r_pipe = mean_squared_error(target, r_pipe_pred)
print(f"L2 Loss (Ridge Regression, preprocessed): {L2_r_pipe}")
print(f"L2 Loss (Ridge Regression): {L2_r}")
```

```
L2 Loss (Ridge Regression, preprocessed): 1.530501814078208
L2 Loss (Ridge Regression): 16.7706267555516
```

> Performed significantly different than results from p6.

Sources referenced: https://scikit-learn.org/stable/modules/preprocessing.html , https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

Problem 9 (5 points)

A colleague suggests that you find better models by repeatedly building decision trees with random depth limits. They say that trying 1000 such models will likely find an improvement as long as you use cross validation. Give a one sentence response to this suggestion.

> Although trying 1,000 such models with random depth limits might yield better performances, the lack of proper model selection techniques and regularization risks overfitting.

Problem 10 (5 points)

Pick a best model from all the models that you built and otherwise described in this project. Explain how you picked it, including what criteria you chose, and how the other

models compared by that criteria. As much as possible, justify that problem in the context of the original data set.

In [51]:
```
# YOUR CODE HERE
print(f"5-Fold CV L2 Loss (Linear Regression): {lr_L2_cv}")
print(f"L2 Loss (Linear Regression, preprocessed): {L2_lr_pipe}")
print(f"Full Data L2 Loss (Linear Regression): {L2_lr}")
print(f"5-Fold CV L2 Loss (Random Forest Regressor): {rf_L2_cv}")
print(f"L2 Loss (Random Forest Regressor, preprocessed): {L2_rf_pipe}")
print(f"Full Data L2 Loss (Random Forest Regressor): {L2_rf}")
print(f"5-Fold CV L2 Loss (Ridge Regression): {r_L2_cv}")
print(f"L2 Loss (Ridge Regression, preprocessed): {L2_r_pipe}")
print(f"Full Data L2 Loss (Ridge Regression): {L2_r}")
```

```
5-Fold CV L2 Loss (Linear Regression): 0.9657764823286226
L2 Loss (Linear Regression, preprocessed): 1.5272041202364754
Full Data L2 Loss (Linear Regression): 1.5272041202364655
5-Fold CV L2 Loss (Random Forest Regressor): 0.9645227892547978
L2 Loss (Random Forest Regressor, preprocessed): 0.33789282042250723
Full Data L2 Loss (Random Forest Regressor): 0.33796344458791405
5-Fold CV L2 Loss (Ridge Regression): 0.6949836962503431
L2 Loss (Ridge Regression, preprocessed): 1.530501814078208
Full Data L2 Loss (Ridge Regression): 16.7706267555516
```

Model	5-Fold CV L2	Preprocessed L2	Full Data L2
Linear Regression	0.9657	1.5272	1.5272
Random Forest Regressor	0.9643	0.3378	0.3379
Ridge Regression	0.6949	1.5305	16.7706

> Random Forest Regressor gave the lowest error of the three models. Based on this information, I am selecting it as the best model of the project.

Problem 11 (5 points)

Please give session links for any AI tools that you used for this final project. If you did not use any such tools, please say "None".

You will get full credit for this problem as long as you add links in the cell below or update the text to say None. There will be no penalties for this tool usage; we simply to understand how you are handling these problems now.

Answers to this problem will not be accepted outside the normal submission process. Emails after the deadline saying that you forgot will not get you credit.

https://chatgpt.com/share/68115717-422c-8002-96ec-cc636a39228c