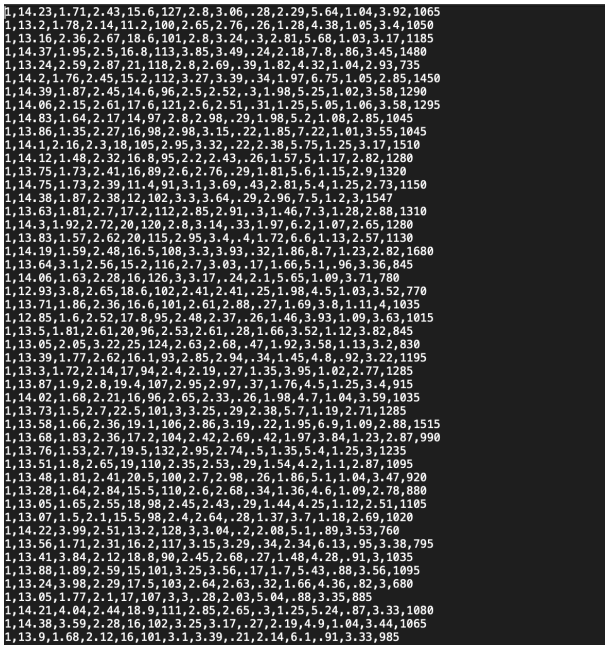
# Mini Project One

CSCI-347: Data Mining
February 8th, 2024
Megan Steinmasel
Rohan Kamat
Michael Belmear

## Part One: Introduction

The data set we have chosen is the Wine data set. This data are the results of a chemical analysis of Italian wines that are derived from 3 different cultivars. The data analysis determined the quantities of 13 constituents found in each of the different cultivars of wine. This data was uploaded on 6/30/1991 by Riccardo Leardi. With a total of 14 attributes and 178 entities, the dataset contains both numerical and categorical attributes. Out of these attributes, there are 13 numerical and 1 categorical attribute. The numerical attributes include Alcohol, Malic Acid, Ash, Alkalinity of Ash, Magnesium, Total Phenols, Flavonoids, Non Flavonoid Phenols, Proanthocyanidins, Color Intensity, Hue, OD280/OD315 of Diluted Wines, and Proline. The single categorical attribute is Class, with values of either 1, 2, or 3. The Class attribute represents wines that are derived from 3 different cultivars. For the reason that the categorical variable  is already in numerical form, we do not have to perform encoding. In the Wine data set, there are no missing values. This data set was of interest to us because we find wine varieties to be an interesting topic. Here is a snippet of some of the wine data we are working with:



```
1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735
1,14.2,1.76,2.45,15.2,112,3.27,3.39,.34,1.97,6.75,1.05,2.85,1450
1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290
1,14.06,2.15,2.61,17.6,121,2.6,2.51,.31,1.25,5.05,1.06,3.58,1295
1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045
1,13.86,1.35,2.27,16,98,2.98,3.15,.22,1.85,7.22,1.01,3.55,1045
1,14.1,2.16,2.3,18,105,2.95,3.32,.22,2.38,5.75,1.25,3.17,1510
1,14.12,1.48,2.32,16.8,95,2.2,2.43,.26,1.57,5,1.17,2.82,1280
1,13.75,1.73,2.41,16,89,2.6,2.76,.29,1.81,5.6,1.15,2.9,1320
1,14.75,1.73,2.39,11.4,91,3.1,3.69,.43,2.81,5.4,1.25,2.73,1150
1,14.38,1.87,2.38,12,102,3.3,3.64,.29,2.96,7.5,1.2,3,1547
1,13.63,1.81,2.7,17.2,112,2.85,2.91,.3,1.46,7.3,1.28,2.88,1310
1,14.3,1.92,2.72,20,120,2.8,3.14,.33,1.97,6.2,1.07,2.65,1280
1,13.83,1.57,2.62,20,115,2.95,3.4,.4,1.72,6.6,1.13,2.57,1130
1,14.19,1.59,2.48,16.5,108,3.3,3.93,.32,1.86,8.7,1.23,2.82,1680
1,13.64,3.1,2.56,15.2,116,2.7,3.03,.17,1.66,5.1,.96,3.36,845
1,14.06,1.63,2.28,16,126,3,3.17,.24,2.1,5.65,1.09,3.71,780
1,12.93,3.8,2.65,18.6,102,2.41,2.41,.25,1.98,4.5,1.03,3.52,770
1,13.71,1.86,2.36,16.6,101,2.61,2.88,.27,1.69,3.8,1.11,4,1035
1,12.85,1.6,2.52,17.8,95,2.48,2.37,.26,1.46,3.93,1.09,3.63,1015
1,13.5,1.81,2.61,20,96,2.53,2.61,.28,1.66,3.52,1.12,3.82,845
1,13.05,2.05,3.22,25,124,2.63,2.68,.47,1.92,3.58,1.13,3.2,830
1,13.39,1.77,2.62,16.1,93,2.85,2.94,.34,1.45,4.8,.92,3.22,1195
1,13.3,1.72,2.14,17,94,2.4,2.19,.27,1.35,3.95,1.02,2.77,1285
1,13.87,1.9,2.8,19.4,107,2.95,2.97,.37,1.76,4.5,1.25,3.4,915
1,14.02,1.68,2.21,16,96,2.65,2.33,.26,1.98,4.7,1.04,3.59,1035
1,13.73,1.5,2.7,22.5,101,3,3.25,.29,2.38,5.7,1.19,2.71,1285
1,13.58,1.66,2.36,19.1,106,2.86,3.19,.22,1.95,6.9,1.09,2.88,1515
1,13.68,1.83,2.36,17.2,104,2.42,2.69,.42,1.97,3.84,1.23,2.87,990
1,13.76,1.53,2.7,19.5,132,2.95,2.74,.5,1.35,5.4,1.25,3,1235
1,13.51,1.8,2.65,19,110,2.35,2.53,.29,1.54,4.2,1.1,2.87,1095
1,13.48,1.81,2.41,20.5,100,2.7,2.98,.26,1.86,5.1,1.04,3.47,920
1,13.28,1.64,2.84,15.5,110,2.6,2.68,.34,1.36,4.6,1.09,2.78,880
1,13.05,1.65,2.55,18,98,2.45,2.43,.29,1.44,4.25,1.12,2.51,1105
1,13.07,1.5,2.1,15.5,98,2.4,2.64,.28,1.37,3.7,1.18,2.69,1020
1,14.22,3.99,2.51,13.2,128,3,3.04,.2,2.08,5.1,.89,3.53,760
1,13.56,1.71,2.31,16.2,117,3.15,3.29,.34,2.34,6.13,.95,3.38,795
1,13.41,3.84,2.12,18.8,90,2.45,2.68,.27,1.48,4.28,.91,3,1035
1,13.88,1.89,2.59,15,101,3.25,3.56,.17,1.7,5.43,.88,3.56,1095
1,13.24,3.98,2.29,17.5,103,2.64,2.63,.32,1.66,4.36,.82,3,680
1,13.05,1.77,2.1,17,107,3,3,.28,2.03,5.04,.88,3.35,885
1,14.21,4.04,2.44,18.9,111,2.85,2.65,.3,1.25,5.24,.87,3.33,1080
1,14.38,3.59,2.28,16,102,3.25,3.17,.27,2.19,4.9,1.04,3.44,1065
1,13.9,1.68,2.12,16,101,3.1,3.39,.21,2.14,6.1,.91,3.33,985
```

## Part Two: Write Python Code for Data Analysis

- A function that will compute the **mean** of a numerical, multidimensional data set input as a 2-dimensional numpy array:

```python
def compute_mean(data):

    num_rows = len(data)
    num_cols = len(data.columns)

    # Initialize a list to hold the sum for each dimension
    dimension_sum = [0] * num_cols

    # Calculate the sum for each dimension
    for col in data.columns:
        dimension_sum[col] = data[col].sum()

    # Calculate the mean for each dimension
    mean_values = [sum_value / num_rows for sum_value in dimension_sum]

    return mean_values


data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

print(compute_mean(data_set))
```

- A function that will compute the **sample covariance** between two attributes that are input as one-dimensional numpy vectors:

```python
def compute_sample_covariance(x, y):

    n = len(x)

    # Compute means
    mean_x = sum(x) / n
    mean_y = sum(y) / n

    # Compute covariance
    covariance = sum((x[i] - mean_x) * (y[i] - mean_y) for i in range(n)) / (n - 1)

    return covariance

# Read the data from the CSV file
data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

# Extract the first and second attributes from the dataset
x = data_set.iloc[:, 0].tolist()  # Assuming the first attribute is in the first column
y = data_set.iloc[:, 1].tolist()  # Assuming the second attribute is in the second column

print(compute_sample_covariance(x, y))
```

- A function that will compute the **correlation** between two attributes that are input as two numpy vectors:

```python
def compute_correlation(x, y):

    n = len(x)

    # Compute means
    mean_x = sum(x) / n
    mean_y = sum(y) / n

    # Compute standard deviations
    std_dev_x = (sum((xi - mean_x) ** 2 for xi in x) / n) ** 0.5
    std_dev_y = (sum((yi - mean_y) ** 2 for yi in y) / n) ** 0.5

    # Compute correlation
    correlation = sum((x[i] - mean_x) * (y[i] - mean_y) for i in range(n)) / (n * std_dev_x * std_dev_y)

    return correlation

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

# Extract the first and second attributes from the dataset
x = data_set.iloc[:, 0].tolist()  # Assuming the first attribute is in the first column
y = data_set.iloc[:, 1].tolist()  # Assuming the second attribute is in the second column

print(compute_correlation(x, y))
```

- A function that will normalize the attributes in a two-dimensional numpy array using **range normalization**:

```python
def range_normalize(data):

    num_rows, num_cols = data.shape

    normalized_data = pd.DataFrame()

    for col in data.columns:
        # Find the minimum and maximum values for each column
        min_val = data[col].min()
        max_val = data[col].max()

        # Normalize each value in the column
        normalized_column = (data[col] - min_val) / (max_val - min_val)
        normalized_data[col] = normalized_column

    return normalized_data

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

normalized_data_set = range_normalize(data_set)
print(normalized_data_set)
```

- A function that will normalize the attributes in a two-dimensional numpy array using **standard normalization**:

```
def standard_normalize(data):

    num_rows, num_cols = data.shape

    normalized_data = pd.DataFrame()

    for col in data.columns:
        # Calculate the mean and standard deviation for each column
        mean_val = data[col].mean()
        std_dev = data[col].std()

        # Normalize each value in the column
        normalized_column = (data[col] - mean_val) / std_dev if std_dev != 0 else 0
        normalized_data[col] = normalized_column

    return normalized_data

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

normalized_data_set = standard_normalize(data_set)
print(normalized_data_set)
```

- A function that will compute the **covariance matrix** of a data set:

```
def compute_covariance_matrix(data):

    num_features = data.shape[1]
    num_samples = data.shape[0]

    # Compute the mean of each feature
    means = data.mean()

    # Compute the covariance between each pair of features
    covariance_matrix = pd.DataFrame(index=data.columns, columns=data.columns)
    for i in range(num_features):
        for j in range(num_features):
            covariance = ((data.iloc[:, i] - means[i]) * (data.iloc[:, j] - means[j])).sum() / (num_samples - 1)
            covariance_matrix.iloc[i, j] = covariance

    return covariance_matrix

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

covariance_matrix = compute_covariance_matrix(data_set)
print(covariance_matrix)
```

- A function that will **label-encode** a two-dimensional categorical data array that is passed in as input:

```python
def label_encode(data):

    label_map = {}
    encoded_data = pd.DataFrame()

    # Iterate over each column
    for col in data.columns:
        unique_values = data[col].unique()

        # Create a mapping from unique values to labels
        for i, value in enumerate(unique_values):
            if value not in label_map:
                label_map[value] = i

        # Label encode the data in the current column
        encoded_column = [label_map[value] for value in data[col]]
        encoded_data[col] = encoded_column

    return encoded_data, label_map

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine.data', header=None)

encoded_data, label_map = label_encode(data_set)
print("Encoded data:")
print(encoded_data)
print("\nLabel map:")
print(label_map)
```

## Part Three: Analyze Data

In Part Three, we will analyze the data with the code from Part Two. Before answering the following questions, it is noted that we must convert all categorical attributes using label encoding or one-hot-encoding, and fill in missing values with the attribute mean. The wine data set contains no missing values and one categorical variable that is already in numerical form, so we will continue with the original data set. We will analyze the wine data with the following analysis.

(Megan S.) The **multivariate mean** of the wine data matrix is [1.9382022471910112, 13.00061797752809, 2.3363483146067416, 2.3665168539325845, 19.49494382022472, 99.74157303370787, 2.295112359550562, 2.0292696629213487, 0.3618539325842696, 1.5908988764044945, 5.058089882022472, 0.9574494382022471, 2.6116853932584267, 746.8932584269663]. The multivariate mean of the data set shows the average value of each variable across all observations of the data set. The first index of the multivariate mean is the mean of the categorical variable, Class, and the rest of the indexes follow the order of the attributes listed below.

1) Mean Alcohol: 13.00061797752809,
2) Mean Malic Acid: 2.3363483146067416
3) Mean Ash: 2.3665168539325845
4) Mean Alkalinity of Ash:  19.49494382022472
5) Mean Magnesium: 99.74157303370787
6) Mean Total Phenols: 2.295112359550562

7) Mean Flavonoids: 2.0292696629213487
8) Mean Non Flavonoid Phenols: 0.3618539325842696
9) Mean Proanthocyanidins: 1.5908988764044945
10) Mean Color Intensity: 5.058089882022472
11) Mean Hue: 0.9574494382022471
12) Mean OD280/OD315 of Diluted Wines: 2.6116853932584267
13) Mean Proline: 746.8932584269663

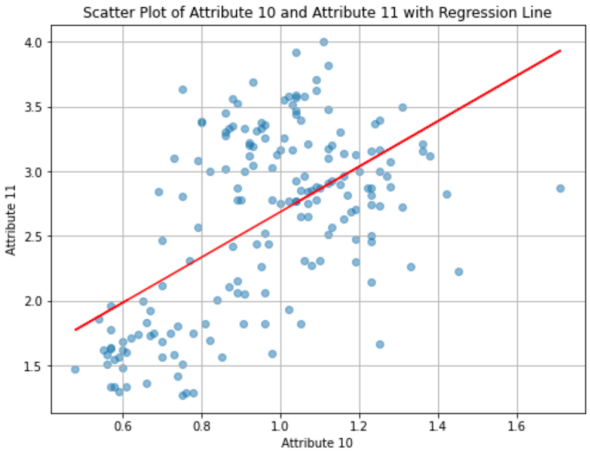
(Megan S.) The **covariance matrix** of the wine data matrix is shown below. The matrix displays the covariance between each pair of attributes. Specifically, the covariance between attributes Xi and Xj is given by the (i, j)th element of the covariance matrix.

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0.600679 | -0.206515 | 0.379039 | -0.010555 | 1.340364 | -2.315495 |
| 1 | -0.206515 | 0.659062 | 0.085611 | 0.047115 | -0.841093 | 3.139878 |
| 2 | 0.379039 | 0.085611 | 1.248015 | 0.050277 | 1.076332 | -0.87078 |
| 3 | -0.010555 | 0.047115 | 0.050277 | 0.075265 | 0.406208 | 1.122937 |
| 4 | 1.340364 | -0.841093 | 1.076332 | 0.406208 | 11.152686 | -3.97476 |
| 5 | -2.315495 | 3.139878 | -0.87078 | 1.122937 | -3.97476 | 203.989335 |
| 6 | -0.348835 | 0.146887 | -0.234338 | 0.022146 | -0.671149 | 1.91647 |
| 7 | -0.656091 | 0.192033 | -0.45863 | 0.031535 | -1.172083 | 2.793087 |
| 8 | 0.047177 | -0.015754 | 0.040733 | 0.006358 | 0.150422 | -0.455563 |
| 9 | -0.221413 | 0.063518 | -0.141147 | 0.001516 | -0.377176 | 1.932832 |
| 10 | 0.477339 | 1.028283 | 0.644838 | 0.164654 | 0.145024 | 6.620521 |
| 11 | -0.109368 | -0.013313 | -0.143326 | -0.004682 | -0.209118 | 0.180851 |
| 12 | -0.433737 | 0.041698 | -0.292447 | 0.000762 | -0.656234 | 0.669308 |
| 13 | -154.667651 | 164.567185 | -67.548867 | 19.319739 | -463.355345 | 1769.1587 |

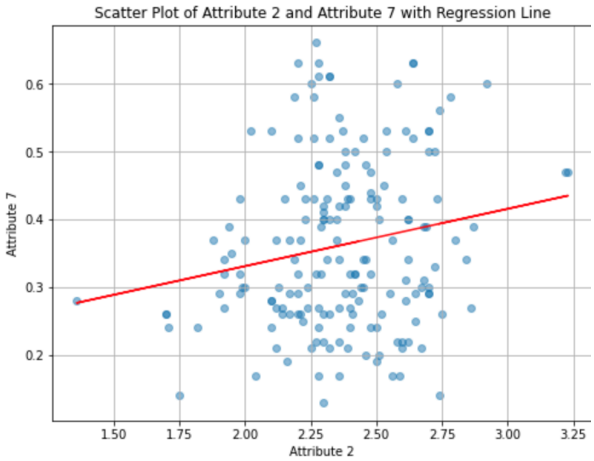
| | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|
| 0 | -0.348835 | -0.656091 | 0.047177 | -0.221413 | 0.477339 | -0.109368 |
| 1 | 0.146887 | 0.192033 | -0.015754 | 0.063518 | 1.028283 | -0.013313 |
| 2 | -0.234338 | -0.45863 | 0.040733 | -0.141147 | 0.644838 | -0.143326 |
| 3 | 0.022146 | 0.031535 | 0.006358 | 0.001516 | 0.164654 | -0.004682 |
| 4 | -0.671149 | -1.172083 | 0.150422 | -0.377176 | 0.145024 | -0.209118 |
| 5 | 1.91647 | 2.793087 | -0.455563 | 1.932832 | 6.620521 | 0.180851 |
| 6 | 0.39169 | 0.54047 | -0.035045 | 0.219373 | -0.079998 | 0.062039 |
| 7 | 0.54047 | 0.997719 | -0.066867 | 0.373148 | -0.399169 | 0.124082 |
| 8 | -0.035045 | -0.066867 | 0.015489 | -0.02606 | 0.040121 | -0.007471 |
| 9 | 0.219373 | 0.373148 | -0.02606 | 0.327595 | -0.033504 | 0.038665 |
| 10 | -0.079998 | -0.399169 | 0.040121 | -0.033504 | 5.374449 | -0.276506 |
| 11 | 0.062039 | 0.124082 | -0.007471 | 0.038665 | -0.276506 | 0.052245 |
| 12 | 0.311021 | 0.558262 | -0.044469 | 0.210933 | -0.705813 | 0.091766 |
| 13 | 98.171057 | 155.447492 | -12.203586 | 59.554334 | 230.76748 | 17.000223 |

| | 12 | 13 |
|---|---|---|
| 0 | -0.433737 | -154.667651 |
| 1 | 0.041698 | 164.567185 |
| 2 | -0.292447 | -67.548867 |
| 3 | 0.000762 | 19.319739 |
| 4 | -0.656234 | -463.355345 |
| 5 | 0.669308 | 1769.1587 |
| 6 | 0.311021 | 98.171057 |
| 7 | 0.558262 | 155.447492 |
| 8 | -0.044469 | -12.203586 |
| 9 | 0.210933 | 59.554334 |
| 10 | -0.705813 | 230.76748 |
| 11 | 0.091766 | 17.000223 |
| 12 | 0.504086 | 69.927526 |
| 13 | 69.927526 | 99166.717355 |

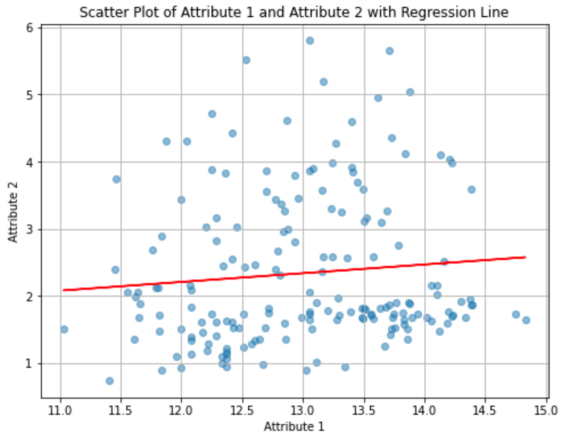
We choose **5 pairs of attributes** that we thought could be related. We created **scatter plots** of all 5 pairs along with a description and analysis that summarizes why these pairs of attributes might be related.



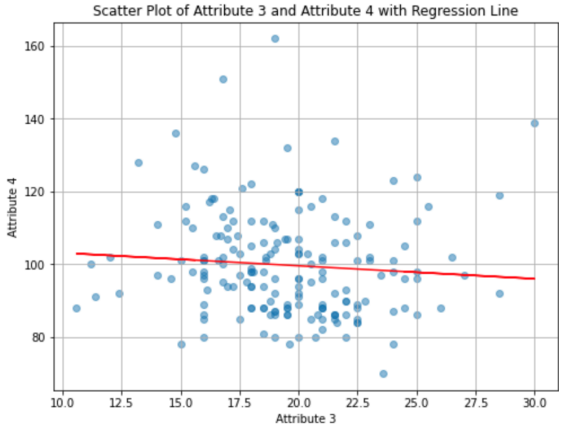

(Megan S.) As seen above, this is a scatterplot of attribute 10 (Color Intensity) and attribute 11 (Hue). Color intensity and hue are two attributes that relate to the color of wine, for this reason, we assume they are related. The scatterplot has a linear, positive, and moderate strength. This shows that color intensity and hue have an association because the data tends in a positive direction and the slope of the line of regression is not 0.



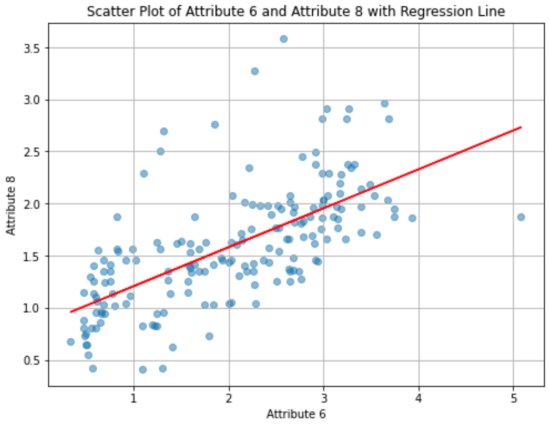

(Megan S.) As seen above, this is a scatterplot of attribute 2 (Malic Acid) and attribute 7 (Flavonoids). Malic acid is an organic compound found in various fruits and flavonoids are a diverse group of phytonutrients found in fruits, vegetables, tea, wine, and other plant-based foods. Because of this, we believe the two attributes could be related. The scatterplot shows a weak relationship in the positive direction. This means that there is not a relationship between malic acid and flavonoids, but it is not the strongest relationship.




(Megan S.) As seen above, this is a scatterplot of attribute 1 (Alcohol) and attribute 2 (Malic Acid). We assumed there might be a relationship between alcohol and malic acid because both are key attributes of wine. The scatterplot shows a very weak relationship because of the random distribution of the data points and because of this, we can conclude that if alcohol and malic acid does not have a relationship, it is a very weak one.




(Rohan K.) As seen above, this is a scatterplot of attribute 3 (Ash) and attribute 4 (Alkalinity of ash). Ash is defined as all those products remaining after igniting the residue left after the evaporation of the wine whereas the Alkalinity of as is defined as the sum of cations, other than the ammonium ion, combined with the organic acids in the wine. Our reason for thinking they are related is pretty obvious, the more Ash there is should have an effect on the overall Alkalinity of it. Looking at the graph, the relation seems to be a linear, negative, and weak strength relationship between Ash and the Alkalinity of ash with 2 outliers.

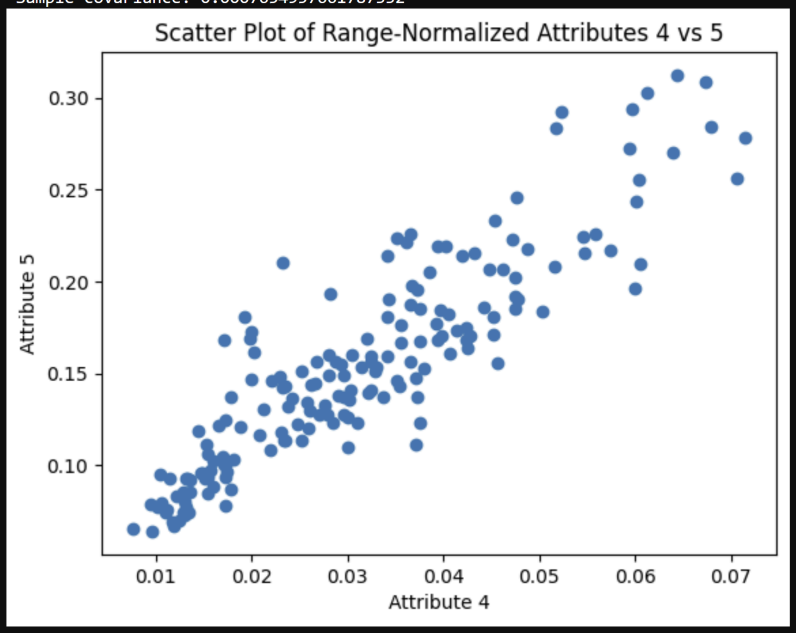



(Rohan K.) As seen above, this is a scatterplot of attribute 6 (Total Phenols) and attribute 8 (Non Flavonoid phenols). Phenols are defined as a class of chemical compounds consisting of one or more hydroxyl groups (−OH) bonded directly to an aromatic hydrocarbon group and Non Flavonoid Phenols are defined as hydroxycinnamates, stilbenes, and benzoic acids. Our reason for thinking they are related is obvious, because the total amount of Phenols should have an effect on the Non Flavonoid phenols. Looking at the graph the relation seems to support our initial query that they are related. In this case, it seems to be a linear, positive, and moderate relation between Total Phenols and Non Flavonoid Phenols.

(Michael B.) From the scatter plot and the output of the code, we can see that the **two range-normalized attributes with the greatest sample covariance** are attribute 4 and attribute 5 which are: Alkalinity of Ash and Magnesium respectively. Their sample covariance is 0.0007654957661787332. The positive covariance suggests a tendency for Alkalinity of Ash and Magnesium to vary together positively, although the magnitude of the covariance is relatively small.

```
def find_greatest_covariance(data_path):
    """Finds which range-normalized numerical attributes have the greatest sample covariance."""
    data = read_csv_data(data_path)
    normalized_data = normalize_data(data)
    covariance_matrix = compute_covariance_matrix(normalized_data)

    max_cov_value = -float('inf')
    max_cov_indices = None
    num_features = len(covariance_matrix)

    for i in range(num_features):
        for j in range(i+1, num_features):
            if covariance_matrix[i][j] > max_cov_value:
                max_cov_value = covariance_matrix[i][j]
                max_cov_indices = (i, j)

    return max_cov_indices, max_cov_value
```

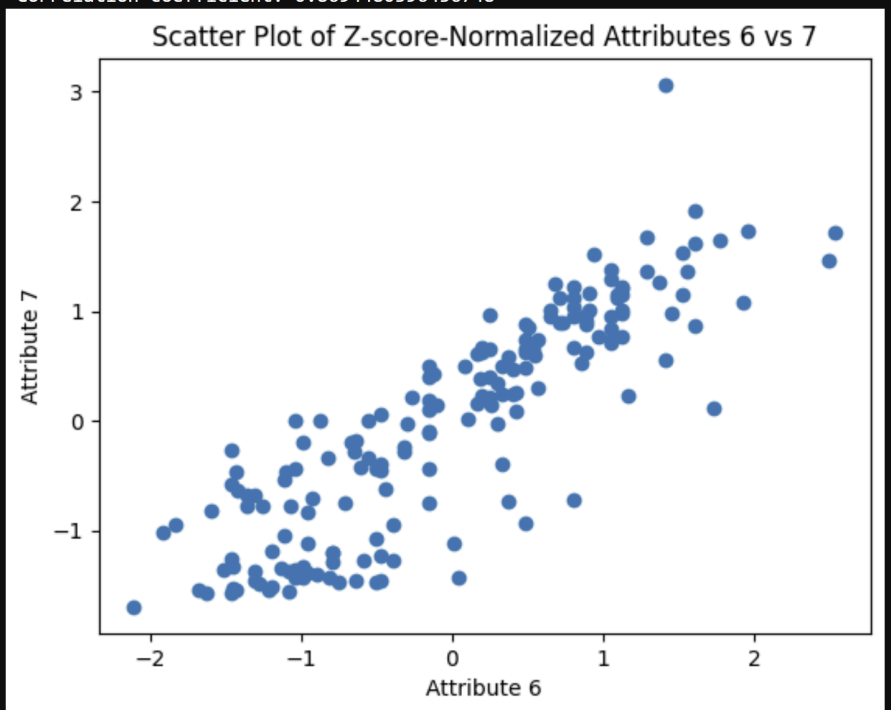
```
Range-normalized attributes with the greatest sample covariance: (4, 5)
Sample covariance: 0.0007654957661787332
```




(Michael B.) From the scatter plot and the output of the code, **we can see that the two Z-score-normalized attributes with the greatest correlation** are attribute 6 and attribute 7 which are: Total Phenols and Flavonoids respectively. Their correlation coefficient is 0.8694480396436748. This implies that as the value of Total Phenols increases, there is a corresponding increase in the value of Flavonoids, and vice versa. Such a high correlation coefficient suggests a robust association between Total Phenols and Flavonoids in the dataset.

```python
def find_greatest_correlation(data_path):
    """Finds which Z-score-normalized numerical attributes have the greatest correlation."""
    data = read_csv_data(data_path)
    normalized_data = z_score_normalize(data)
    correlation_matrix = compute_correlation_matrix(normalized_data)

    max_correlation_value = -float('inf')
    max_correlation_indices = None
    num_attributes = len(correlation_matrix)

    for i in range(num_attributes):
        for j in range(i+1, num_attributes):
            if correlation_matrix[i][j] > max_correlation_value:
                max_correlation_value = correlation_matrix[i][j]
                max_correlation_indices = (i, j)

    return max_correlation_indices, max_correlation_value
```

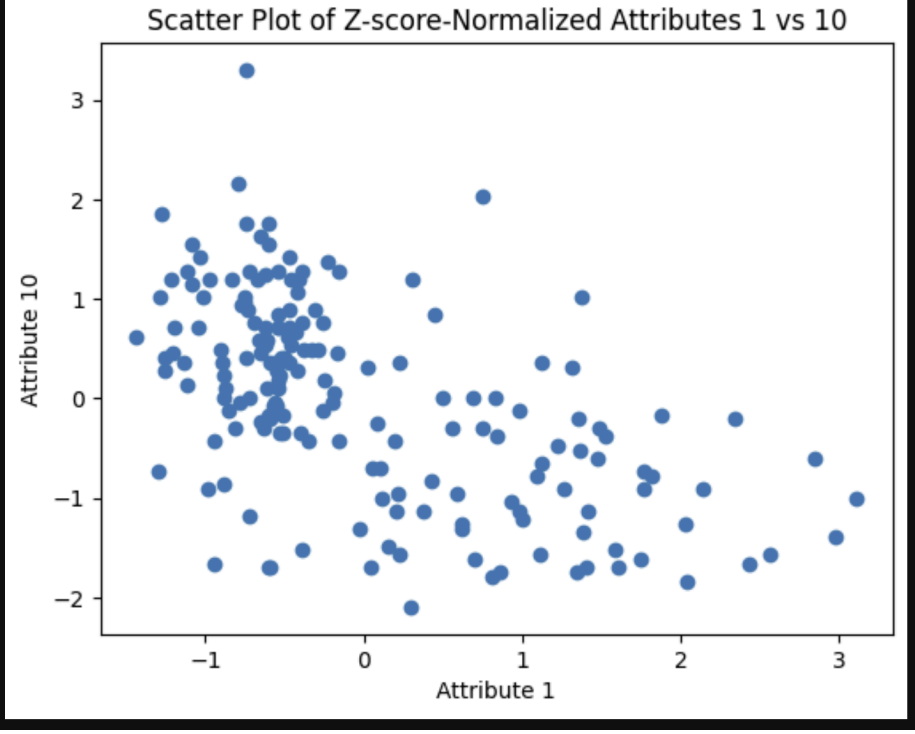
```
Z-score-normalized attributes with the greatest correlation: (6, 7)
Correlation coefficient: 0.8694480396436748
```




(Michael B.) From the scatter plot and the output of the code, we can see that the **two Z-score-normalized attributes with the smallest correlation** are attribute 1 and attribute 10 which are: Alcohol and Color Intensity. Their correlation coefficient is -0.5644668507477977. This negative correlation suggests that as the alcohol content of the wine increases, its color intensity tends to decrease, and vice versa.

```python
def find_smallest_correlation(data_path):
    """Finds which Z-score-normalized numerical attributes have the smallest correlation."""
    data = read_csv_data(data_path)
    normalized_data = z_score_normalize(data)
    correlation_matrix = compute_correlation_matrix(normalized_data)

    min_correlation_value = float('inf')
    min_correlation_indices = None
    num_attributes = len(correlation_matrix)

    for i in range(num_attributes):
        for j in range(i+1, num_attributes):
            if correlation_matrix[i][j] < min_correlation_value:
                min_correlation_value = correlation_matrix[i][j]
                min_correlation_indices = (i+1, j+1)  # Increment by 1 to account for skipped first column

    return min_correlation_indices, min_correlation_value
```

```
Z-score-normalized attributes with the smallest correlation: (1, 10)
Correlation coefficient: -0.5644668507477977
```

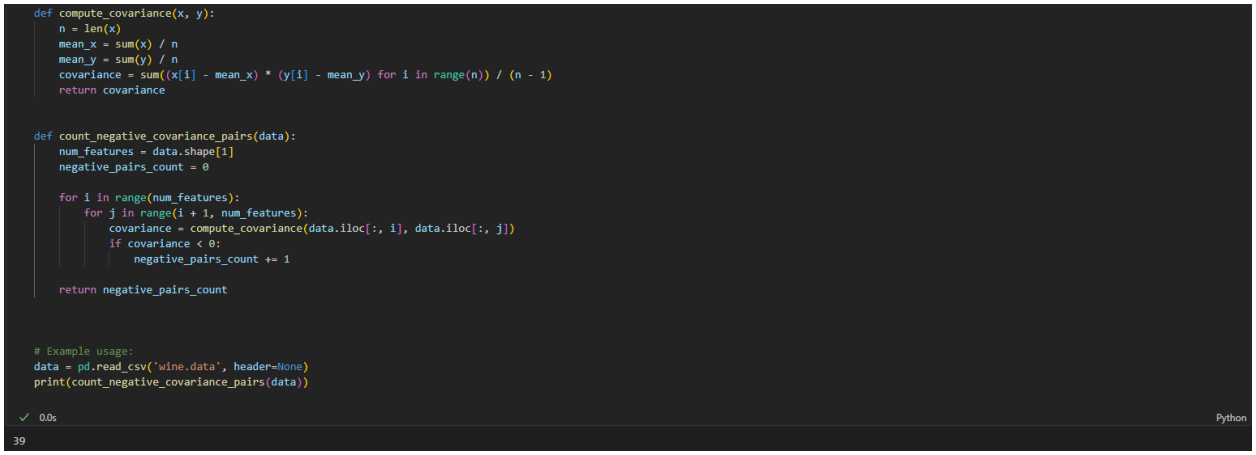



(Megan S.) There are 11 pairs of features that have **correlation greater than or equal to 0.5**. To find how many pairs of features have correlation greater than or equal to 0.5, we added the code below that can be used with the correlation function. Therefore, we found 11 pairs of features that have a 0.5 correlation or greater.

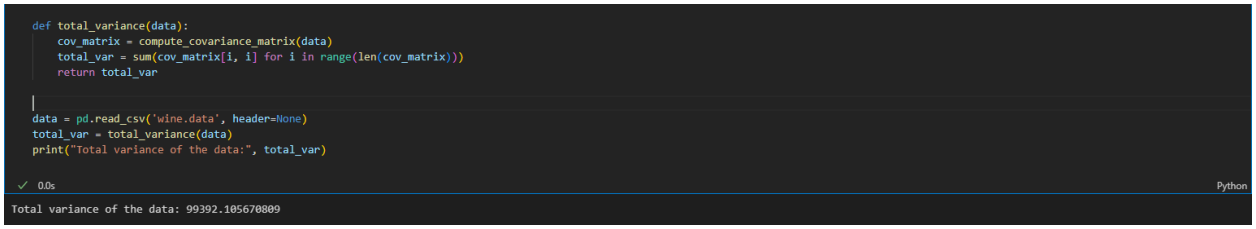
```python
def count_high_correlation_pairs(correlation_matrix, threshold=0.5):

    num_features = len(correlation_matrix)
    num_high_correlation_pairs = 0

    for i in range(num_features):
        for j in range(i+1, num_features):  # Avoid counting pairs twice
            if correlation_matrix.iloc[i, j] >= threshold:
                num_high_correlation_pairs += 1

    return num_high_correlation_pairs

data_set = pd.read_csv('/Users/megansteinmasel/Desktop/wine/wine-one.data', header=None)

# Compute the correlation matrix
correlation_matrix = compute_correlation_matrix(data_set)

# Count the number of pairs of features with correlation greater than or equal to 0.5
num_pairs = count_high_correlation_pairs(correlation_matrix)
print("Number of pairs of features with correlation greater than or equal to 0.5:", num_pairs)
```

```
Number of pairs of features with correlation greater than or equal to 0.5: 11
```

(Rohan K.) There are a total of **39 pairs of features** within the dataset exhibiting negative sample covariance. This observation indicates an inverse relationship between these pairs of variables: when one feature increases, the other tends to decrease, and vice versa. Such negative covariance between features is valuable for understanding the interdependencies within the dataset, potentially offering insights into underlying patterns and relationships




(Rohan K.) The **total variance** of the dataset amounts to 99392.105670809. It encompasses the variability present within the entire dataset, taking into account the individual variances of each feature as well as their covariances. In this case, the calculated value of 99392.105670809 serves as a quantitative measure of the overall variability present within the dataset, offering an understanding of its characteristics and dynamics.




(Rohan K.) The **total variance** of the data, constrained to the five features with the highest sample variance, amounts to 98830.11958419316. By focusing on the features with the greatest sample variance, we can effectively capture a substantial portion of the overall variability present in the data.