TEAM ID : H24 MEMBER : Yujin Kim, ykim68

ALDA Fall 2025
HW1
Due Date: 09/03/2025

HW1 contains 7 questions. Please read and follow the instructions.

- **DUE DATE FOR SUBMISSION: 09/03/2025 11:45 PM**
- **TOTAL NUMBER OF POINTS: 130** (+5 bonus points if you follow all the instructions and 0 otherwise)
- **NO PARTIAL CREDIT** will be given so provide concise answers.
- **You MUST manually add ALL team members in the submission portal when you submit through Gradescope. One submission per group. Team member(s) who are left out will lose 20 points if added after the deadline.**
- Make sure you clearly list **your homework team ID, all team members' names and Unity IDs, for those who have contributed to the homework contribution** at the top of your submission.
- The materials on this course website are only for use of students enrolled in this course and **MUST NOT** be retained or disseminated to others.
- By uploading your submission, you agree that you have not violated any university policies related to the student code of conduct (`https://policies.ncsu.edu/policy/pol-11-35-01/`), and you are signing the Pack Pledge: **"I have neither given nor received unauthorized aid on this test or assignment"**.

TEAM ID : H24 MEMBER : Yujin Kim , ykim68

[GradeScope and GitHub Submission Instructions for Coding Questions]: For coding questions, you must submit your written solutions as a PDF on GradeScope and also provide your code in a private GitHub repository with instructor access.

1. **Create a private GitHub repository for your group.** Name your repository using the following format:

   `ncsu-engr-ALDA-Fall2025-HXX`

   Replace `XX` with your homework group number. **Example:** `ncsu-engr-ALDA-Fall2025-H2`

   You will use one homework repository for the whole semester: **do not** create a new repository for each homework.

2. **Set up the repository:**
   - Log in to GitHub and click the green **"New"** button.
   - Use the required naming convention.
   - Set the repository to **private**.
   - Click **"Create repository"**.

3. **Add collaborators:**
   - Navigate to **Settings** > **Collaborators**.
   - Add all group members.
   - **Add the TAs.**

4. **Organize your code:**
   - Create separate folders for each of five assignments (e.g., `HW1`, `HW2`, etc.).
   - Place all relevant code in the correct folder.
   - Upload your complete `.py` files **before the submission deadline**. Late or missing code will not be graded.

5. **Include your GitHub repository link in the GradeScope PDF.**

6. **Clearly reference your code in the PDF:**
   - Indicate the relevant file for each question.
     Example: "The solution to Question 2 is in `matrix.py`."
   - Code must be executable; outputs must be generated by running the file.
   - **Do not hardcode results.** Submissions without proper computation will receive no credit.

7. **Code excerpts and outputs:** Your PDF must include output for each part and key code snippets (not full scripts) that demonstrate your implementation logic.

TEAM ID : H24 MEMBER : Yujin kim, ykim68

1. (20 points) [**Data Attributes**] [**Graded by Rajesh Debnath**] Classify the following attributes as:
   1) nominal, ordinal, interval, or ratio; **and** 2) as binary, discrete, or continuous. Some cases may have more than one interpretation, so briefly justify your answer if you think there may be some ambiguity.

   Sample:
   Attribute: Number of cars in a household
   Expected Answer: Ratio, Discrete

   (a) (1 point) Income earned in a week : Ratio, Continuous
   (b) (1 point) Month of a year (e.g. January, February, March, etc.) : Ordinal, Discrete
   (c) (1 point) Temperature in degrees Fahrenheit : Interval, Continuous
   (d) (1 point) Temperature in degrees Kelvin : Ratio, Continuous
   (e) (1 point) Years of work experience : Ratio, Discrete
   (f) (1 point) The pH of a substance : Interval, Continuous
   (g) (1 point) A random 10-digit number used as an identifier : Nominal, Discrete
   (h) (1 point) The amount of Calories found on a nutrition label : Ratio, Continuous
   (i) (1 point) The distance traveled on a road trip measured in miles : Ratio, continuous
   (j) (1 point) Percentage of daily vitamin C intake from a glass of orange juice. : Ratio, Continuous
   (k) (1 point) IQ scores : Interval, Continuous
   (l) (1 point) Restaurant rating: 1, 2, 3, 4, or 5 stars (remember that the difference between 1 and 2 stars is not necessarily the same as the difference between 4 and 5 stars) : ordinal, Discrete
   (m) (1 point) Circumferences of tree trunks : Ratio, Continuous
   (n) (1 point) True or False : Nominal, Binary
   (o) (1 point) Clothing size (XS, S, M, L, XL) : Nominal, Discrete
   (p) (1 point) CPU processing speed (actual or base) measured in Gigahertz : Ratio, Continuous
   (q) (1 point) A person's height measured in feet and inches : Ratio, Continuous
   (r) (1 point) Blood type (O+, O-, A+, A-, B+, B-, AB+, AB-) : Nominal, Discrete
   (s) (1 point) Letter grade in a class (E.g. A+, A, A-, B+, B, etc.) : Ordinal, Discrete
   (t) (1 point) Placement in a race (e.g. 1st, 2nd, 3rd, etc.) : Ordinal, Continuou

2. (15 points) [**Coding - Matrix Operations**] [**Graded by Rajesh Debnath**] Write Python code to complete each of the tasks listed below. You must use NumPy for all numerical computations.

   Follow the instructions under "GradeScope and GitHub Submission Instructions for Coding Questions" (see page 2).

   (a) (1 point) Generate a 4*4 matrix, **A**, consisting entirely of ones.

   (b) (1 point) Modify **A** so that all elements in its second column are replaced with the value 5.

   (c) (1 point) Compute the sum of each row in **A** and store the results in a one-dimensional array **s**.

   (d) (2 points) Append **s** as the last column of **A** to make it a matrix with 5 columns.

   (e) (2 points) Transpose the updated matrix **A** ($\mathbf{A} = \mathbf{A}^T$).

   (f) (2 points) Generate a matrix **B** such that 1) its first row contains the standard deviation of each row of A, and 2) the second row contains random numbers drawn uniformly from [0, 1]. Use np.random.seed(2025) for reproducibility.

   (g) (2 points) Compute the matrix product: $\mathbf{C} = \mathbf{B} \cdot \mathbf{A}$.

   (h) (2 points) Let $\mathbf{X} = [2, 4, 6, 8]^T$, $\mathbf{Y} = [12, 9, 6, 3]^T$, $\mathbf{Z} = [5, 10, 15, 10]^T$. Compute the variance of each vector separately, and then calculate the Pearson correlation coefficient between **X** and **Y**.

   (i) (2 points) Let $\mathbf{x} = [8, 15, 10, 12, 20, 18, 14]^T$. Verify the equation: $\bar{\mathbf{x}^2} = (\bar{\mathbf{x}}^2 + \sigma^2(\mathbf{x}))$ for the two cases:

      i. $\sigma(\mathbf{x})$ is the **population** standard deviation.

      ii. $\sigma(\mathbf{x})$ is the **sample** standard deviation.

   In both cases, show detailed calculations and justify the results using Python (You may use the *math* library)

2. Write Python code to complete each of the tasks listed below.

In [48]:
```
import numpy as np
```

(a) Generate a 4*4 matrix, A, consisting entirely of ones.

In [49]:
```
A = np.ones((4,4), dtype=float)
print("(a) = ", A)
```

```
(a) =  [[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

(b) Modify A so that all elements in its second column are replaced with the value 5.

In [50]:
```
A[:, 1] = 5
print("(b) = ", A)
```

```
(b) =  [[1. 5. 1. 1.]
 [1. 5. 1. 1.]
 [1. 5. 1. 1.]
 [1. 5. 1. 1.]]
```

(c) Compute the sum of each row in A and store the results in a one-dimensional array s.

In [51]:
```
s = A.sum(axis=1)
print("(c) = ", s)
```

```
(c) =  [8. 8. 8. 8.]
```

(d) Append s as the last column of A to make it a matrix with 5 columns.

In [52]:
```
A = np.column_stack([A, s])
print("(d) = ", A)
```

```
(d) =  [[1. 5. 1. 1. 8.]
 [1. 5. 1. 1. 8.]
 [1. 5. 1. 1. 8.]
 [1. 5. 1. 1. 8.]]
```

(e) Transpose the updated matrix A (A = AT)

In [53]:
```
A = A.T
```

```
print ("(e) A = ", A)
```

```
(e) A =  [[1. 1. 1. 1.]
 [5. 5. 5. 5.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [8. 8. 8. 8.]]
```

(f) Generate a matrix B such that

1. its first row contains the standard deviation of each row of A, and
2. the second row contains random numbers drawn uniformly from [0, 1]. Use np.random.seed(2025) for reproducibility.

In [54]:
```
row_first = np.std(A, axis=1, ddof=0)
np.random.seed(2025)
row_second = np.random.uniform(0, 1, size=A.shape[0])
B = np.vstack([row_first, row_second])
print("(f) B = ", B)
```

```
(f) B =  [[0.         0.         0.         0.         0.        ]
 [0.13548816 0.8878517  0.93260564 0.44556816 0.38823555]]
```

(g) Compute the matrix product: C = B*A

In [55]:
```
C = B @ A
print("(g) C = ", C)
```

```
(g) C =  [[0.         0.         0.         0.        ]
 [9.05880485 9.05880485 9.05880485 9.05880485]]
```

(h) Let X = [2,4,6,8].T, Y = [12, 9, 6, 3].T, Z = [5, 10, 15, 10].T. Compute the variance of each vector seperately, and then calculate the Pearson correlation coefficient between X and Y.

In [56]:
```
X = np.array([2, 4, 6, 8], dtype = float)
Y = np.array([12, 9, 6, 3], dtype = float)
Z = np.array([5, 10, 15, 10], dtype = float)

var_X = np.var(X, ddof = 0)
var_Y = np.var(Y, ddof = 0)
var_Z = np.var(Z, ddof = 0)

Pearcorr_XY = np.corrcoef(X, Y)[0, 1]

print("(h) variance X = ", var_X)
print("(h) variance Y = ", var_Y)
print("(h) variance Z = ", var_Z)
print("(h) Pearson correlation coefficient = ", Pearcorr_XY)
```

```
(h) variance X =  5.0
(h) variance Y =  11.25
(h) variance Z =  12.5
(h) Pearson correlation coefficient =  -1.0
```

(i) Let x = [8, 15, 10, 12, 20, 18, 14].T. Verify the equation: x_bar_squa = (x_bar_squa + sigma_bar_squa(x)) for the two cases: i) sigma(x) is the popluation standard deviation. ii) sigam(x) is the sample standard deviation. In both cases, show detailed calculations and justify the results using Python

In [57]:
```
x = np.array([8, 15, 10, 12, 20, 18, 14], dtype = float)

x_bar = np.mean(x)
x_bar_squa = np.mean(x**2)

## i) popluation standard deviation ##
var_pop = np.var(x, ddof = 0)
result_pop = x_bar**2 + var_pop

## ii) sample standard deviation ##
var_sam = np.var(x, ddof = 1)
result_sam = x_bar**2 + var_sam

print("x_bar_squa = ", x_bar_squa)
print("(i).  i) = ", result_pop)
print("(i). ii) = ", result_sam)
```

```
x_bar_squa =  207.57142857142858
(i).  i) =  207.57142857142858
(i). ii) =  210.16326530612247
```

The equality holds when using the population variance, but not when using the sample variance.

3. (24 points) [**Coding - Data Visualization**] [**Graded by Ian Holmes**] You are tasked with summarizing and exploring the `"iris.csv"` dataset. The dataset description can be found in the `"iris.names"` file. These files are located in the `"HW1_data"` folder.

   Follow the instructions under "GradeScope and GitHub Submission Instructions for Coding Questions" (see page 2).

   (a) (4 points) Compute the mean, median, standard deviation, range, 25th percentile, 50th percentile, and 75th percentile for the following attributes: *sepallength*, *sepalwidth*, *petallength*, and *petalwidth*.

   (b) (3 points) Make box-and-whisker plots for the attributes *sepallength* and *sepalwidth* grouped by the *class* label. Be sure to include a title for each plot to describe what feature is being displayed.

   (c) (4 points) Create histogram plots using 16 bins for the two features *petallength* and *petalwidth*, respectively.

   (d) (4 points) Create a scatter matrix of the data. Include only the following features: *sepallength*, *sepalwidth*, *petallength*, and *petalwidth*. Use the *classes* attribute to change the color of the data points (for convenience, you may use a library for this). For the diagonal of the scatter matrix, plot the kernel density estimation (KDE).

   (e) (5 points) Write code to produce a three-dimensional scatter plot using *sepallength*, *sepalwidth*, and *petallength* as dimensions, and color the data points according to the *classes* attribute.

   (f) (4 points) The quantile-quantile plot can be used for comparing the distribution of data against the normal distribution. Create a quantile-quantile plot for the two features *sepallength* and *sepalwidth*, respectively. Provide a brief analysis of the two plots.

3. summarizing and exploring the "iris.csv" dataset.

In [27]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("HW1_data/iris/iris.csv")

print(df.head())
print(df.info())
print(df.describe())
```

```
   sepallength  sepalwidth  petallength  petalwidth        class
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
       sepallength  sepalwidth  petallength  petalwidth
count   150.000000  150.000000   150.000000  150.000000
mean      5.843333    3.054000     3.758667    1.198667
std       0.828066    0.433594     1.764420    0.763161
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
75%       6.400000    3.300000     5.100000    1.800000
max       7.900000    4.400000     6.900000    2.500000
```

(a) Compute the mean, median, standard deviation, range, 25th percentile, 50th percentile, and 75th percentile for the following attributes: sepallength, sepalwidth, petallength, petalwidth

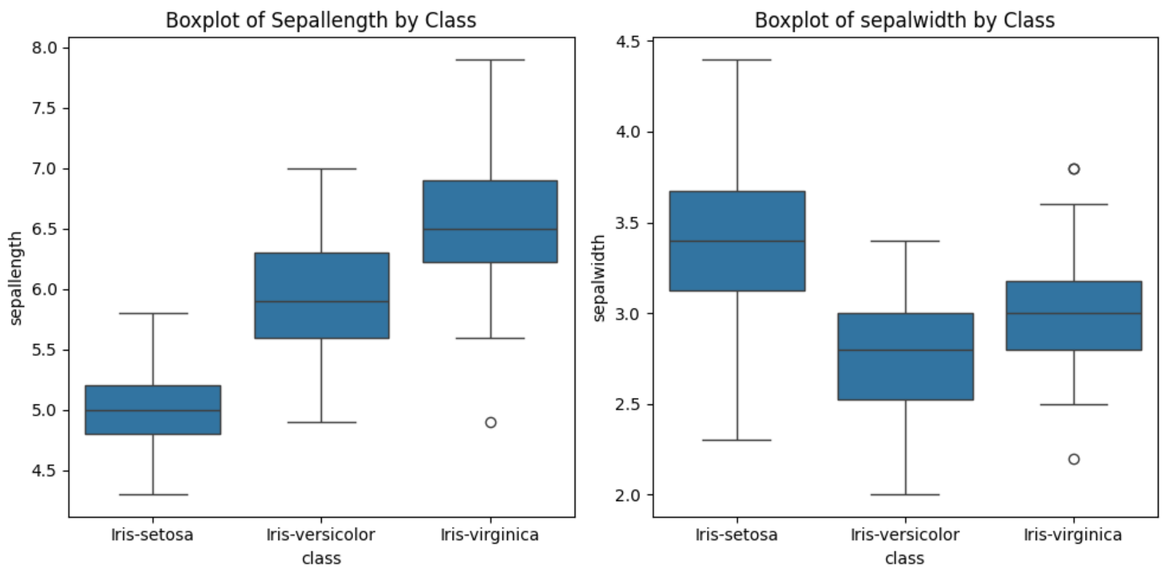
In [28]:
```
features = ["sepallength", "sepalwidth", "petallength", "petalwidth"]
agg_stats = df[features].agg(["mean", "median", "std", "min", "max"]).

q = df[features].quantile([0.25, 0.50, 0.75]).T
```

```
agg_stats["25%"] = q[0.25]
agg_stats["50%"] = q[0.50]
agg_stats["75%"] = q[0.75]

agg_stats["range"] = agg_stats["max"] - agg_stats["min"]

summary = agg_stats[["mean", "median", "std", "range", "25%", "50%", "
print(summary)
```

```
                 mean  median       std  range  25%   50%  75%
sepallength  5.843333    5.80  0.828066    3.6  5.1  5.80  6.4
sepalwidth   3.054000    3.00  0.433594    2.4  2.8  3.00  3.3
petallength  3.758667    4.35  1.764420    5.9  1.6  4.35  5.1
petalwidth   1.198667    1.30  0.763161    2.4  0.3  1.30  1.8
```

(b) Make box-and-whisker plots for the attributes sepallength and sepalwidth grouped by the class label. Be sure to include a title for each plot to describe what feature is being displayed.

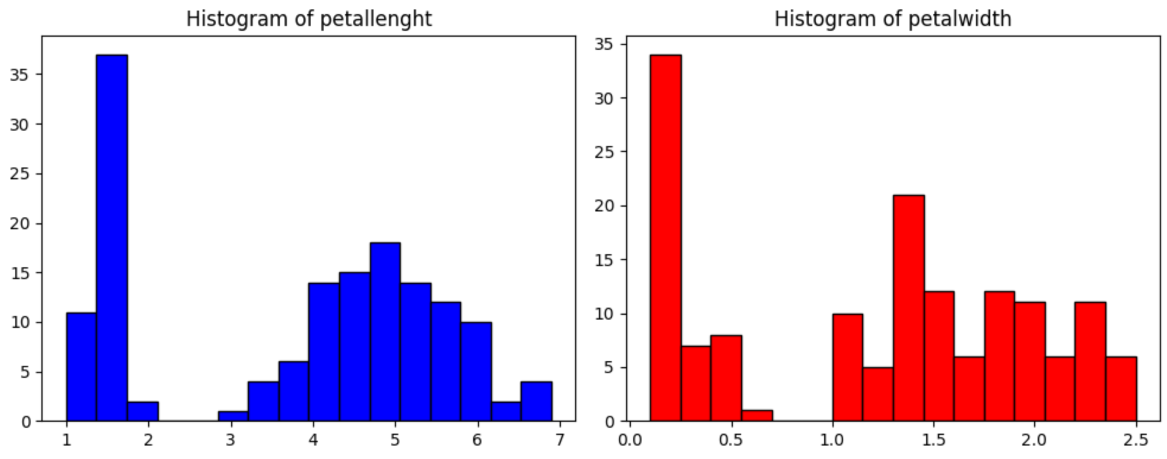
In [29]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
sns.boxplot(x="class", y="sepallength", data=df)
plt.title("Boxplot of Sepallength by Class")

plt.subplot(1, 2, 2)
sns.boxplot(x="class", y="sepalwidth", data=df)
plt.title("Boxplot of sepalwidth by Class")

plt.tight_layout()
plt.show()
```



(c) Create histogram plots using 16 bins for the two features petallenght and petalwidth, repectively.

In [30]:
```
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.hist(df["petallength"], bins=16, color="blue", edgecolor="black")
plt.title("Histogram of petallenght")

plt.subplot(1, 2, 2)
plt.hist(df["petalwidth"], bins=16, color="red", edgecolor="black")
plt.title("Histogram of petalwidth")

plt.tight_layout()
plt.show
```

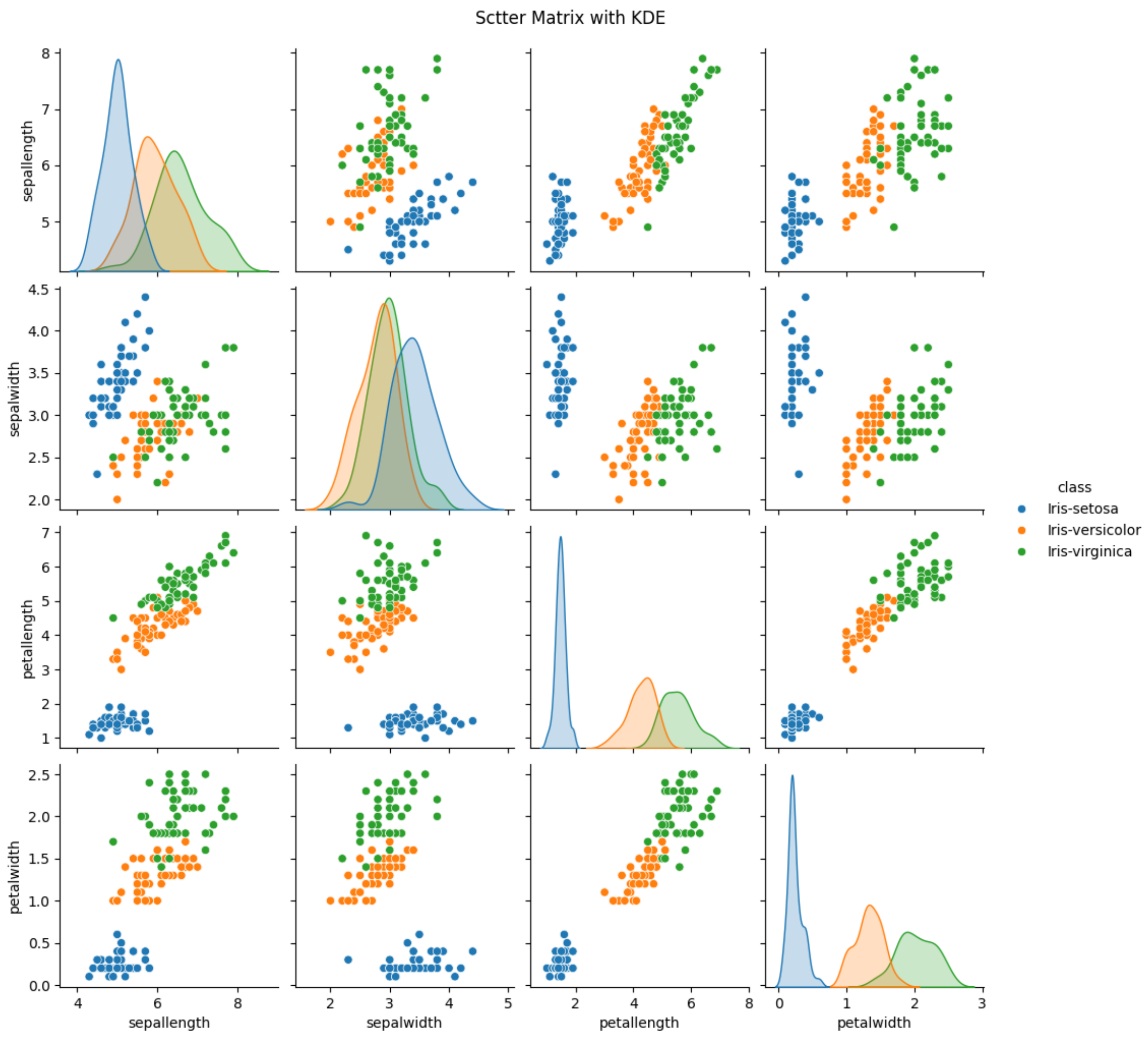
Out[30]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



(d) Create a scatter matrix of the data. Include only the following features: sepallength, sepalwidth, petallenght, and petalwidth. Use the classes attribute to change the color of the data points (for convenience, you may use a library for this). For the diagonal of scatter matrix, plot the kernel density estimation (KDE).

https://seaborn.pydata.org/generated/seaborn.pairplot.html

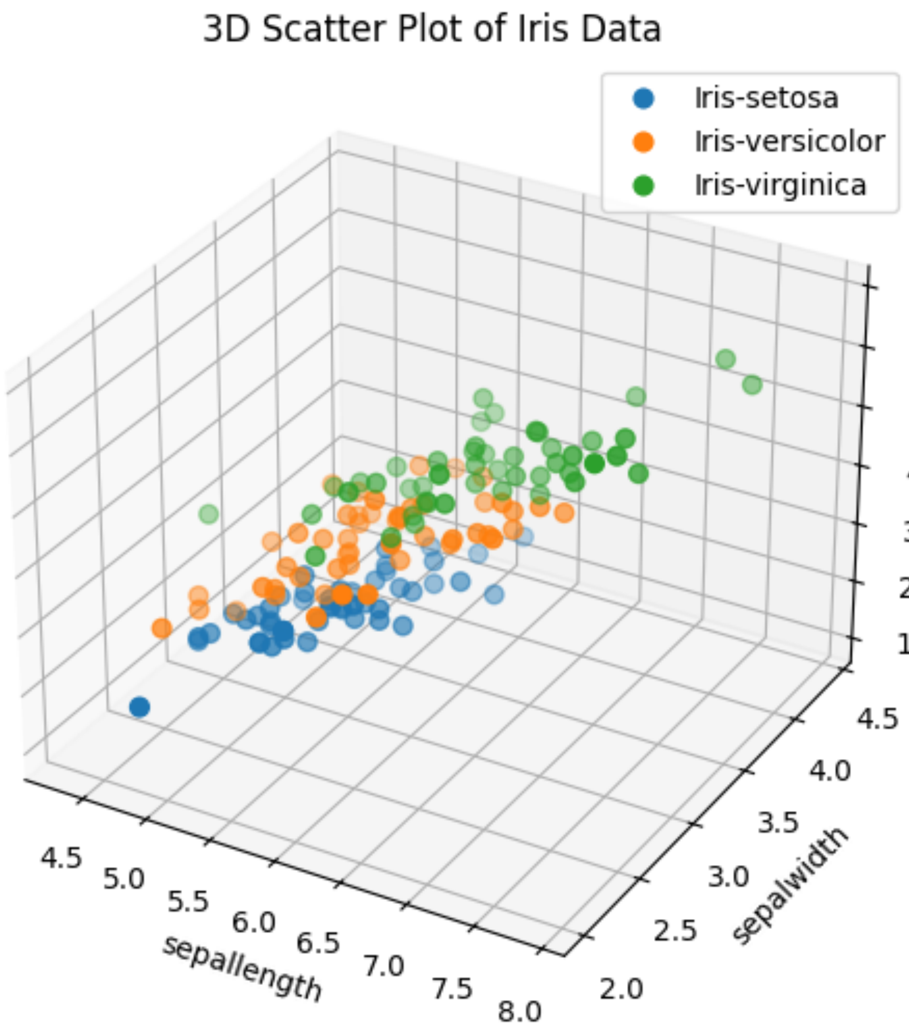
In [31]:
```
sns.pairplot(df[features + ["class"]], hue="class", diag_kind="kde")
plt.suptitle("Sctter Matrix with KDE", y=1.02)
plt.show()
```



(e) Write code to produce a three-dimensional scatter plot using sepallength, sepalwidth, and petallength as dimensions, and color the data points according to the classes attribute.

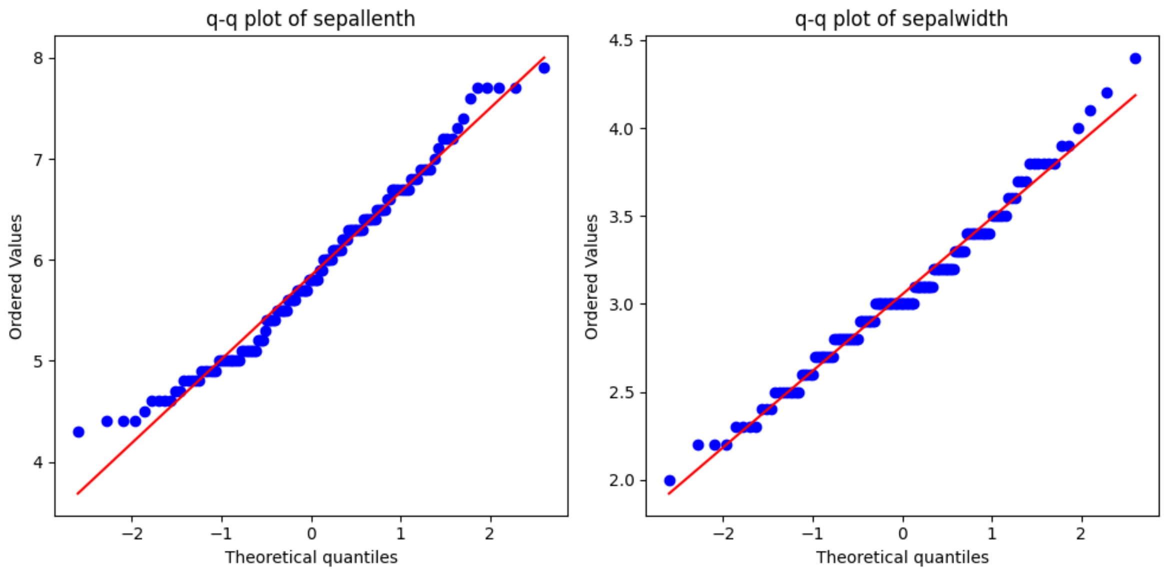
In [32]:
```python
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection="3d")
colors = plt.cm.tab10.colors

for i, c in enumerate(df["class"].unique()):
    subset = df[df["class"] == c]
    ax.scatter(
        subset["sepallength"],
        subset["sepalwidth"],
        subset["petallength"],
        marker = 'o',
        color = colors[i],
        label=c,
        s=40
    )
```

```
ax.set_xlabel("sepallength")
ax.set_ylabel("sepalwidth")
ax.set_zlabel("petallength")

plt.title("3D Scatter Plot of Iris Data")
plt.legend()
plt.show()
```



(f) The quantile-quantile plot can be used for comparing the distribution of data against the normal distribution. Create a quantile-quantile plot for the two features sepallength and sepalwidth, respectively. Provide a brief analysis of the two plots.

In [33]:
```
import scipy.stats as stats

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
stats.probplot(df["sepallength"], dist="norm", plot=plt)
plt.title("q-q plot of sepallenth")
plt.subplot(1, 2, 2)
stats.probplot(df["sepalwidth"], dist="norm", plot=plt)
plt.title("q-q plot of sepalwidth")

plt.tight_layout()
plt.show()
```



The closer the points are to the diagonal, the closer the data follow a normal distribution. sepallength deviates from normality more than sepalwidth.

4. (16 points) [**Short Answer Questions**] [**Graded by Ian Holmes**]

   (a) (10 points) Please review lecture notes to answer the following short Answer

      i. (3 points) Which distance metric would best describe how far a drone is from its target location in a three-dimensional space? Justify your answer.

      ii. (3 points) What is the definition of correlation? If variables X and Y have a correlation of -0.85 while variables Y and Z have a correlation of 0.5, what claims can you draw? Can we say anything about the correlation between X and Z? Justify your answer.

      iii. (4 points) You are working with data that comes from sensors which trigger whenever they detect a wind gust. The sensors then record the wind speed (mph) and direction/angle (between 0 and 360 degrees). The sensors are supposed to only log each wind gust once, which they do by turning off for 5 seconds after recording a gust. You suspect that this behavior is failing sometimes, and that they are recording the same gust multiple times. The sensors record the wind speed and angle to a very high precision, so even if a repeat measurement is taken of a certain gust, the measurements will be very slightly different. If you don't have any information on the order in which the entries were recorded, nor what time they were recorded at, what's a good way to figure out which data entries are repeat measurements of the same gust?

   (b) (6 points) Data Quality

      i. (2 points) In your own words, explain what is noise. Can noise ever be desirable? If so, give an example. If not, provide an explanation of why not.

      ii. (2 points) In your own words, explain what is an outlier. How could outliers be detected? How do outliers differentiate from noise?

      iii. (2 points) Suppose you are analyzing a dataset of students' exam scores for a particular course. As you explore the data, you notice that there are a few instances where students have exceptionally high or low scores that are significantly different from the rest of the dataset. Upon closer examination, you find that these extreme scores are the result of data entry errors made during the recording process. For example, a student may have mistakenly been assigned a score of 1000 instead of 100. In addition to these extreme values, you also observe that there are some scores that slightly deviate from the expected range, but they are not related to any specific data entry errors or exceptional circumstances. These deviations occur sporadically throughout the dataset. Based on this scenario, please answer the following question: Are the extreme scores (high and low) outliers or noisy data? What about the slightly deviating scores? Justify your answer.

4(a)-i) Euclidean Distance. It directly measures the straight-line distance between two points 'p' and 'q' in 3-dimensional space.
This reflects the actual physical distance between drone and its target.

$$Dist = \sqrt{\sum_{k=1}^{d} (p_k - q_k)^2}$$ d: number of dimensions

-ii) Correlation measures the strength and direction of the linear relationship between 2 variables. $corr = \frac{Cov(X.Y)}{VAR(X) \cdot VAR(Y)}$

0.85 indicates a strong negative correlation, and 0.5 indicates a moderate positive correlation. X and Y move strongly in opposite directions, and Y and Z tend to move in the same direction.

-iii) we can use [similarity / Dissimilarity] measure, which indicate how similar or different 2 data object are. The choice depends on whether the data are nominal, ordinal, interval or ratio. Since both wind direction and wind speed are ratio values, dissimilarity can be evaluated using $d = |p-q|$ and similarity using $s = \frac{1}{1+d}$ or $s = 1 - \frac{d_{min_d}}{max_d min_d}$

(b)-i) Noise refers to distorted values in data caused by random error or fluctuations, where the observed values are altered due to interference during the measurement process. In general, noise is undesirable, but it can sometimes be intentionally added to reduce overfitting and improve generalization. Therefore, in developing models with scalability in mind, the use of noise can actually be beneficial.

-ii) Outliers are value in real data that deviate significantly from the general distribution or pattern. They can be detected through statistical techniques such as Gaussian distribution-based methods or through visualization. Unlike noise, outliers are not errors but may represent true extreme values, Which can hold important meaning depending on the purpose of the analysis

iii) The extreme score (a score of 1,000 recorded instead of 100) are noise data.
These values result from data entry errors and do not represent true observation. Therefore, they should be removed.
The slightly devitating scores should be considered outliers. These values are not directly linked to input errors or special circumstances and may represent genuine but extreme observations. Although, they deviate from the expected range, they can hold meaningful information depending on the analysis objective and thus should be carefully evaluated rather than discarded.

5. (9 points) [**Sampling**] [**Graded by Ian Holmes**] Answer the following questions:
   (a) (5 points) A chess dataset contains records for 10,000 unique games, where 5,000 games result in a win for player white and 5,000 games result in a win for player black. Among the 10,000 unique games, 3% witness an en passant move and 11% have a castling move occur at least once on either the King's side or Queen's side. For simplicity, assume that the two events - en passant or castling - are mutually exclusive. Suppose we are developing a classifier in hopes of predicting the outcome of a chess game based on whether we see these moves. However, we are unable to use the entire data set due to computational limitations, and thus can only use a sample of the entire data set. Which sampling method would be appropriate and why? If we are sampling 2,000 games from the provided dataset, how many games should be selected from each group using your choice of sampling methods? Briefly justify your answer.
   (b) (4 points) Consider the following scenario: 0.5% of the total population are allergic to dairy products. Are they noise or outliers? Briefly justify your answer and describe the differences between noise and outliers.

(a) 50 / 50. → win rate : 0.5 en passant ⟷ castling : $E \cap C = 0$

En passant = 3% (300/10,000)
Castling = 11% (1,100/10,000)
No E & C = 86% (8,600/10,000)

→ stratified Sampling [Same number from each group
2,000 └ numbe drawn proportional to group size

{ En passant : $0.03 \times 2{,}000 = 60$
Castling : $0.11 \times 2000 = 220$
No E&C : $0.86 \times 2000 = 1720$

(b) Outliers.
It is a rare group (0.5%), not a measurement/input error (noise).
Depending on the purpose of the analysis, it is a meaningful observation and therefore, should not be removed.

Noise : Random errors in the measurement/input process that distort values → subject to removal
Outlier : A true value that lies far from the distribution → may have value to preserve.

6. (16 points) Data Transformation. [**Graded by Tural Mehtiyev**]

   (a) Please identify the appropriate data transformation methods for the following situations. Give a brief description about your answers:

      i. (3 points) Consider a dataset containing information about student performance in two subjects: Math and English. The Math scores range from 89 and 100 (mean = 93, standard deviation = 2), while the English scores range from 54 to 88 (mean = 68, standard deviation = 4).
      1) For each subject, apply normalization (transformed data has: $x' \in [0, 1]$) and calculate the new mean and new standard deviation of the normalized subject, then compare their means and standard deviations.
      2) For each subject, apply standardization to it and show the range of transformed data, then compare the standardized ranges of both subjects.

      ii. (1 point) You have a set of numbers and do the following: first, you find the absolute max of the data, i.e. if it was -9, 2, and 8, the absolute max is 9. Then, you divide every number by this absolute max. Compare this operation to normalization and standardization, and give an example of when it might be useful.

      iii. (4 points) During the design of an artificial neural network, we sometimes need to transform a variable x that has a range of $(-\infty, \infty)$ to an open set $z \in (-1, 1)$. Note that z monotonically increases as x increases in this transformation. Please specify a proper function for such transformation.

   (b) In natural language processing (NLP), there are diverse ways to represent words such as one-hot encoding, bag of words, TF*IDF, and distributed word representations. In **one hot encoding**, a bit vector whose length is the size of the vocabulary of words is created, where only the associated word bit is on (i.e., 1) while all other bits are off (i.e., 0). Here is a toy example: suppose there is a 5-dimensional feature vector to represent a vocabulary of five words: [king, queen, man, woman, power]. In this case, 'king' is encoded into [1,0,0,0,0], 'queen' is encoded into [0,1,0,0,0], etc. Due to the nature of this representation, the feature vector encodes the vocabulary of a sentence where all words are equally distant. On the other hand, in **distributed word vectors**, a real-valued vector whose length is defined by *some common properties of words* is created, then each word can be represented as a linear combination of the defined properties. Using the toy example above, given a 3-dimensional feature vector of [man, woman, power] as the common properties, then words such as 'king', 'queen', 'man', and 'woman' could be encoded into [0.9, 0, 0.8], [0, 0.9, 0.8], [0.99, 0, 0.5], and [0, 0.99, 0.5], respectively. In this case, if you subtract a vector of 'man' from a vector of 'king', and add a vector of 'woman', then you will get a vector close to a vector of 'queen'.

      i. (4 points) You'd like to come up with a combination of these two approaches. You have the idea to use vectors of word properties like with distributed word vectors, but with a 0 or 1 for each property instead of some real value (i.e. "queen" becomes [0, 1, 1] instead of [0, 0.9, 0.5]). Give one non-computational (not about computation space or time) advantage or disadvantage of this approach compared to one-hot encoding, and one non-computational advantage or disadvantage compared to distributed word vectors.

ii. (4 points) You have a second new idea for an approach: still using one-hot encoding, you want to allow more than one index of the vector to be set to 1 so as to allow more words to be represented (i.e. if we are using 5-dimensional features vectors, we could have [1, 0, 0, 0, 1] be "throne"). The ordering of the indices here are not necessarily meaningful except for distinguishing them (i.e. think of the vectors as binary nominal data). Now consider this approach and the previous approach you just came up with from (i): using the same vector sizes, do you think one approach would allow you to store more distinct words than the other? Justify your answer.

(a) - i) 1. Normalization

- Math (range : 100 - 89 = 11)

mean : $\mu_M' = \frac{93-89}{11} = \frac{4}{11} = 0.3636$

standard deviation : $\sigma_M' = \frac{2}{11} = 0.1818$

- English (range : 88 - 54 = 34)

mean : $\mu_E' = \frac{68-54}{34} = \frac{14}{34} = 0.4118$

standard deviation : $\sigma_E' = \frac{4}{34} = 0.1176$

Comparison : Originally, the distribution of Math score had a higher mean and smallor spread compared to English. After normalization, both subjects are rescaled to the same scale, so the differences in mean and standard deviation are no longer prominent.

2. Standardization

· Math : $\mu = 93$, $\sigma = 2$

min : $\frac{89-93}{2} = -2$

max : $\frac{100-93}{2} = 3.5$

range : $[-2, 3.5]$

· English : $\mu = 68$, $\sigma = 4$

min : $\frac{54-68}{2} = -3.5$

max : $\frac{88-68}{2} = 5$

range : $[-3.5, 5]$

Comparison : Math originally had a smaller variance while English had a larger one, but after standardization both subjects are adjusted to the same range, making the relative differences disapear and leaving only the shape of the distributions

a) ii) $x = \frac{x}{\max |x|}$

- Comparison with Normalization : Min-Max scaling uses both the minimum and maximum values to map data into $[0, 1]$, whereas max-abs scaling does not use the minimum and instead preserves the sign, typically mapping values into $[-1, 1]$.

- Comparison with Standardization : Standardization forces the data have mean "0" and standard deviation "1", but max-abs scaling does not fix the mean or variance.

- Useful : Max-abs scaling can be helpful for sparse, zero-centered features.

(a) - iii) $x : (\infty \to \infty) \to (-1, 1)$
monotonically increase

(b) - i) One-hot encoding : The dimensionality grows with the vocabulary size, often reaching 10 or even 100, which makes the representation very sparse.
pros : when dimensionality is reduced, the representation becomes easier to interpret, and both computation and storage efficiency improve.
Cons : The expressive power is limited, since it depends on how the properties are defined.

distributed word vectors : Each property dimension is assigned a real value, which allows for capturing nuances and intensity of meaning between words.

pros : Because the properties are explicit, the representation is more interpretable for human.

cons : It cannot fully reflect subtle semantic differences between words.

(b) - ii) Both approach theoretically allow the same number of district words to be stored.
↳ both rely purely on binary combination.

i) Each index corresponds to a specific "property", so not every binary combination makes sense as a real word.
→ allows fewer actual words due to semantic constrains but easier to interpret.

ii) The indices are just nominal "labels", so binary combination is possible
→ allows the maximum number of actual words since there are no constraints, but it is harder to interpret.

7. (30 points) [**Distance**] [**Graded by Tural Mehtiyev**] For this exercise, use the provided file "wine.zip", which includes wine.data and wine.names. These files contain a list of 178 data instances. There are 7 attributes, including the class attribute; please refer to the included link for documentation. For this exercise, we will only be concerned with a select few – namely, the *Flavanoids* and *Proline* attributes. Write code in Python to perform the following tasks, please report your output and relevant code in the document file.

   Note: You must include your output and the relevant key codes in the pdf submitted through Gradescope and must upload the original .py code on NCSU GitHub before the submission deadline.

   (a) (4 points) Data is not always readily available in .csv files, and sometimes must be appropriately formatted to facilitate data manipulation or analysis. Therefore, parse the .data file into a more accessible representation e.g. a Pandas DataFrame. You may consider beginning by examining the provided .data file for useful patterns to use with a regular expression.

   Then, generate a plot between the *Flavanoids* and the *Proline* of the observations. Label the axes (*Flavanoids* should be x-axis and *Proline* should be y-axis). Call this plot "Flavanoids and Proline Data". What general interpretation can you make from this plot?

   (b) (2 points) Compute the mean of the attributes *Flavanoids* and *Proline*. Define a data point called P such that $P = (\texttt{mean}(\textit{Flavanoids}), \texttt{mean}(\textit{Proline}))$.

   (c) (10 points) Compute the distance between P and the 178 data points using the following distance measures: 1) Euclidean distance, 2) Manhattan block metric, 3) Minkowski metric (for power=7), 4) Chebyshev distance, and 5) Cosine distance. List the closest 6 points for each distance.

   (d) For each distance measure, identify the 5 points from the dataset that are the closest to the point P from (b). (You are allowed to use any package functions to calculate the distances.)

      i. (10 points) Create plots, one for each distance measure. Place P on the plot and mark the 5 closest points. To mark them, you could use different colors or shapes. Make sure the points can be uniquely identified.

      ii. (4 points) Verify if the set of points is the same across all the distance measures. If there is any big difference, briefly explain why it is.

7. "win.zip" exercise

In [43]:
```
import pandas as pd

df = pd.read_csv("HW1_data/wine/wine.data", header = None)

print(df.head())
print(df.shape)
print(df.dtypes)
```

```
   0      1     2     3     4    5     6     7     8     9     10    11  \
12  \
0  1  14.23  1.71  2.43  15.6  127  2.80  3.06  0.28  2.29  5.64  1.04
3.92
1  1  13.20  1.78  2.14  11.2  100  2.65  2.76  0.26  1.28  4.38  1.05
3.40
2  1  13.16  2.36  2.67  18.6  101  2.80  3.24  0.30  2.81  5.68  1.03
3.17
3  1  14.37  1.95  2.50  16.8  113  3.85  3.49  0.24  2.18  7.80  0.86
3.45
4  1  13.24  2.59  2.87  21.0  118  2.80  2.69  0.39  1.82  4.32  1.04
2.93

     13
0  1065
1  1050
2  1185
3  1480
4   735
(178, 14)
0       int64
1     float64
2     float64
3     float64
4     float64
5       int64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
12    float64
13      int64
dtype: object
```

In [44]:
```
with open("HW1_data/wine/wine.names", "r") as f:
    for i in range(30):
        print(f.readline())
```

```
1. Title of Database: Wine recognition data

        Updated Sept 21, 1998 by C.Blake : Added attribute information



2. Sources:

   (a) Forina, M. et al, PARVUS - An Extendible Package for Data

       Exploration, Classification and Correlation. Institute of Pharma
ceutical

       and Food Analysis and Technologies, Via Brigata Salerno,

       16147 Genoa, Italy.



   (b) Stefan Aeberhard, email: stefan@coral.cs.jcu.edu.au

   (c) July 1991

3. Past Usage:



   (1)

   S. Aeberhard, D. Coomans and O. de Vel,

   Comparison of Classifiers in High Dimensional Settings,

   Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of

   Mathematics and Statistics, James Cook University of North Queenslan
d.

   (Also submitted to Technometrics).



   The data was used with many others for comparing various

   classifiers. The classes are separable, though only RDA

   has achieved 100% correct classification.

   (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))

   (All results using the leave-one-out technique)
```

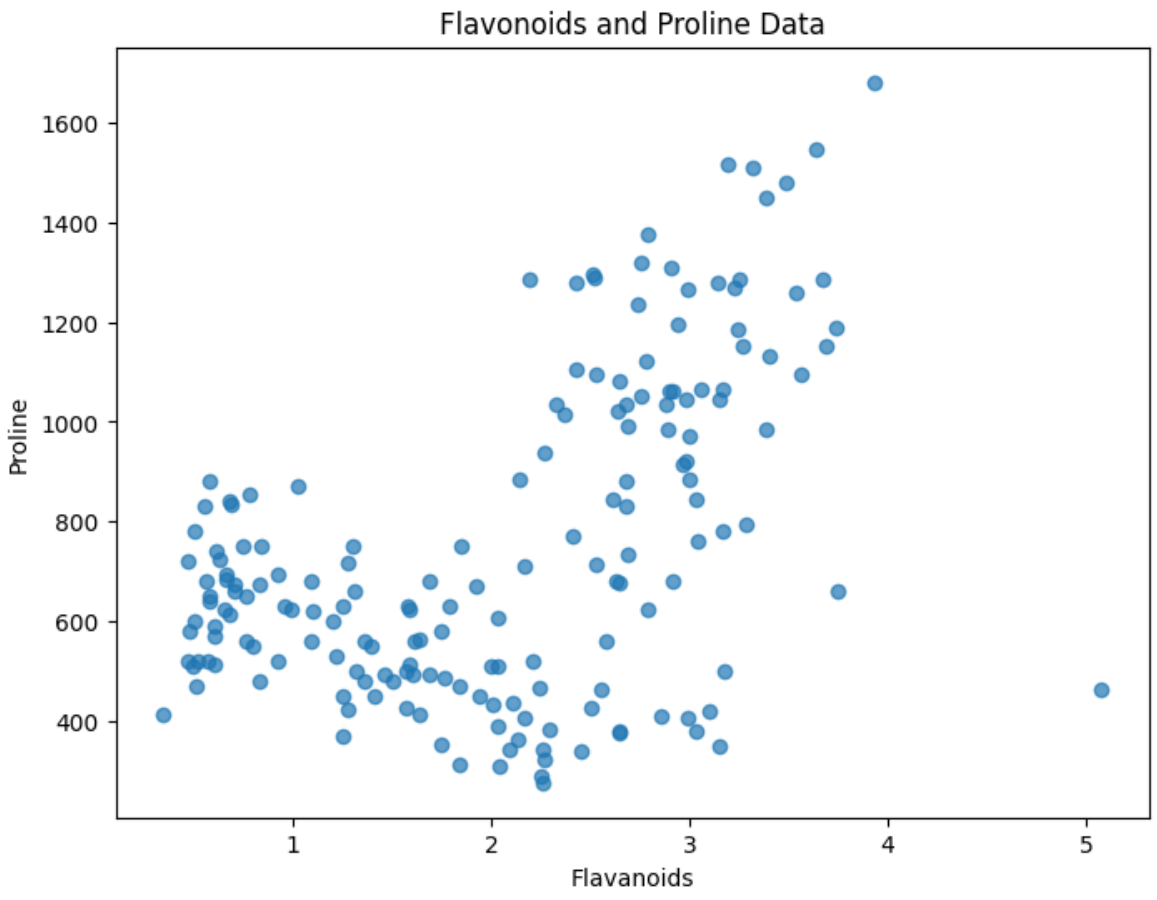
```
In a classification context, this is a well posed problem

with "well behaved" class structures. A good data set

for first testing of a new classifier, but not very

challenging.
```

In [45]:
```
cols = [
    "Class",
    "Alcohol",
    "Malic_acid",
    "Ash",
    "Alcalinity_of_ash",
    "Magnesium",
    "Total_phenols",
    "Flavanoids",
    "Nonflavanoid_phenols",
    "Proanthocyanins",
    "Color_intensity",
    "Hue",
    "OD280/OD315",
    "Proline"
]

df = pd.read_csv("HW1_data/wine/wine.data", names=cols)
print(df.head())
```

```
   Class  Alcohol  Malic_acid   Ash  Alcalinity_of_ash  Magnesium  \
0      1    14.23        1.71  2.43               15.6        127
1      1    13.20        1.78  2.14               11.2        100
2      1    13.16        2.36  2.67               18.6        101
3      1    14.37        1.95  2.50               16.8        113
4      1    13.24        2.59  2.87               21.0        118

   Total_phenols  Flavanoids  Nonflavanoid_phenols  Proanthocyanins  \
0           2.80        3.06                  0.28             2.29
1           2.65        2.76                  0.26             1.28
2           2.80        3.24                  0.30             2.81
3           3.85        3.49                  0.24             2.18
4           2.80        2.69                  0.39             1.82

   Color_intensity   Hue  OD280/OD315  Proline
0             5.64  1.04         3.92     1065
1             4.38  1.05         3.40     1050
2             5.68  1.03         3.17     1185
3             7.80  0.86         3.45     1480
4             4.32  1.04         2.93      735
```

(a) Then, generate a plot between the Flavanoids and the Proline of the observations. Label the axes (Flavanoids should be x-axis and Proline should be y-axis). Call this plot "Flavanoids and Proline Data".

```
In [46]: import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(df["Flavanoids"], df["Proline"], alpha=0.7)
plt.xlabel("Flavanoids")
plt.ylabel("Proline")
plt.title("Flavonoids and Proline Data")
plt.show()
```



The scatter plot of Flavanoids versus Proline shows a general positive correlation: as the Flavanoids value increases, the Proline value also tends to increase. This suggests that wines with higher Flavanoids content also tend to have higher Proline levels.

(b) Compute the mean of the attributes Flavanoids and Proline. Define a data point called P such that P = (mean(Flavanoids), mean(Proline)).

```
In [47]: mean_flav = df["Flavanoids"].mean()
mean_prol = df["Proline"].mean()
```

```
P = (mean_flav, mean_prol)
print("Mean Flavanoids:", mean_flav)
print("Mean Proline:", mean_prol)
print("Point P:", P)
```

```
Mean Flavanoids: 2.0292696629213487
Mean Proline: 746.8932584269663
Point P: (np.float64(2.0292696629213487), np.float64(746.893258426966
3))
```

(c) Compute the distance between P and the 178 data points using the following distance measures: 1) Euclidean distance, 2) Manhattan block metric, 3) Minkowski metric (for power=7), 4) Chebyshev distance, and 5) Cosine distance. List the closest 6 points for each distance.

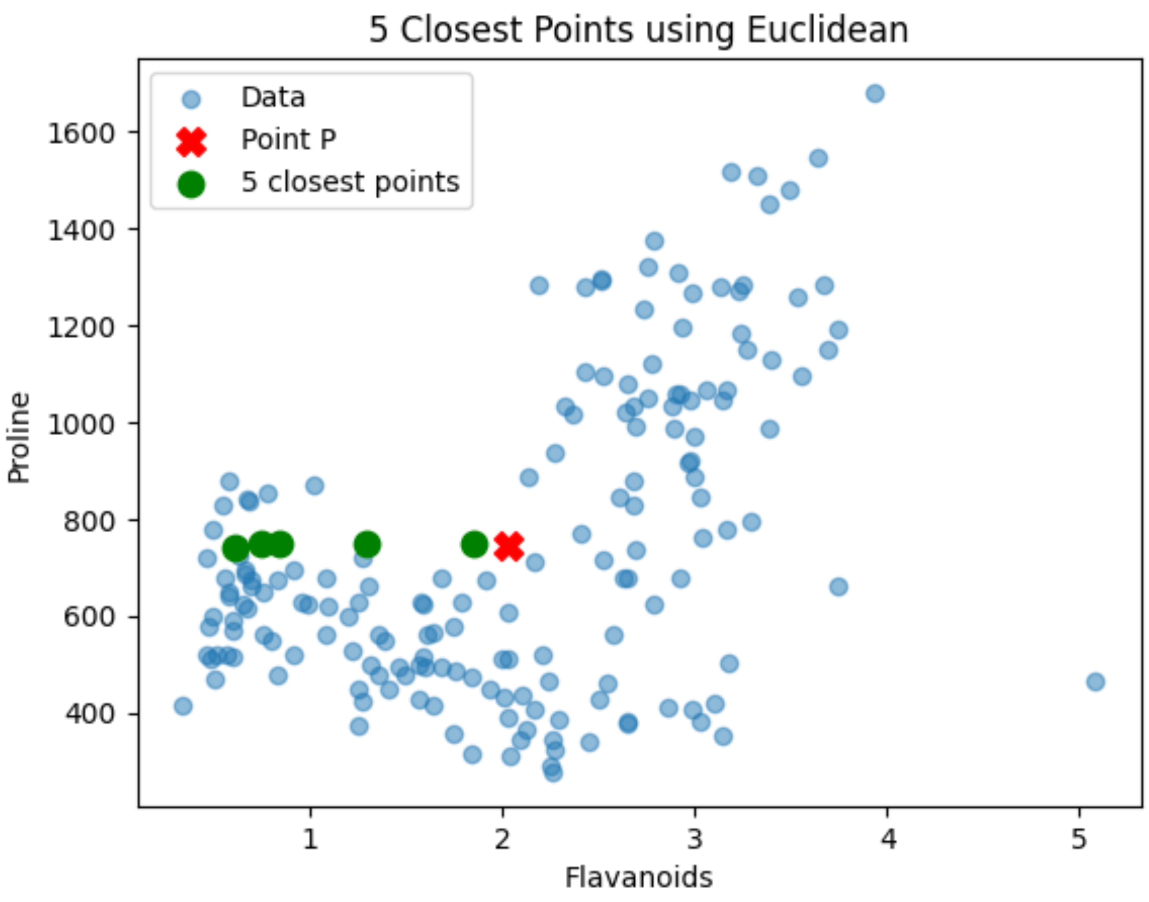
In [48]:
```
import numpy as np
from scipy.spatial import distance

X = df[["Flavanoids", "Proline"]].values

P = np.array([df["Flavanoids"].mean(), df["Proline"].mean()])

distances = {
    "Euclidean": [distance.euclidean(P, x) for x in X],
    "Manhattan": [distance.cityblock(P, x) for x in X],
    "Minkowski (p=7)": [distance.minkowski(P, x, 7) for x in X],
    "Chebyshev": [distance.chebyshev(P, x) for x in X],
    "Cosine": [distance.cosine(P, x) for x in X]
}

closest_points = {}
for metric, dists in distances.items():
    idx = np.argsort(dists)[:6]
    closest_points[metric] = idx

for metric, idx in closest_points.items():
    print(f"{metric} - Closest 6 points: {idx}")
```

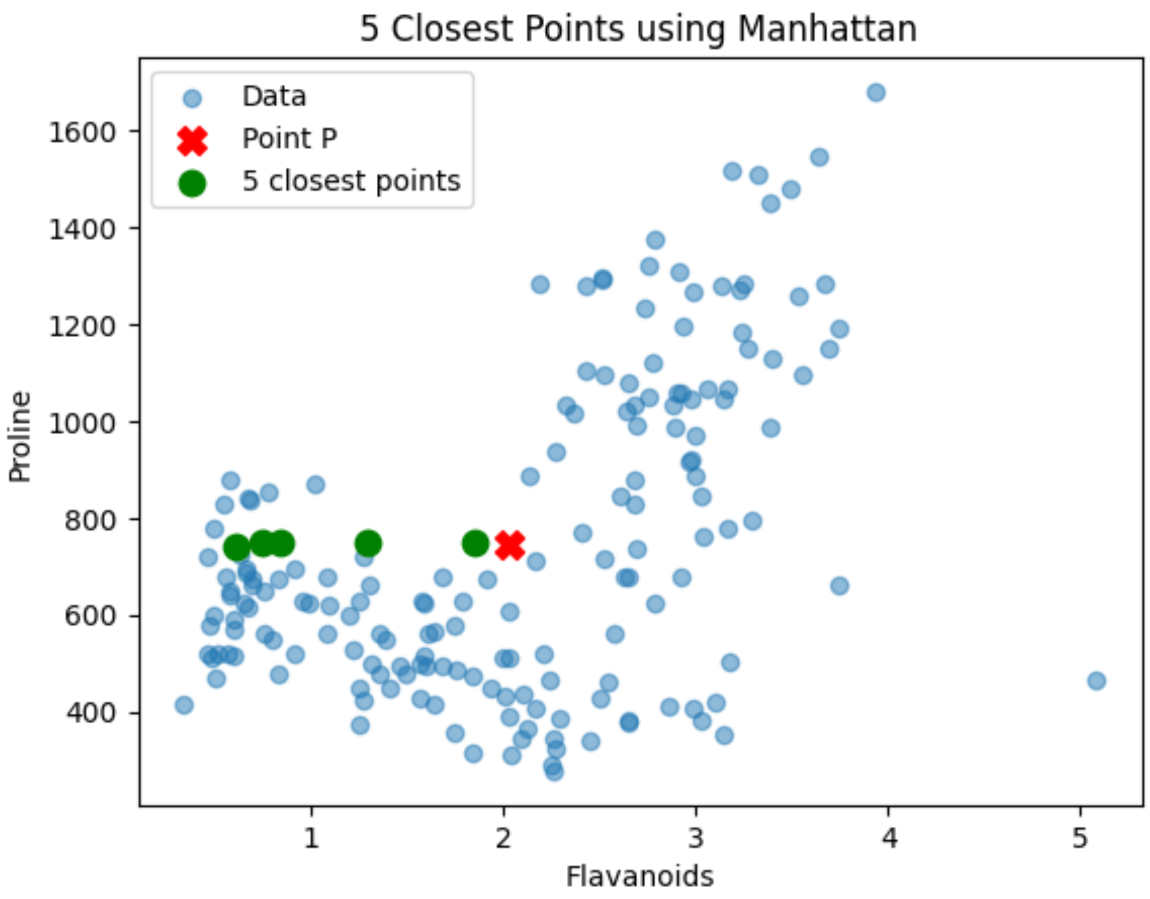
```
Euclidean - Closest 6 points: [ 78  68 168 174 173   4]
Manhattan - Closest 6 points: [ 78  68 168 174 173   4]
Minkowski (p=7) - Closest 6 points: [ 78  68 168 174 173   4]
Chebyshev - Closest 6 points: [ 78 174  68 168 173   4]
Cosine - Closest 6 points: [32  2 54 48 91 22]
```

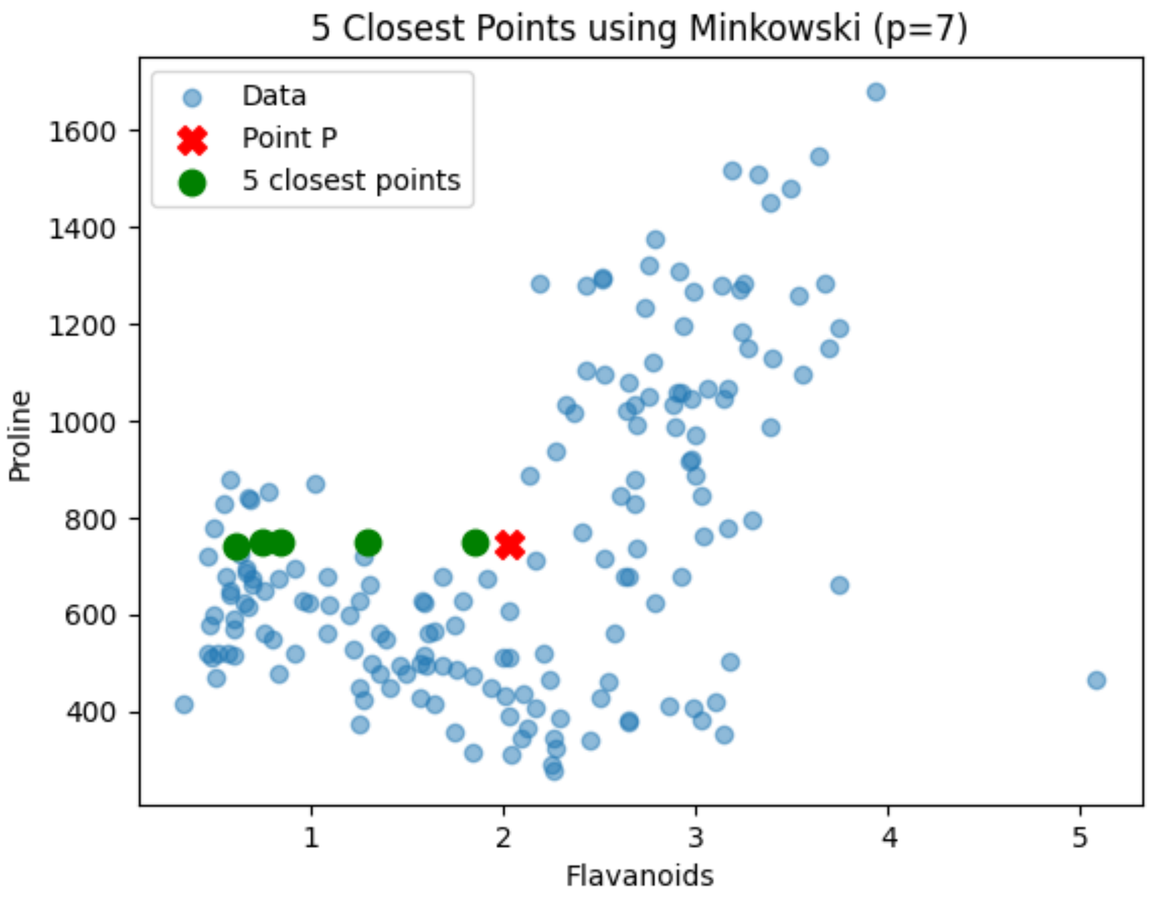
(d) For each distance measure, identify the 5 points from the dataset that are the closest to the point P from (b).

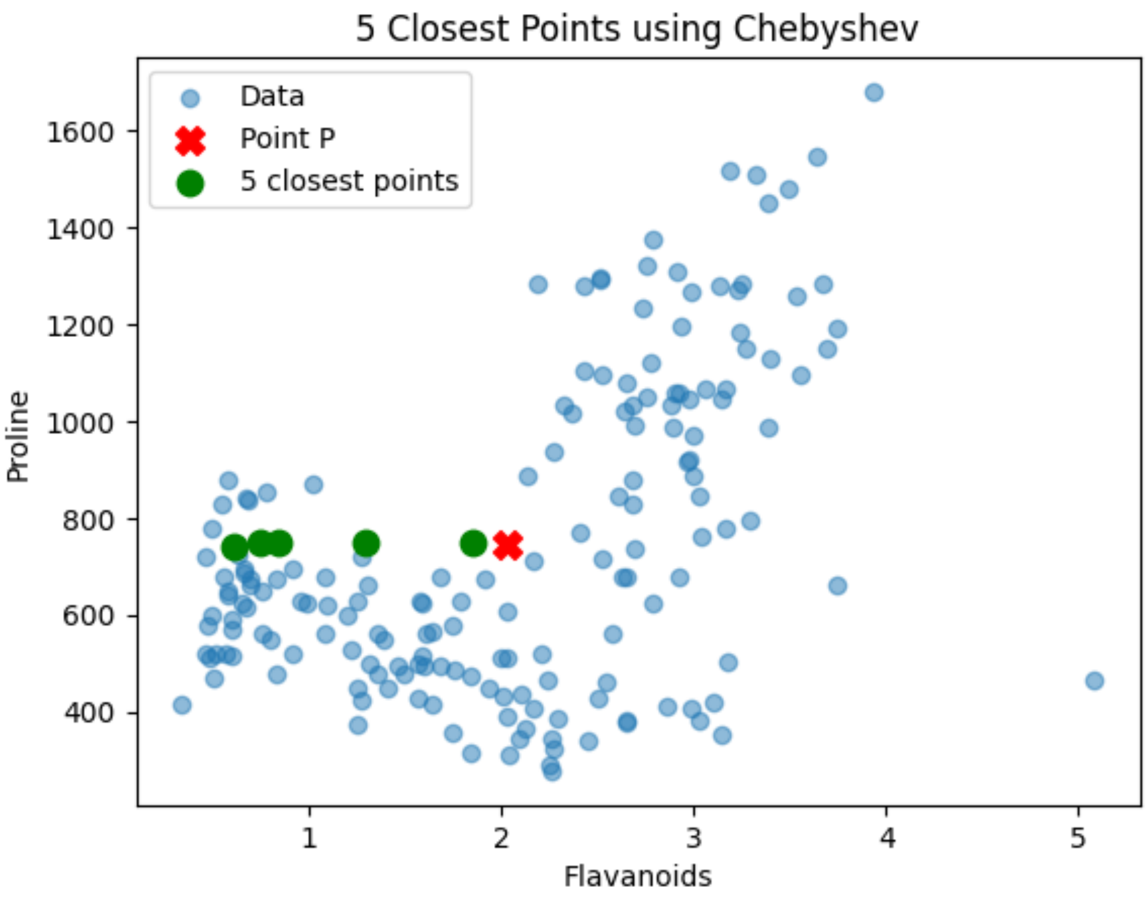
In [49]:
```
closest_points_5 = {}
for metric, dists in distances.items():
    idx = np.argsort(dists)[:5]
    closest_points_5[metric] = idx
```

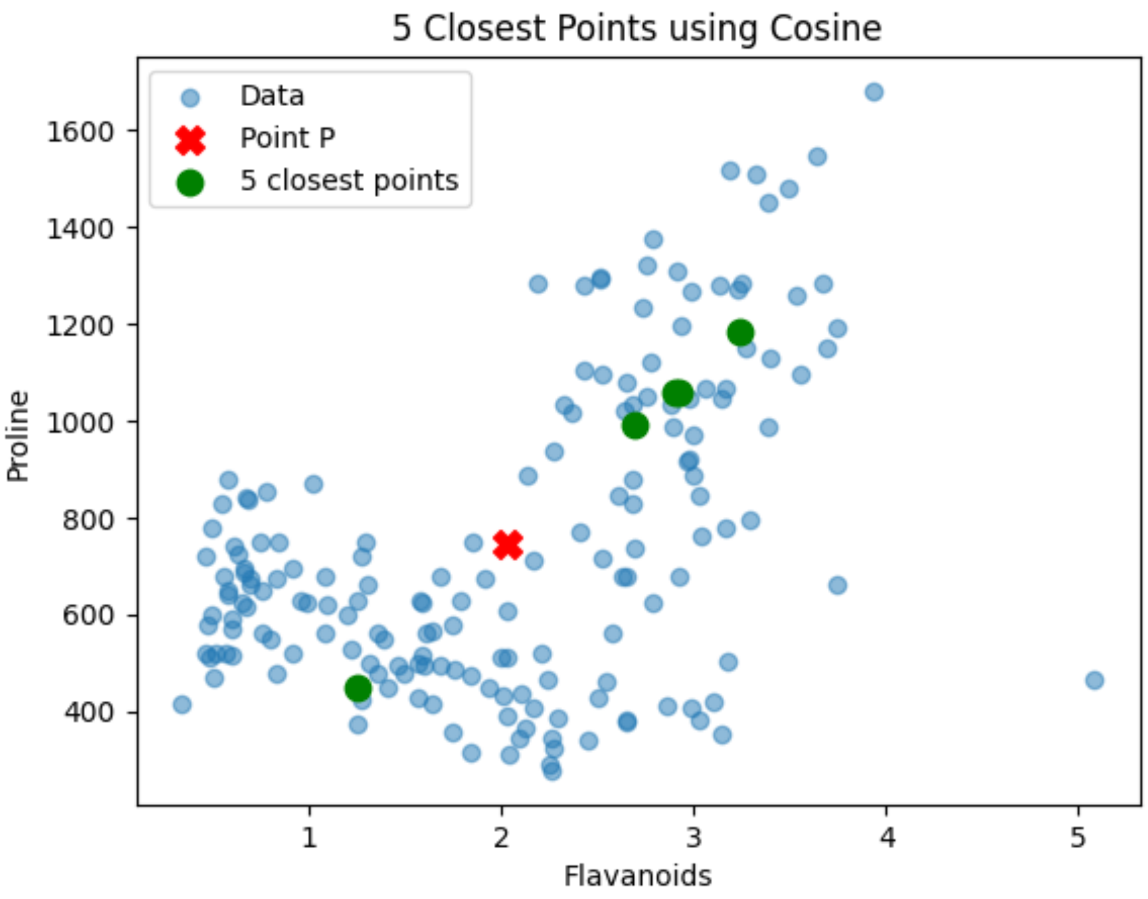
In [50]:
```
for metric, idx in closest_points_5.items():
    plt.figure()
    plt.scatter(df["Flavanoids"], df["Proline"], alpha=0.5, label="Dat
    plt.scatter(P[0], P[1], color="red", marker="X", s=100, label="Poi
    plt.scatter(df.iloc[idx]["Flavanoids"], df.iloc[idx]["Proline"],
                color="green", marker="o", s=80, label="5 closest poin
    plt.xlabel("Flavanoids")
    plt.ylabel("Proline")
    plt.title(f"5 Closest Points using {metric}")
    plt.legend()
    plt.show()
```




5 Closest Points using Manhattan
Data
Point P
5 closest points
Proline
Flavanoids
1600
1400
1200
1000
800
600
400
1
2
3
4
5


5 Closest Points using Minkowski (p=7)
Data
Point P
5 closest points
Proline
Flavanoids
1600
1400
1200
1000
800
600
400
1
2
3
4
5


5 Closest Points using Chebyshev
Data
Point P
5 closest points
Proline
Flavanoids
400
600
800
1000
1200
1400
1600
1
2
3
4
5


5 Closest Points using Cosine
Data
Point P
5 closest points
Proline
Flavanoids
400
600
800
1000
1200
1400
1600
1
2
3
4
5

The four distance measures (Euclidean, Manhattan, Minkowski, and Chebyshev) selected almost the same nearest points, whereas the Cosine distance produced different results. This is because Cosine reflects directional similarity (ratios) rather than the magnitude of values.

7. (a) The scatter plot of Flavanoids vs. Proline shows a general positive correlation.
Flavanoids value increase, the Proline value also tends to increase. This suggests that wines with higher Flavanoids content also tend to have higher Proline levels.

(b) Mean Flavanoids : 2.029

Mean Proline : 746.89

Poin P = (2.03 , 746.89)

(c) in code

(d) i) in code

ii) The 4 distance measures (Euclidean, Manhattan, Minkowski, Chebyshev) selected almost the same nearest points,
whereas the Cosine distance produced different results.
This is because Cosine reflects directional similarity rather than the magnitude of value.