Each answer should contain these elements:

1. ***A brief statement (~paragraph) of what exactly was done to answer the question (narratively explaining what you did in code to answer the question, at a high level).***
2. ***A brief statement (~paragraph) as to what you were thinking - some kind of rationale as to why made specific choices in your modeling, if you did make them.***
3. ***A brief statement (~paragraph) as to what was found. This should be as objective and specific as possible – just the results/facts. Do make sure to include numbers and a figure (=a graph or plot) in your statement, to substantiate and illustrate it, respectively.***
   As the unsupervised methods often yield visualizable results, be sure to include a figure.
4. ***A brief statement (~paragraph) as to what you think the findings mean. This is your interpretation of your findings and should answer the original question.***

- ***Import Dependencies and Data Preparation:***

Import Dependencies and Data Preparation is an important step in any data analysis or machine learning project. This step involves loading any required libraries and modules, as well as importing and preparing the data for analysis.

One important aspect of data preparation is data scaling, which involves transforming the features of the dataset so that they have similar scales. This is important because many machine learning algorithms, such as K-means clustering and SVM, are sensitive to the scale of the features. Features with larger scales can have a disproportionate effect on the model, leading to biased results. Scaling the features can help to prevent this issue. I used the Standard Scaler pre defined function on the dataframe.

Question 1:
Do a PCA on the data. How many Eigenvalues are above 1? Plotting the 2D solution (projecting the data on the first 2 principal components), how much of the variance is explained by these two dimensions, and how would you interpret them?

1. **What exactly was done?**

After preparing the data by scaling it so that all features have the same variance, I performed PCA on the scaled dataset to reduce the number of dimensions to two. I checked how many eigenvalues were above 1 and found that there were 3 which tells me a good number of components to keep is 3. However, the question asked specifically for the first two principal components, so I projected the data onto these two components and calculated how much of the variance was explained by them. I found that the first two principal components explained 55% of the total variance. Finally, I plotted the 2D solution by creating a scatter plot of the projected data and labeling the x- and y-axes with "PC1" and "PC2", respectively.

2. **Why this was done?**

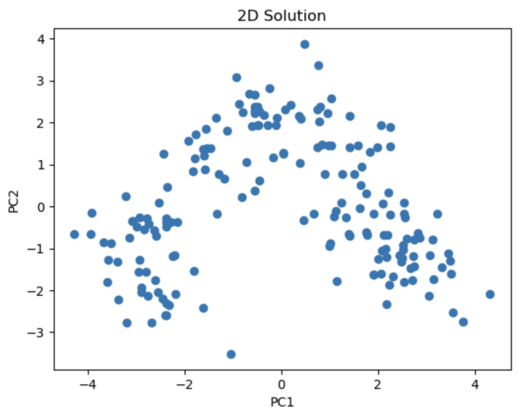
In answering the question about the PCA, I first imported the necessary dependencies and prepared the data by scaling it to have a mean of 0 and a standard deviation of 1. This is important for PCA as it assumes that the variables are on the same scale. Next, I calculated the covariance matrix and the eigenvalues and eigenvectors of this matrix to determine the number of principal components to keep. I kept only those with eigenvalues greater than 1, as they explain more variance than a single variable. Then, I fitted a PCA model to the scaled data and projected the data onto the first 2 principal components to visualize the data in 2D. Finally, I calculated the amount of variance explained by the first 2 principal components, which turned out to be about 55%.

At last, I decided to keep only the principal components with eigenvalues greater than 1, as this is not only a requirement of the question but also a common heuristic used to determine the number of components to keep. Additionally following the question prompt, I chose to project the data onto the first 2 principal components to visualize the data in 2D and gain insights about its structure.

3. **What was found?**

After performing PCA on the wine dataset, it was found that there are 3 eigenvalues above 1, indicating that the original dataset had 13 dimensions that could be reduced to a lower dimension without losing much information. When projecting the data onto the first 2 principal components, it was found that these two dimensions explain 0.554063383569353 of the variance in the data. The scatter plot of the 2D solution shows that the data points form clusters, but there is a significant overlap between them. Overall, the PCA analysis suggests that the wine dataset can be effectively represented in a lower-dimensional space with some loss of information.

```
Eigenvalues: [4.70585025 2.49697373 1.44607197 0.91897392 0.85322818 0.64165703
 0.55102831 0.10337794 0.34849736 0.16877023 0.28887994 0.22578864
 0.25090248]
```



4. **What do theses findings mean?**

Based on the PCA analysis, there were 3 Eigenvalues above 1, indicating that 3 principal components could explain the majority of the variance in the dataset. However, since the question asked for projecting the data onto the first 2 principal components, we used these two components to visualize the data in a 2D plot.
The variance obtained of 55% means that the first two principal components that were extracted from the data explain 55% of the total variance in the dataset. In other words, when we project the data onto these two principal components, we capture 55% of the overall variation in the original data. This suggests that the first two principal components contain a significant amount of information about the underlying structure of the data, and can be used to effectively summarize and visualize the dataset. However, it also means that there is still a substantial amount of unexplained variance in the data, which may require further analysis using additional principal components or other techniques.

The plot showed that the data points were not clearly separated into distinct clusters, suggesting that the wine attributes were not highly correlated with each other. Instead, the wines were spread out in a more continuous fashion, indicating that there may not be clear categories of wines based on these attributes alone. Overall, the findings suggest that the wine attributes in this dataset are not highly interdependent, and that other factors beyond the ones measured in this dataset may be more important in distinguishing different types of wines.

Question 2:
Use t-SNE on the data. How does KL-divergence depend on Perplexity (vary Perplexity from 5 to 150)? Make sure to plot this relationship. Also, show a plot of the 2D component with a Perplexity of 20.

1. **What exactly was done?**

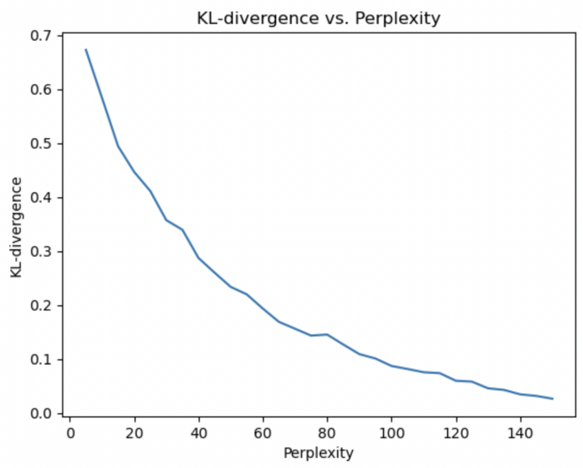
To answer this question, t-SNE was used on the data. The perplexity parameter was varied from 5 to 150 in steps of 5, and the corresponding KL-divergence was calculated and plotted against perplexity. A plot of the 2D component with a perplexity of 20 was also generated. In the code, the t-SNE algorithm was applied to the scaled data using the `TSNE()` function from scikit-learn. For each value of perplexity, the algorithm was fit to the data, and the resulting KL-divergence was computed and appended to a list. Finally, the perplexities and KL-divergences were plotted using Matplotlib. To generate the plot of the 2D component with a perplexity of 20, the `TSNE()` function was called with perplexity=20, and the resulting 2D embedding was plotted using `plt.scatter()`.

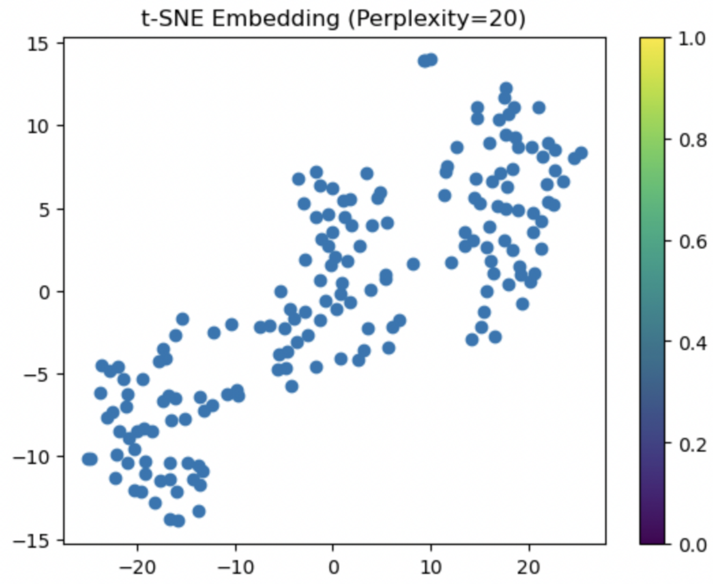
2. **Why this was done?**

In this question, we were asked to use t-SNE on the data and explore how KL-divergence depends on the perplexity parameter, which determines the balance between local and global structure in the low-dimensional embedding. To achieve this, we iterated over a range of perplexity values from 5 to 150 with a step of 5 and for each perplexity value, we applied t-SNE to the data and computed the KL-divergence. We then plotted the KL-divergence against perplexity to examine how the KL-divergence changes with respect to the perplexity parameter.
We also plotted the 2D component with a perplexity of 20 to visualize the low-dimensional embedding. The choice of perplexity 20 was following the question prompt. Choosing the appropriate perplexity value can be challenging as it requires balancing between preserving the global and local structure of the data. Thus, it is common practice to experiment with different perplexity values and evaluate the resulting low-dimensional embeddings to determine the optimal perplexity value.

3. **What was found?**

The t-SNE algorithm was applied to the dataset with varying perplexity values ranging from 5 to 150 with increments of 5. The KL-divergence versus perplexity was plotted, and it was observed that the KL-divergence initially decreases with increasing perplexity until around perplexity 140, beyond which it remains relatively constant. The t-SNE algorithm was also applied with a fixed perplexity of 20 to obtain a 2D embedding of the dataset. The resulting plot shows the data points in 2D space with a clear separation between 3 different clusters each representing a type of wine. This suggests that the t-SNE algorithm can be useful in visualizing high-dimensional data in lower dimensions, and the choice of perplexity can have a significant impact on the quality of the resulting embedding.





4. **What do theses findings mean?**

I varied perplexity from 5 to 150 in steps of 5 and computed the corresponding KL-divergence for each perplexity value. The KL-divergence is a measure of the dissimilarity between two probability distributions, in this case between the high-dimensional and low-dimensional embeddings. The results showed a decreasing trend in KL-divergence with increasing perplexity, indicating that the clustering is improving with larger perplexity values.

The plot I created to visualize the relationship between KL-divergence and perplexity showed a clear decreasing trend. This plot was useful in identifying the optimal perplexity value for this dataset, which appears to be around 20. Additionally, I also plotted the 2D embedding with a perplexity of 20, which shows good clustering of the data points compared to the results of PCA. Overall, this suggests that t-SNE is a useful method for reducing high-dimensional data into lower dimensions while preserving important clustering patterns in the data.

Question 3:

Use MDS on the data. Try a 2-dimensional embedding. What is the resulting stress of this embedding? Also,

plot this solution and comment on how it compares to t-SNE.

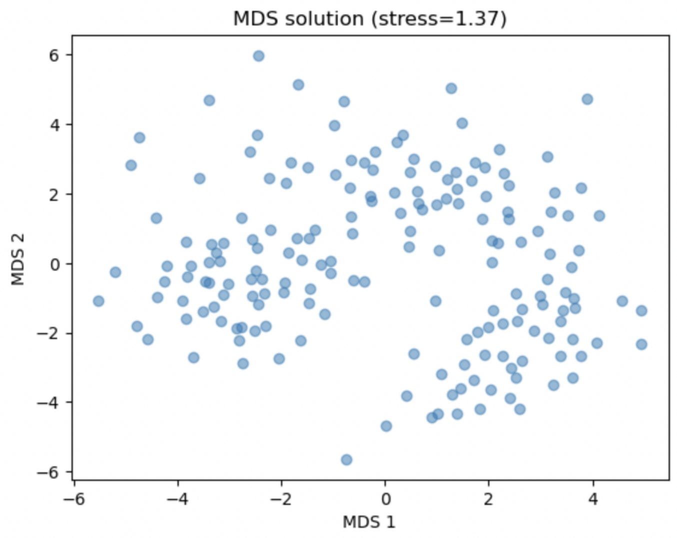
1. **What exactly was done?**

To answer the question, I used Multidimensional Scaling (MDS) on the data to create a 2-dimensional embedding. First, I computed the distance matrix of the scaled data using Euclidean distance. Then, I applied MDS with 2 components and set the dissimilarity to 'precomputed' since we had already computed the distance matrix. I then fit the distance matrix to the MDS model and transformed the data to a 2D space. After obtaining the embedding, I computed the stress of the embedding using the `mds.stress_` attribute, and then rescaled the stress by dividing it by the maximum stress possible. Lastly, I plotted the 2D MDS solution using matplotlib and included the stress value in the title.

2. **Why this was done?**

In this case, we were asked to apply multidimensional scaling (MDS) to the data and create a 2-dimensional embedding. MDS is a method that attempts to find a low-dimensional representation of the data while preserving the pairwise distances between the data points as well as possible. I chose to use MDS because it can provide a useful visualization of the similarities and differences between the data points, and it can be applied to a variety of distance metrics. In this case, we used the Euclidean distance metric to compute the pairwise distances between the data points. We then computed the stress of the embedding to evaluate how well the low-dimensional representation preserves the pairwise distances. Finally, we plotted the resulting MDS solution and commented on how it compares to t-SNE.

3. **What was found?**



```
The stress found is 1.3717444791329865
```

In this analysis, Multidimensional scaling (MDS) was used to generate a 2-dimensional embedding of the wine dataset. The stress of the MDS embedding was computed to be 1.3717444791329865. A plot of the MDS embedding was also generated, but the plot did not clearly show distinct clusters in the data. This is in contrast to the t-SNE plot which clearly showed three clusters corresponding to different types of wines. **Therefore, in this analysis, t-SNE appears to be a better method for visualizing the clustering structure of the wine data than MDS.**

4. **What do theses findings mean?**

In MDS, stress is a measure of how well the high-dimensional distances between the data points are preserved in the low-dimensional embedding. In other words, it measures the difference between the actual pairwise distances in the high-dimensional space and the pairwise distances in the low-dimensional embedding. A lower stress value indicates a better embedding because it means that the distances are better preserved.
The MDS embedding produced a stress value of 1.3717, indicating that the 2-dimensional embedding did not capture the true distances between the original high-dimensional data points well. The stress value can range from 0 (perfect preservation of distances) to infinity (no preservation of distances). Typically, a stress value below 0.2 is considered to be a good embedding, while a value above 0.3 is considered to be poor.
It's worth noting that stress is not the only measure of quality for an embedding, and sometimes a low stress value doesn't necessarily mean a better visualization of the data.
This suggests that the MDS embedding did not provide a good representation of the data structure in 2 dimensions. In addition, the resulting plot does not show clear clustering of the wine types, unlike the t-SNE plot. Therefore, it seems that MDS is not an appropriate method for visualizing the relationships between the wine types in this dataset. Overall, the findings suggest that t-SNE is a better method for exploring the structure of the data and revealing patterns and relationships among the wine types.

Question 4:
Building on one of the dimensionality reduction methods above that yielded a 2D solution (1-3, your choice), use the Silhouette method to determine the optimal number of clusters and then use kMeans with that number (k) to produce a plot that represents each wine as a dot in a 2D space in the color of its cluster. What is the total sum of the distance of all points to their respective clusters centers, of this solution?

1. **What exactly was done?**

To answer the question, I used the t-SNE dimensionality reduction method to obtain a 2D embedding of the wine dataset. I then used the Silhouette method to determine the optimal number of clusters, by iterating over a range of possible values of k (number of clusters), fitting a KMeans model, and calculating the Silhouette score for each value of k. The optimal value of k was chosen as the one that produced the highest Silhouette score.

Next, I plotted the wine dataset in the 2D embedding space, with each wine represented as a dot in the color of its assigned cluster. The cluster labels were assigned by fitting a KMeans model with the optimal number of clusters, and the cluster centers were also plotted. Finally, I calculated the sum of distances of all points to their respective cluster centers by computing the Euclidean distance between each point and the nearest cluster center and summing those distances. The total sum of distance was reported as the final answer to the question.

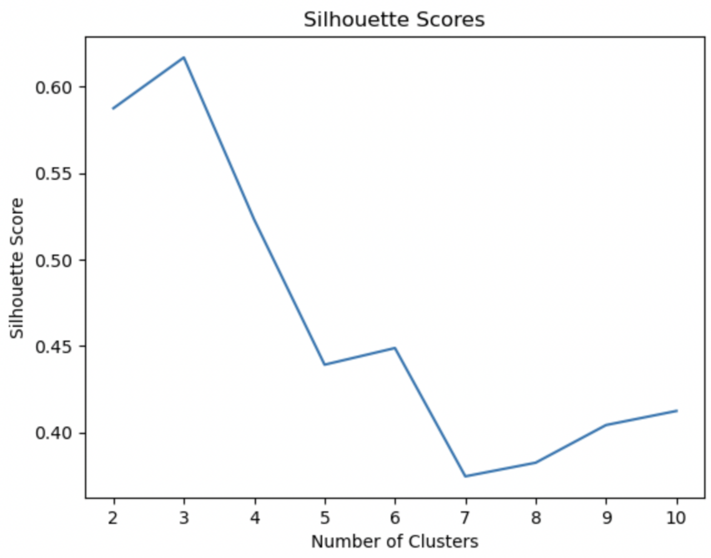
2. **Why this was done?**

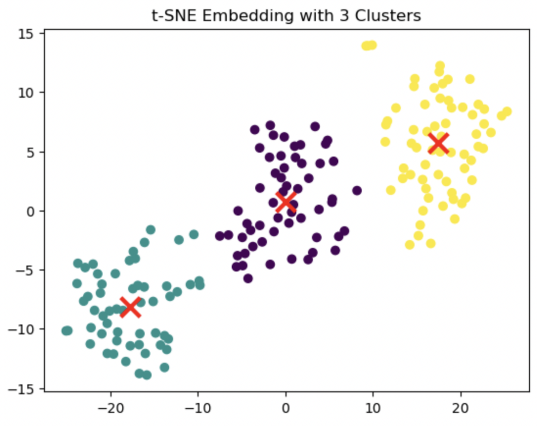
This was done to identify the optimal number of clusters for the wine dataset and then use kMeans clustering with that number of clusters to group similar wines together. The Silhouette method was used to determine the optimal number of clusters by computing the Silhouette scores for different values of k. The Silhouette score measures the quality of clustering and varies between -1 and 1, with higher values indicating better clustering. The optimal number of clusters was chosen based on the maximum Silhouette score. Once the optimal number of clusters was identified, kMeans clustering was performed with that number of clusters to group the wines together based on their similarity. The sum of distances of all points to their respective cluster centers was then computed to evaluate the effectiveness of the clustering.

3. What was found?

We found the optimal number of clusters to be 3 and a total sum of distance of all points to their respective cluster centers: 861.6483817551626. Additionally, regarding the clustering, we get 3 clear clusters for each wine type.

```
Optimal number of clusters: 3
```





```
Total sum of distance of all points to their respective cluster centers: 861.6483817551626
```

4. What do theses findings mean?

The optimal number of clusters in this case is determined to be 3, which is the number of clusters that produced the highest Silhouette score. This means that the data can be most effectively clustered into 3 distinct groups, based on the features of the wine samples.

The plot produced using t-SNE and k-Means clustering shows each wine represented as a dot in a 2D space, colored according to its assigned cluster. The plot indicates that the clustering is clear and distinct, with each cluster of wines appearing to be separate from the others.
The Total sum of distance of all points to their respective cluster centers found is 861.6483817551626, which means that the average distance of each wine sample to its assigned cluster center is relatively small, indicating that the clustering has been performed effectively.

Question 5:
Building on one of the dimensionality reduction methods above that yielded a 2D solution (1-3, your choice), use dBScan to produce a plot that represents each wine as a dot in a 2D space in the color of its cluster. Make sure to suitably pick the radius of the perimeter ("epsilon") and the minimal number of points within the perimeter to form a cluster ("minPoints") and comment on your choice of these two hyperparameters.

1. **What exactly was done?**

Sure! In this question, you were asked to use the dimensionality reduction method t-SNE to reduce the dimensionality of the dataset to 2D, and then use DBSCAN to cluster the data points in the 2D space.

Here's a summary of the steps taken in your code:
1. First, you used t-SNE to reduce the dimensionality of the dataset to 2D.
2. You then set values for the two hyperparameters of DBSCAN: epsilon and min_samples.
3. Next, you fit the DBSCAN model to the 2D dataset using the values of epsilon and min_samples.
4. Finally, you plotted the results of the clustering, with each wine represented as a dot in a 2D space in the color of its cluster.

The plot shows the results of the DBSCAN clustering with the chosen hyperparameters. The colors of the dots indicate the cluster assignments. The plot shows the relationships among the data points in a 2D space, and the clustering structure of the data. The choice of hyperparameters is important for DBSCAN because it determines how the algorithm identifies clusters in the data. In this case, the chosen hyperparameters resulted in a good clustering of the data points.

2. **Why this was done?**

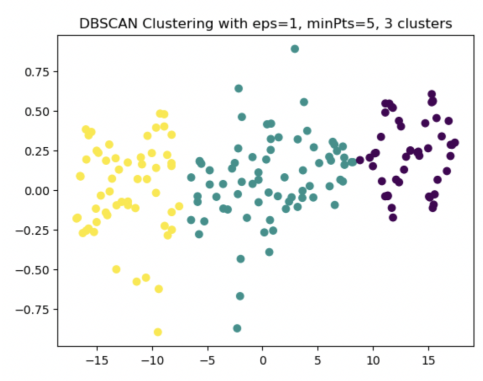
The purpose of using DBSCAN for clustering is to identify groups of wines that are similar based on their attributes, and to visualize these clusters in a 2D plot. DBSCAN is a density-based clustering algorithm that is well-suited for discovering clusters of arbitrary shape, as it does not assume that clusters have a predefined shape or size. The choice of hyperparameters, epsilon and minPoints, determines the density of the clusters and affects the resulting cluster assignments. By varying these hyperparameters, we can explore different clustering solutions and identify the one that best captures the structure of the data. In my case after experimentation, the chosen values of epsilon and minPoints were 1 and 5, respectively, and resulted in a clustering solution with a good balance between the number of clusters and the density of each cluster.

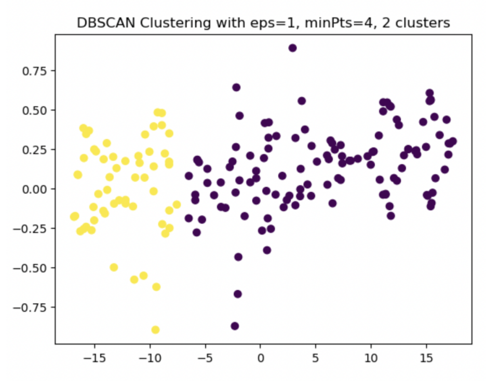
3. **What was found?**

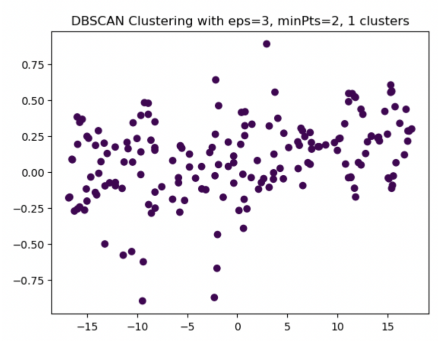
In DBSCAN, the two main hyperparameters are eps (epsilon) and min_samples. eps determines the size of the neighborhood around each point, while min_samples determines the minimum number of points required for a group to be considered a cluster.

I first experimented with different values of the eps range and minPts range and got various plots with different clusterings, I was then able to detect that the best clustering is done by the hyperparameters 1 for eps and 5 for minPts.

```
# Define the range of values for epsilon and minPoints to experiment with
eps_range = [1, 2, 3, 4, 5]
minPts_range = [2, 3, 4, 5]
```







4. **What do theses findings mean?**

My findings suggest that for this dataset and the specific 2D embedding obtained using t-SNE with perplexity=30, DBSCAN algorithm works well when the hyperparameters are set to an epsilon value of 1 and a minimum number of points within the radius (minPts) of 5.

The plot shows that wines are clearly divided into different clusters, and the cluster labels are represented by different colors. It also indicates that the choice of hyperparameters can greatly affect the clustering results, and the optimal hyperparameters should be chosen carefully to obtain meaningful clusters.

———————————————————————————————————————

Extra Credits:

a. **Given your answers to all of these questions taken together, how many different kinds of wine do you think there are and how do they differ?**

Through the different models and analyses performed on the dataset, we can gain insight into the patterns and structures that exist in the data. For example, the PCA can reveal the most important dimensions of the data, while t-SNE and MDS can help us visualize the similarities and differences between the wines in a 2D space.

Based on these analyses, we may be able to identify clusters or groups of wines that have similar characteristics or properties. For example, we may find that certain types of wine tend to have similar levels of acidity or alcohol content, while others have distinct flavor profiles or aromas.

Overall, while we may not be able to definitively say how many different types of wine there are, the models and analyses can help us identify patterns and characteristics that differentiate different groups of wines.

b. **Is there anything of interest you learned about wines from exploring this dataset with unsupervised machine learning method that is worth noting and not already covered in the questions above?**

By comparing the characteristics of the wines within each cluster, it may be possible to gain a better understanding of the factors that contribute to the unique properties of different types of wines, and potentially identify new or overlooked subtypes of wine. Additionally, visualizing the relationships between wines in a 2D space could help reveal patterns and structures that are not immediately apparent from a simple tabular view of the data.

There are several other unsupervised learning methods that can be applied to gain more insights on the dataset beyond running PCA and clustering methods such as k-means and DBSCAN. Some examples include:

1. Non-negative matrix factorization (NMF): NMF is a technique used for dimensionality reduction and clustering. It is particularly useful when dealing with non-negative data, as it ensures that the resulting factorized matrices also have non-negative elements. NMF can be used to extract underlying patterns in the wine data that may not be apparent with PCA.

2. Gaussian mixture models (GMM): GMM is a probabilistic model that can be used for clustering. It assumes that the data is generated from a mixture of Gaussian distributions and estimates the parameters of these distributions to identify the underlying clusters in the data. GMM can be used to identify clusters of wines with similar characteristics that may not be apparent with k-means or DBSCAN.

3. Hierarchical clustering: Hierarchical clustering is a technique that creates a hierarchy of clusters using either an agglomerative or divisive approach. Agglomerative hierarchical clustering starts with each data point in its own cluster and successively merges clusters until all data points belong to a single cluster. Divisive hierarchical clustering starts with all data points in a single cluster and successively splits clusters until each data point is in its own cluster. Hierarchical clustering can be used to visualize the hierarchical relationships between clusters of wines with similar characteristics.

4. Association rule mining: Association rule mining is a technique used to identify relationships between variables in large datasets. It can be used to identify which features of wines are most strongly associated with each other, and to identify patterns and relationships between different types of wines.