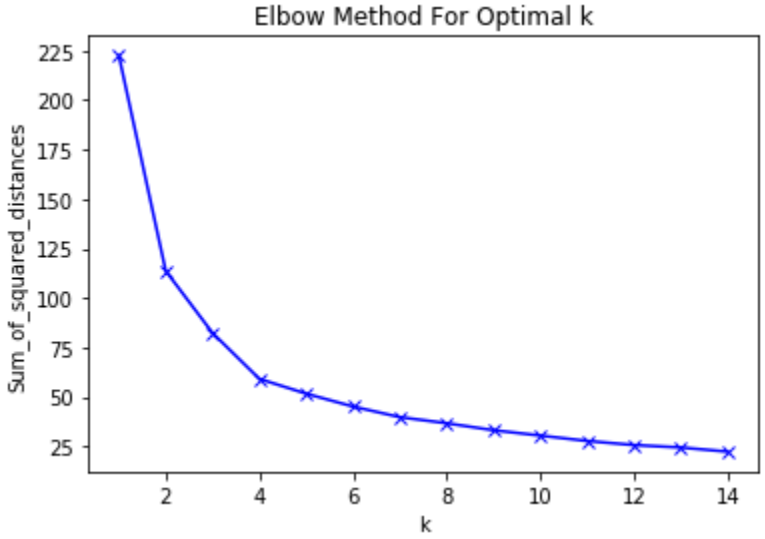
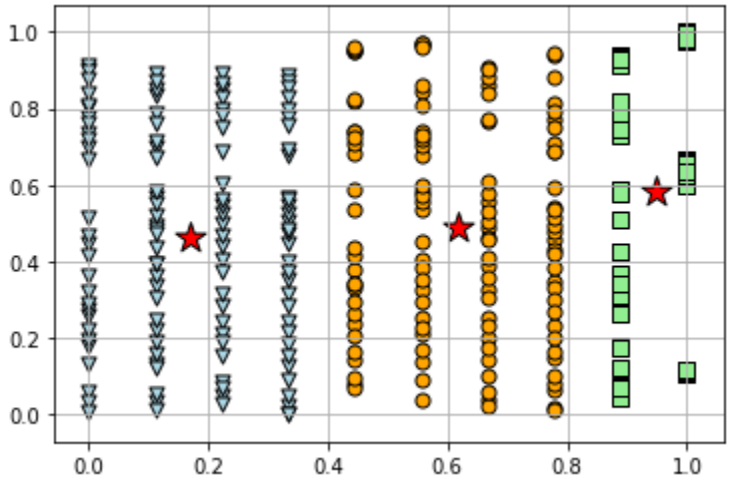
Question 1

For this problem, use all the columns.

• Conduct k-means clustering on the dataset above.

• Determine the optimal number of clusters for k-means and discuss the reasoning behind your choice.

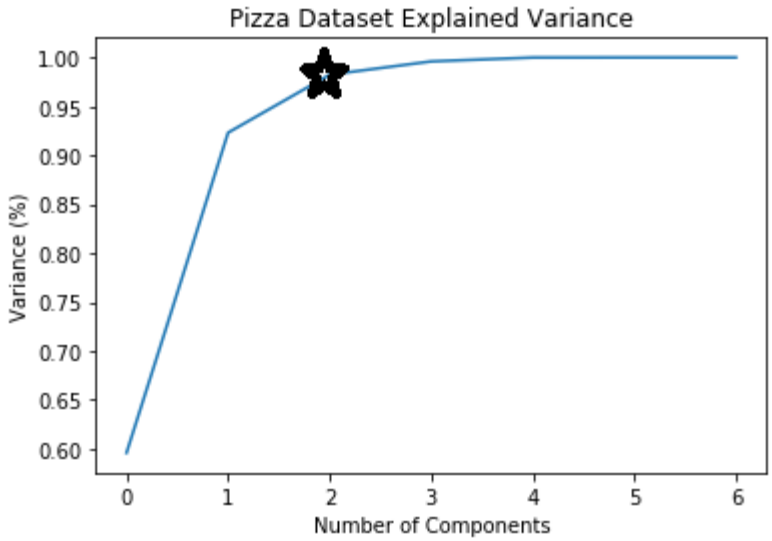




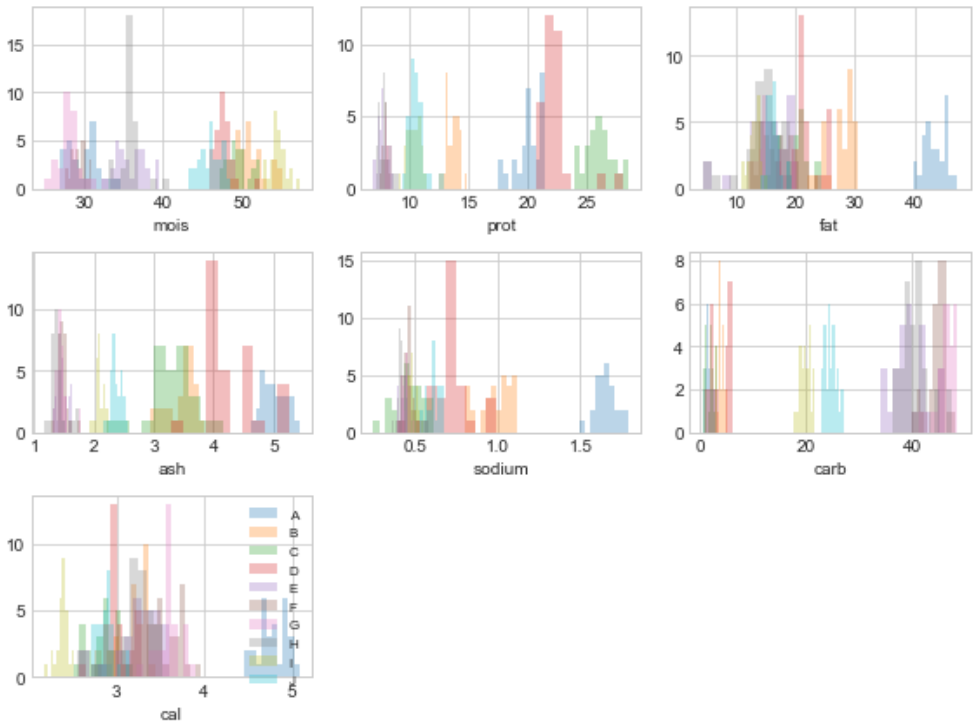
Using elbow method, we can see that one of the optimal values of the K is 3. I consider it according to Marginal Diminishing Returns theory proposed in Economics. Here we can see that adding extra k values will produce less significant Sum_of_Squared_Distances when the break-even point is 3. Please refer to code file for more reference.

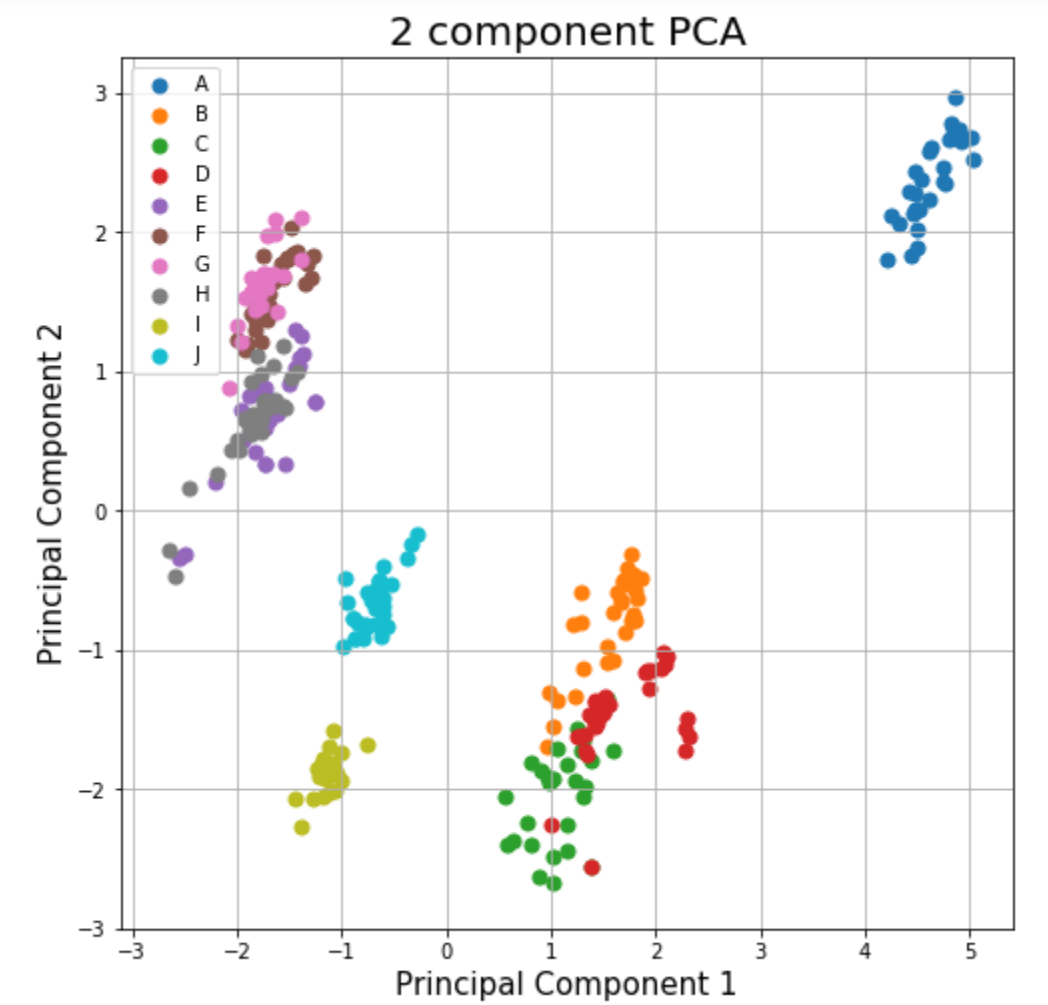
Question 2

What's the optimal number of components for the pizza dataset?



• What does the feature distribution look like?
(Hint: Compare original features to New PCA components)





• What is the explained variance for every new PCA feature?

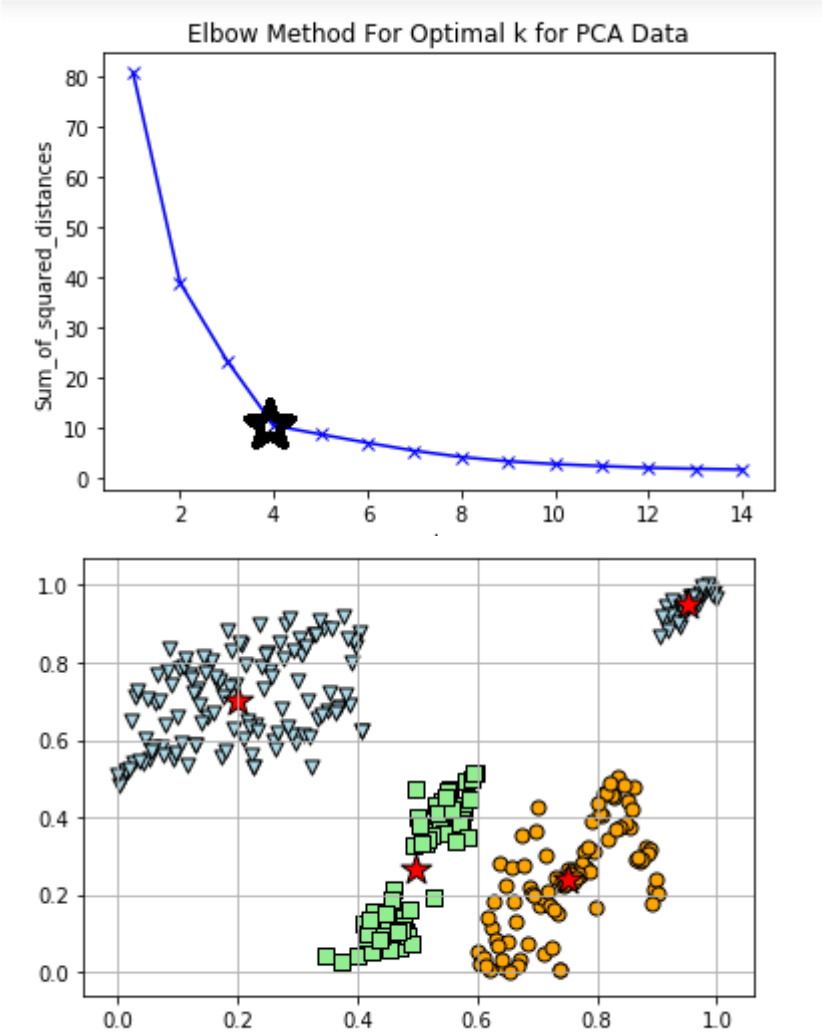
```
In [106]: pca.explained_variance_ratio_
Out[106]: array([0.59596884, 0.3272082 ])

In [88]: print(abs( pca.components_ ))
[[0.06470937 0.3787609  0.44666592 0.47188953 0.43570289 0.42491371
  0.2444873 ]
 [0.62827587 0.26970665 0.23437908 0.11099042 0.20166165 0.32031208
  0.56745756]]
```

The explained variance tells us how much information (variance) can be attributed to each of the principal components. By using the attribute explained_variance_ratio_, we can see that the first principal component

contains 59.59% of the variance and the second principal component contains 32.72% of the variance. Together, the two components contain 92.31% of the information.

Question 3

The third exercise is to perform k-means clustering on the features obtained from PCA. (For this experiment, exclude all original features of the dataset and use only the PCA components)
• Do you see a change in the ideal number of clusters that are to be used? Explain your intuition based on the results observed.



Based on the result of PCA data we can conclude that optimal number of cluster reached 4 with clear division of cluster among 4 clusters which can be visible from the above plot.

Question 4

The key milestones to build robust and implementable recommender system model we need to address three main components. The first is considers how to collect 'known' matrix that is how to collect data for utility matrix. The second is we want to know what our customers want rather than what they do not want. And the last one is how to measure the performance of recommendation method. To build our model we need the data regarding customer past behavior or there should be some kind of metric that can be used to understand customer perceptions like ratings, reviews or the frequency of their particular actions like clicks. Thus gathering data can be Explicit (ask people to rate our books or pay third party vendors like Amazon Mechanical Turk to conduct surveys for us) and Implicit (learn from customer actions such as purchasing patterns, clicks, revisit frequency). Other than that, we can ask short and clear questions in the beginning of sing up process to get initial screening of the client's basic preferences, reading habits and interests. Therefore, there are two popular approaches for recommender systems content based and collaborative. The main idea behind content based approach is to recommend particular book according to our customer's previous purchases or interests. This is the most common use for a recommendation system as it ranks products by how much a user would like them. If a user is browsing or searching for products, we want to show them the products they would like most first in the list. For example, recommend particular book with the same writer genre or popularity. Here we create item profiles where profile is set of features like author, genre. Then feature selection should be used to extract important features which can be done with TF-IDF. The main advantages of content based approach is we do not need to cold start or sparsity of matrix; can be self-explanatory as the recommendation is based upon content-features. However, the main drawbacks would be, overspecialization and finding optimal features is troublesome. Recommendation systems can also be used to find out how similar different products are to each other. If products are very similar to each other, they might appeal to the same users. In collaborative filtering, the recommendation is done based on preferences of similar group. In other words, here we are trying discover similar customers. As we mentioned above the term similar which means we will be using similarity measure to compute distance. Pearson Correlation Coefficient is used for calculation here. The main advantage of this method is that it does not require feature selection, while disadvantages are: can produce popularity bias, cold start and cannot recommend item if that has not been previously bought or rated, especially when new arrival books are on the catalogue. Overall we can articulate that, this recommendation system has no knowledge of the actual book it is recommending. It only knows how other users rated the book.

The input needed for model training plays a key role. Depending on our business goal, model building can be based on such types of data as content, historical data, or user data involving views, clicks, and likes. The data used for training a model to make recommendations can be split into several categories. . User behavior data (historical data): Log on-site activity: clicks, searches, page, and item views, Off-site activities: tracking clicks in emails, in mobile applications, and in their push notifications. Particular item details: Title, Category, Price, Description, Style. Contextual information: Device used, Current location, Referral URL.

One particular technique which has been widely used in recommendation systems is graph-ranking. This technique estimates the importance of a node in a graph, and it has been introduced into recommendation systems to model the interaction between users and items, through a graph. Several graph-based recommendation methods have been introduced, such as the random walk method. One famous random walk model is the PageRank algorithm, which has been used in search engines to rank items.

Similar what we saw in PageRank algorithm, we build a graph using all books we have in our catalogue. So, customers and books are both nodes. If the user likes a book, we can draw two directed edges between them, I mean towards to each other. We find that if we run a Page Rank algorithm on this graph, the book with the highest value will be the most popular book. It is global optimal result. But our goal is to find the book that this particular customer may like. We need to find personalized highest rank. In order to do this, we assign the value of 1 to the user node that we want to find highest rank. All other nodes will have rank value of 0. We can use factor alpha here. It will represent the fraction of value that stays at its own node. We assume that the user we focus on will have k neighboring nodes. First we only distribute (1-a)/k to nodes connected to the user node. We keep value alpha at the user node. Next we distribute 1-a fraction of values from every nodes and keep alpha fraction of values. We repeat all that step unit convergence. Then we can pick up a book node with the highest value. That book is what we should recommend to the user we focus on. Therefore, it is useful to use Pagerank for recommendation systems