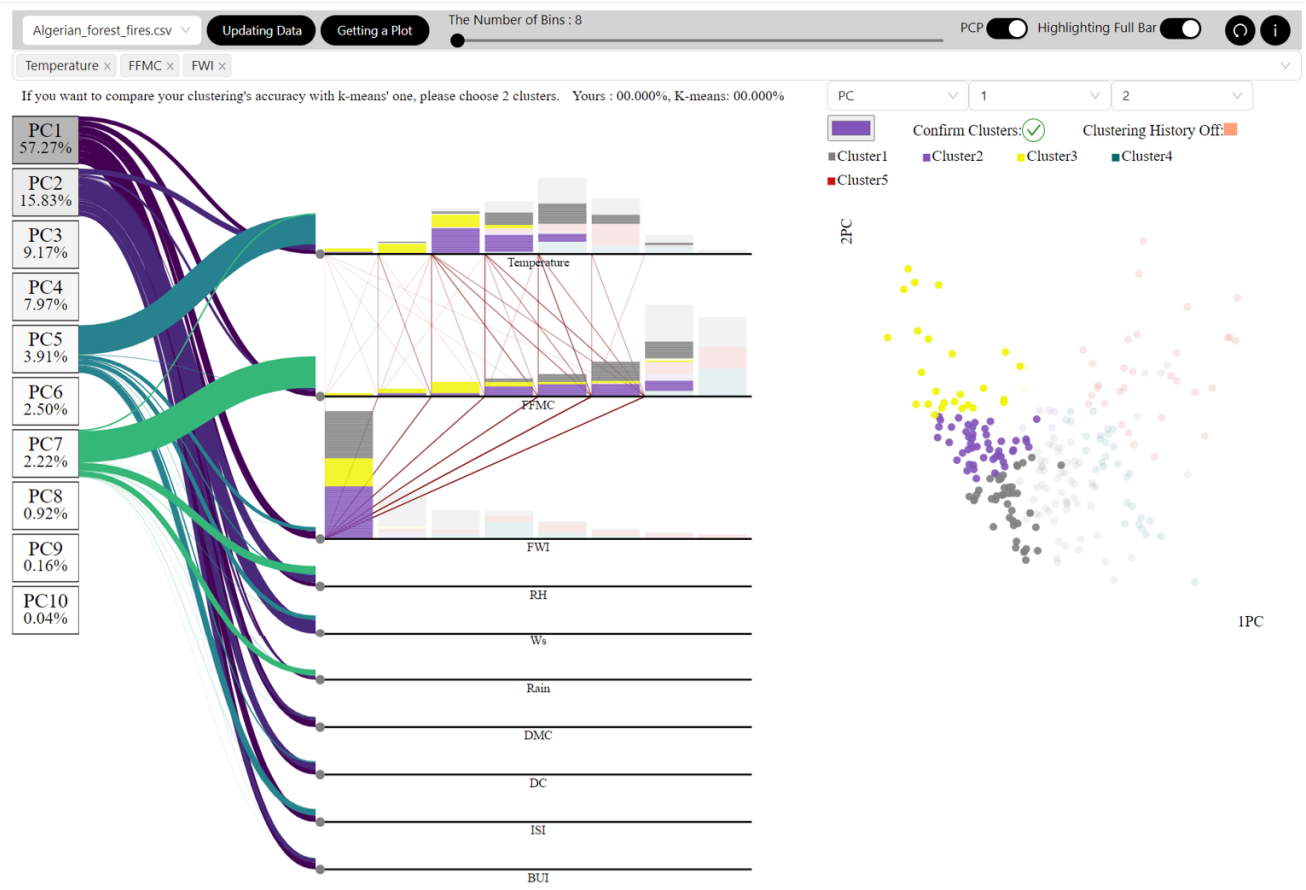
Multi-dimensional Data Clustering based on Parallel Histogram Plot



Master Thesis
08.Aug.2023

by Minjoo Kwak, 20-713-764

Supervisors:
Prof. Dr. Renato Pajarola
Haiyan Yang

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

Abstract

Histograms are widely used because they are easy to implement and provide a simple overview of the underlying data. However, histograms are limited to two dimensions and thus not suited for multi-dimensional data. To resolve this, several models have been designed in the existing literature. These typically combine parallel coordinates plot (PCP) with histograms, so that they can represent multidimensional data. However, these existing models typically do not enable clustering of multivariate data or user interaction. To fill this gap, this thesis introduces a new "clustering PHP application" which offers a visual explorative framework with user interaction for the purpose of clustering. This application integrates PHP, Principal Component analysis (PCA), and scatter plots to merge their respective advantages. First, the PCA part offers ideas about variables such as how important they are and how they are related. Variables of interest can then be plotted on the PHP, which was adjusted for clustering (clustering PHP), to visually find relationships between variables. Axes on the clustering PHP can be reordered to focus on specific variables. Finally, a scatter plot helps users to observe local features and allows for the selection of principal components or variables. Interactions are immediately synchronized on the scatter plot and clustering PHP to detect data points sharing similarities on subspaces effortlessly. Overall this "clustering PHP application" thus helps users to determine clustering groups and improve clustering accuracy. In summary, "clustering PHP application" can help a user to explore data and make subspace clustering with complex multi-dimensional data more easy and efficient.

Contents

1 Introduction

Histograms are some of the most popular data analytic tools due to their ease of implementation and ability to show the frequency distribution of the underlying data. However, histograms are not suited for multivariate data because they only have two dimensions. To overcome this limitation, there are several ideas in the existing literature such as combining parallel coordinates plot(PCP) and histograms. For example, there are Parallel Histogram Plot(PHP)[BKS22], screen-space metrics for parallel coordinates[DK10], parallel sets with interactive exploration and visual analysis of categorical data[KBH06], and dynamic query sliders with brushing histograms[LN03]. However, these existing papers do not focus on clustering or include the ability for much user interaction.

To fill this gap, this thesis introduces a new ”clustering PHP application”. This ”clustering PHP application” offers visual designs with user interaction for clustering while taking advantages of PHP, Principal Component analysis(PCA), and scatter plots. Therefore, the PHP redesigned for clustering (clustering PHP), PCA lines, and a scatter plot form the main three parts of the clustering PHP application. To begin with, there are PCA lines that consist of PCA variance explained and PCA contributions. They offer ideas to a user about how important variables are and aid them to find interesting variables. Plus, clustering PHP is extendable and flexible with a user's intention. Desired variables can be compared side by side by reordering axes. Bins can be resized. When a user clicks a histogram bar on the clustering PHP, related attributes are highlighted on the other axes and relationships are shown via lines. These features allow a users to visually find relationships between attributes and help to determine clusters. Lastly, a scatter plot helps a user to observe local features and allows for the selection of principal components and variables. A user can decide clustering with consideration of relevancy and patterns from PCA lines, clustering PHP, and the scatter plot. The clusters can be assigned on the scatter plot with a color lasso selection. The user interaction is immediately synchronized across the clustering PHP. Furthermore, the scatter plot corresponds with any highlighting events from clustering PHP to detect data points which share similarities on certain subspaces easily. As a result, the main visual designs can improve the clustering result. To sum up, this thesis introduces the clustering PHP application, which can help a user to conduct subspace clustering tasks for complex multi-dimensional data more easily and more efficiently.

This thesis is structured as outlined in the following:
After this Introduction, related previous papers which are the foundation of the thesis's concept are discussed and reviewed in section ”2. Related Work”. Following from this section ”3. Problem Statement” addresses problems which are required to be solved in this thesis. Subsequently, in section ”4. Technical Solution”, technical solutions are proposed for each aspect such as data processing, visualization, and accuracy. The respective solution is explained in depth. Section ”5.Implementation” documents the code, dependencies within the code and the underlying mathematical theories. In Section ”6. Experimental Results”, the application is used on code generated sample datasets and real datasets. Finally, in section ”7. Conclusion and Discussion” the conclusion of this thesis is summarized, limitations are discussed, and future work is proposed to extend the given solution.

2 Related Work

2.1 Parallel Histogram Plot(PHP)

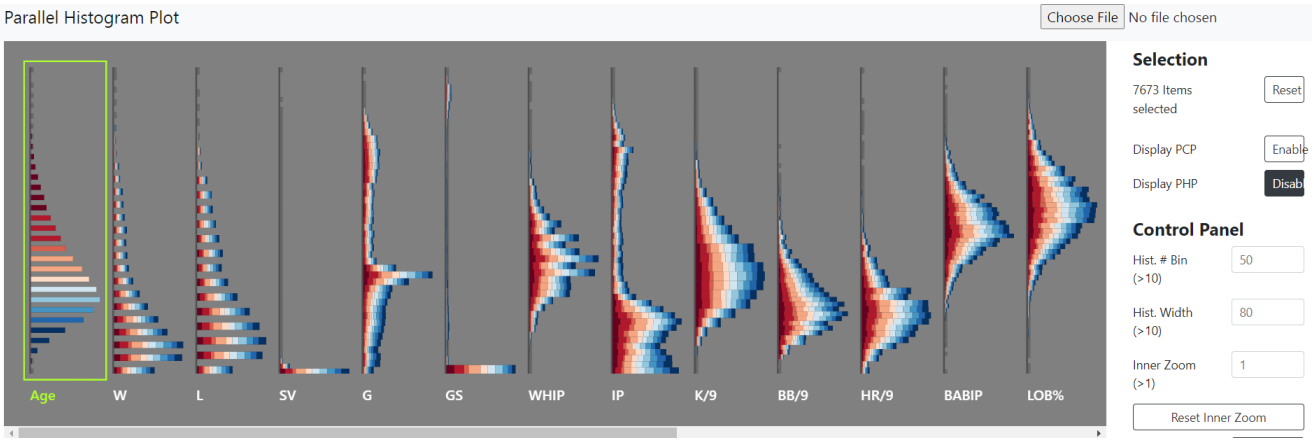


Figure 2.1: An initial overview of Parallel Histogram Plot(PHP)[BKS22].

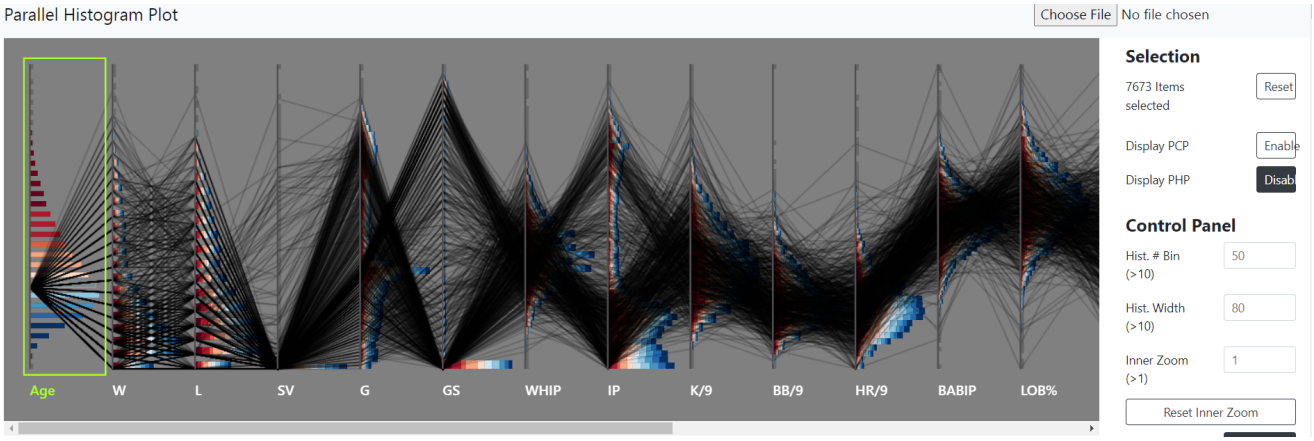


Figure 2.2: After clicking a histogram bar on Parallel Histogram Plot(PHP), lines of PCP are connected[BKS22].

Parallel Histogram Plot (PHP) was introduced by Jinwook Bok et al. [BKS22] to address restrictions of parallel coordinates plot (PCP). The technique is discrete color coded stacked-bar histograms on PCP. The overview of PHP is in figure 2.1. A user can decide a variable with a light green box. Then data is split into equally sized groups according to the user selected attribute and put ranking as the criterion to create groups. Based on the ranking, each group is assigned a color from a ten-level spectrum, ranging from blue to red, representing low to high rankings. The color scheme is created to differentiate between groups and to represent the differences or similarities between them. The other variables, except for the user selected variable, are plotted on a stacked bar chart with each group's color. Plus, related data in the selected histogram bar in figure 2.2 makes lines for PCP. As a result, the color distribution can aid a user to find interesting patterns and relationship among variables even though axes are far away from each other. However, variables can not be changed to compare them next to each other and it is not for clustering.

2.2 Clustering in High-Dimensional Data

Clustering is a unsupervised learning method focused on possible clusters[HP18]. The clusters are classified by similarities or dissimilarities among other sets[Rus69]. However, clustering algorithms often face challenges from the curse of dimensionality because high dimensional data is likely to have noise which hinders clustering effectiveness[HK99]. According to Hans-Peter Kriegel et al.[KKZ09], there are two most common resolutions with dimensional reduction and clustering methods in high-dimensional to solve the problem. First, dimensionality reduction is performed, and then a clustering method is applied [DH04] [JSEB19]. With the process, dimensionality reduction is a global technique with one-dimensionality, but the clusters are no longer separable for returning to the resulting subspace. On the other hand, after clustering data, then dimensionality reduction can be implemented to each resulting cluster. However, it does not contain the correct clustering results anymore. As a result, there is usually no global approach to overcome the challenges of clustering high-dimensional data. It is due to local feature relevance and local feature correlation which influence clustering in the full-dimensional space[KKZ09].

2.2.1 Subspace Clustering

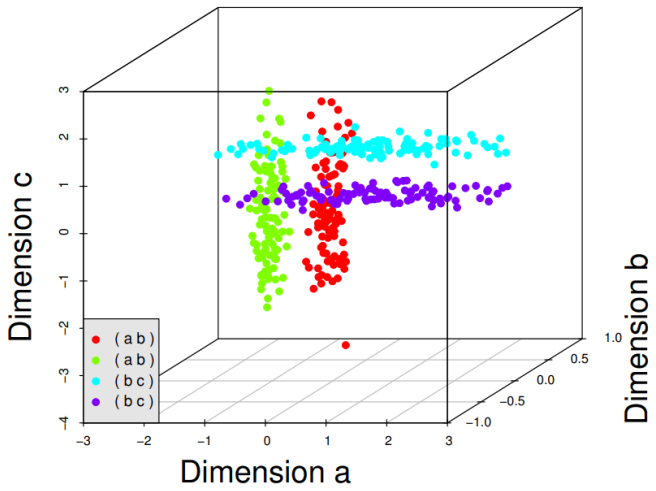
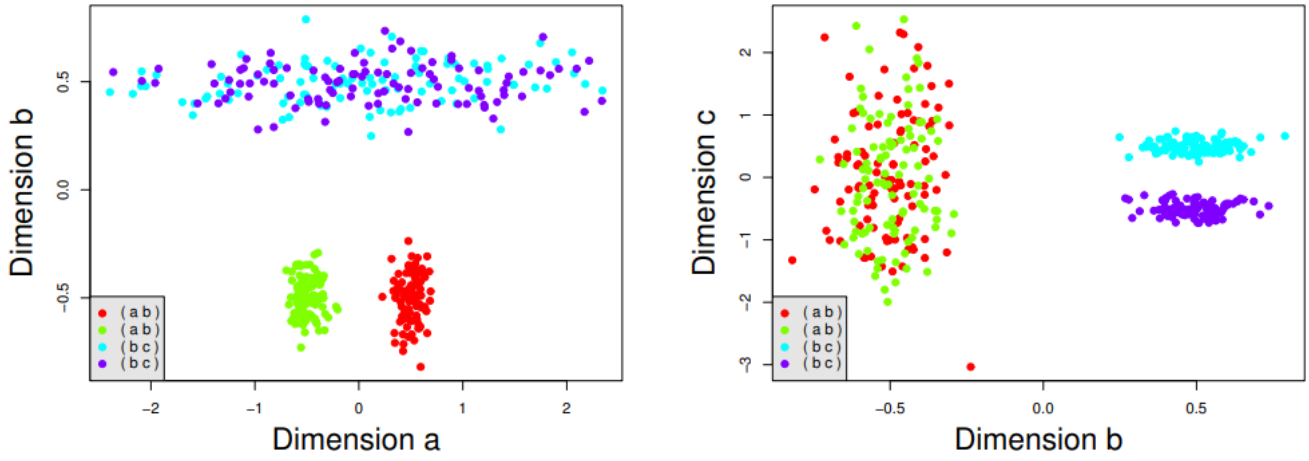


Figure 2.3: Sample dataset to explain why subspace clustering is needed. Many traditional clustering methods can not figure out correct clustering groups since points from two clusters can be very close together[PHL04].

Subspace clustering algorithms is introduced to deal with masking issues in high dimensional data due to irrelevance. Instead of full space, it localizes a clustering search in relevant dimensions to reveal clusters from overlapping subspaces[GFVS12]. Lance Parsons et al.[PHL04] offer a sample data in figure 2.3 to show that traditional clustering techniques such as k-means. When each cluster is distributed over some irrelevant dimension, the traditional methods are difficult to find clusters. The difficulty gets worse in higher dimensional datasets. Hence, it is essential to have appropriate subspaces for accurate clustering. For example, two clusters (red and green) are easily separated in the subspaces a and b in figure 2.4(a). The others(blue and purple) are overlapping. It means because dimension c is the noise to the two clusters (red and green). With the same logic in figure 2.4(b), the blue cluster and the purple cluster are clearly separable in the dimension b and c. However, red and green clusters are mixed in this case. These clusters are masked in the dimension b and c since they were created in dimension a and b. To sum up, some clusters are separated in specific subspaces, while it can be masked in other subspaces.



(a) Dims a & b (b) Dims b & c

Figure 2.4: Points from Figure 2.3 are plotted in two subspaces. In both (a) and (b), two clusters are clearly separated. However, the others are not separable[PHL04].

2.2.2 Clustering Methods with User Interaction

Complex questions can be tackled through data visualization with interactive displays. Results can be improved by allowing the user to explore data with their data analytic skills [SZS+17]. Thanks to the benefit, user interaction is widely used for clustering and dealing with dimensionality such as a interactive-means clustering method[LYB+17], visual interaction with dimensionality reduction[SZS+17], and visual exploration of subspace clusters in high-dimensional data[ZLH+16]. The advantage can be extended to subspace clustering methods as well. Visual user interaction helps users to explore subspaces and discover relationships among clusters. Therefore, it can make a better clustering result. However, it is important to consider the combination of dimensions carefully, since not all subspaces can offer the desired clustering result.[ZLH+16]

2.3 Principal Component Analysis(PCA)

Principal component analysis(PCA) is a dimensionality reduction technique with eigendecomposition. It can show important patterns of the observations and variables[AW10]. By eigendecomposition, eigenvalues show the amount of variance explained by each variable and eigenvectors indicate the weights for factor scores[Suh05]. Principal components are from linear combinations of the original variables in a dataset. The first principal component has the biggest eigenvalue, which can explain the original data the most. The second principal component is perpendicular to the first principal component, and it explains the second most the original data. The rest principal components follow the same manner[HAV13]. Based on two papers [AW10][Kas17],factors scores can be computed as the projections onto the principal components. When an eigenvalue is related with a component, the eigenvalue equals the sum of the squared factor scores of the component. The contribution of component can be calculated with the ratio of the squared factor score. Moreover, the squared cosine represents the contribution of each component for variables. When a squared cosine is high, it indicates a good representation of the variable on the principal component. However, these two papers denotes the squared cosine(cos2) differently.

First, Abdi et al.[Kas17] name cos2 as the squared factor score. cos2 is not a percent of the component. Here is an example decathlon2 dataset from the R package. Cos2(square cosine or squared coordinates) is shown in figure 2.5. Cos2 is displayed with color and the size of circles. For example, Cos2 of 'X100m' is 0.724 on Dim1, 0.03218 on Dim2, 0.0909 on Dim3, 0.00113 on Dim4, and 0.0378 on Dim5. Dim1 has the most cos2 for the variable. 'X100m' has the most contribution on the dimension 1.

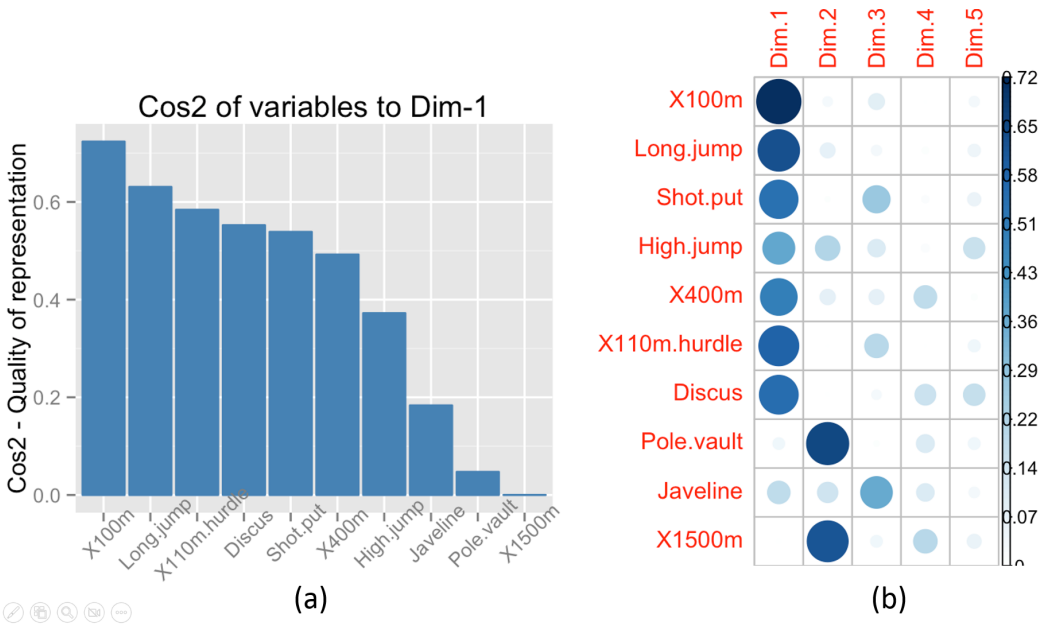


Figure 2.5: (a)An example of variables' contribution to the first principal component (b)An example with contribution of variables to each dimension in full dimensions. cos2 is displayed with color and the size of circles. Abdi et al.[Kas17] name cos2 as the squared factor score. cos2 is not a percent of the component.

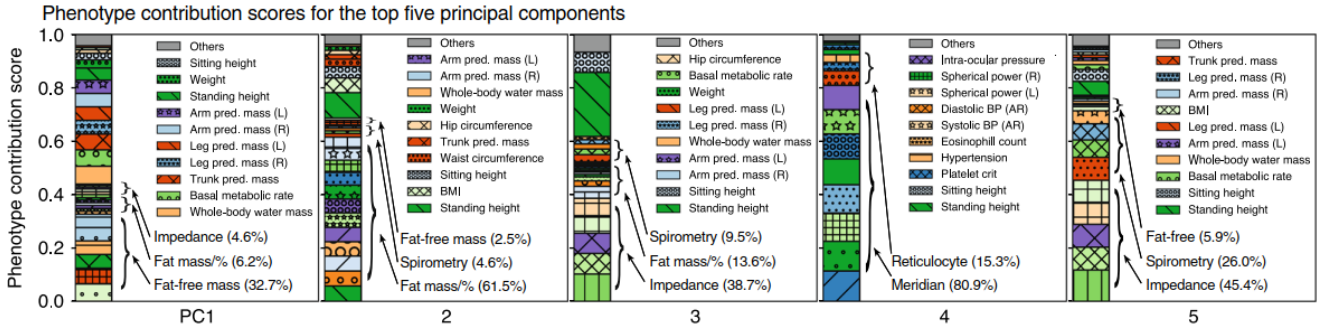


Figure 2.6: Contribution is named squared cosine scores(cos2) by Tanigawa et al.[TLJ+19]. They take cos2 as a percent of the component. cos2 is the value of the squared factor score divided by the sum of squared factor scores of the component. This denotation is taken in this thesis.

$$cos2 = \frac{squared\ factor\ score}{Sum\ of\ all\ squared\ factor\ score\ of\ the\ component} \tag{2.1}$$

On the other hand, Tanigawa et al.[TLJ+19] name contribution Squared cosine scores(cos2). cos2 is a percent of the component as shown in figure 2.6. cos2 is the value of the squared factor score divided by the sum of squared factor scores of the component. This denotation is taken in this thesis. This thesis takes contribution as squared cosine scores(cos2) by Tanigawa et al.[TLJ+19] on (2.1).

3 Problem Statement

This thesis is to develop a PHP software(clustering PHP application) that can be used to help users conduct subspace clustering easier and more efficiently. To accomplish this goal, the three main parts of the software try to accomplish the following:

1. Data Processing:

To run the application, the following data processing steps are required. Firstly, the software needs to compute of principal components. Secondly, it should enable the observation of local features to help the user with cluster determination. For that, it is essential that the software allows user to search and locate the local features. Finally, the program shall run smoothly even when handling big data. In summary, with regards to data processing the program needs to include the following:

- a. Computing the principal components of the dataset.
- b. Efficient search and location of the local features for each cluster.
- c. Handling potential clutter issue when the data size is large and the scalability issue when the number of attributes grows.

2. Visualization:

In step 2, processed data is visualized. Thereby the visualization shall allow for the following. All plots should be adjustable by the user via interaction. The application should show explained variance and contribution ratios of the PCA. Histograms such as PHP should be re-sizable by the user to account for limited space in the visualization and enable focus. Finally, the user should be able to select two principal components or two variables and generate a scatter plot plotting these. By using these all user interface, a user can determines clusters. In summary with regards to visualization the program shall allow for:

- a. Plotting principal components above the PHP plot and find relationships between any Principal Components and any attribute.
- b. Adding a scatter plot on the side to visualize the dataset under any two selected PCs or attributes.
- c. Color-coded PHP : Extend each attribute of the PCP into histograms, with colors indicating clusters.
- d. Building a PCP to visualize multidimensional data.

3. Accuracy :

In step 3, this thesis utilizes accuracy to evaluate performance of the application. Before calculating accuracy, determined clusters by a user from step2 are needed to find the best match between labels from a user and ground truth labels. Afterward, the clustering PHP application should allow the user to compare the accuracy to evaluate the application's performance.

4 Technical Solution

4.1 Data Processing

a. Computing the principal components of the dataset.

Computing the principal components and contribution are mathematically explained in the section 5.4 PCA lines. Here is a brief explanation of the contribution based on the section 2.3.Principal Component Analysis(PCA). By Principal Component Analysis(PCA), there are eigenvectors corresponding with eigenvalues. Explained variance is computed as a ratio of an eigenvalue with total eigenvalues. This application uses a library from Scikit-learn for PCA, so eigenvectors are calculated in a matrix in a row-wise direction. Therefore, the result should be transposed to get a matrix of eigenvectors. Then, factor scores and squared cosine scores are calculated by eigenvectors and eigenvalues. As mentioned on the section 2.3, Tanigawa et al.[TLJ+19]'s squared cosine scores(cos2) is considered as contribution because the ratio is easy to be compared and represented.

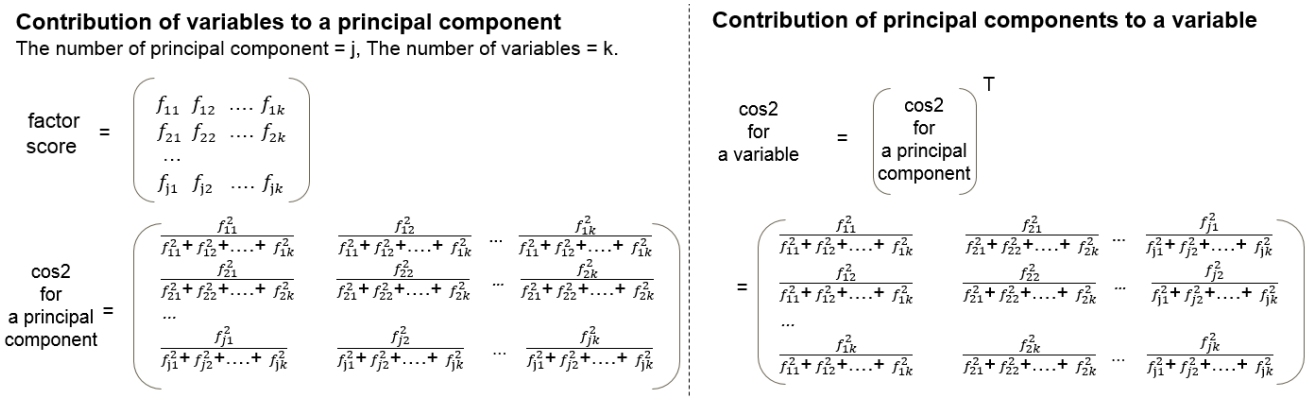


Figure 4.1: Brief explanation of how to get the contribution by using squared factor scores. The left part explains the distribution of variables to a principal component and the right one is the distribution of principal components to a variable.

Figure 4.1 explains data of PCA lines. The number of principal component is j and the number of variables is k. On the left of figure 4.1, cos2 means the distribution of variables to a principal component. For example, the first row of cos2 explains the contribution of k variables to the first principal component. Then the second row is about the contribution to the second principal component. On the right of figure 4.1, cos2 for the contribution of j components to a variable comes from a transpose of cos2 for a principal component. Therefore, the first row of the cos2 for a variable indicates the first variable's distribution on j principal components. In this thesis, the number of principal component is the same as the number of variables due to offering ideas of all variables and principal components. In the other word, the number of principal component is k and the number of variables is also k.

b. Efficient search and location of the local features for each cluster.

Since clusters are decided by a user, it is important to observe local features. Especially, with which values the cluster are decided and how the cluster changed are crucial. Therefore, a user can see histories of local features

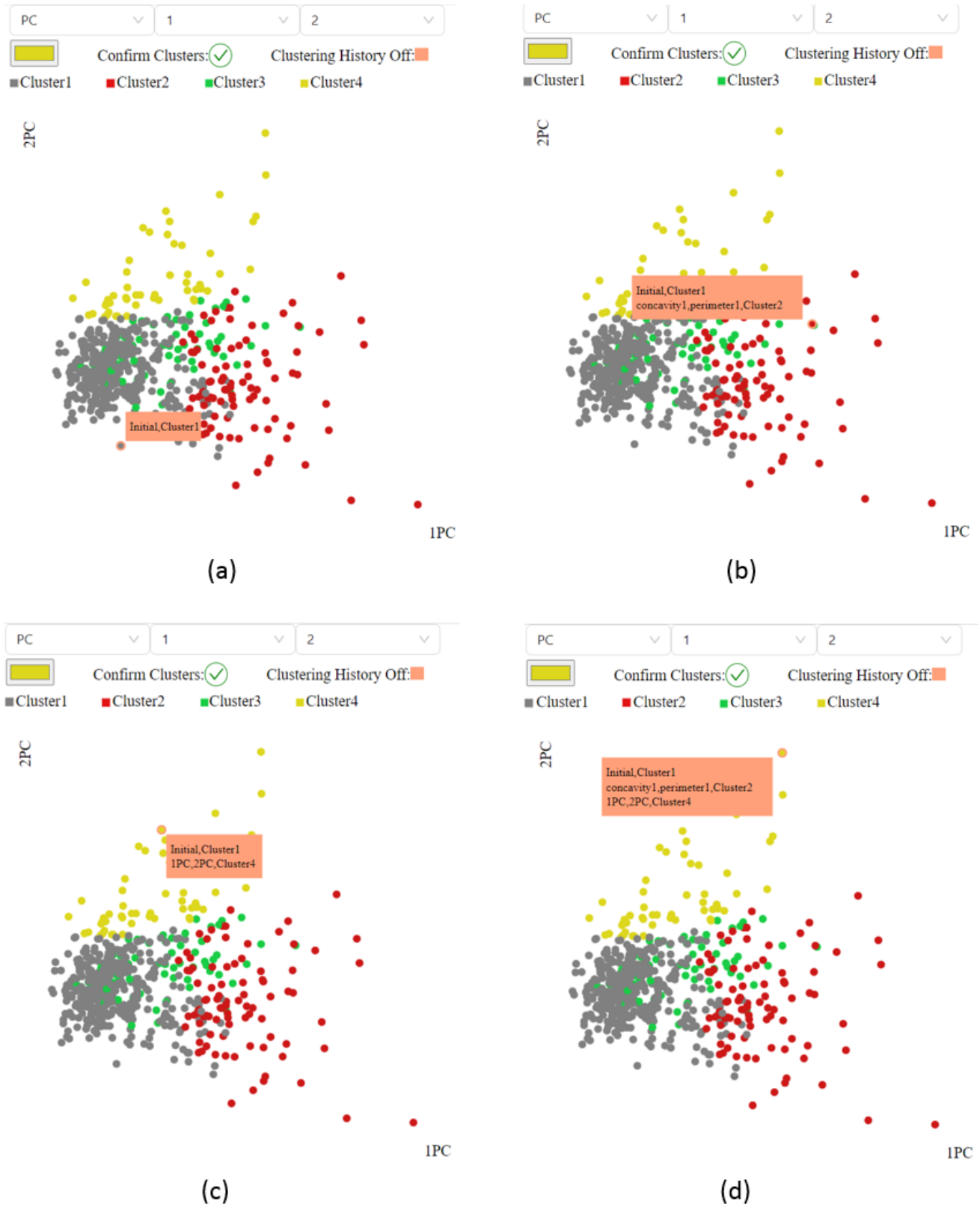


Figure 4.2: Local features for each cluster can be represented as a text box. The location of the text box is flexible depending on the selected point. Based on the center of the scatter plot, there are four types of location on (a)selected point(left lower), text(right upper) (b)selected point(right lower), text(left upper) (c)selected point(left upper), text(right lower) (d)selected point(right upper), text(left lower)

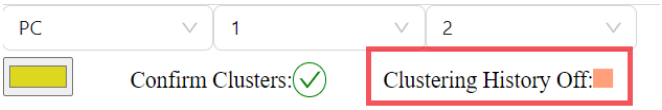


Figure 4.3: The red highlighted part points to a function for removing the text box by clicking the pink rectangle.

for each cluster in the application. The information is saved in a library with a key as ID and a value as all history of local features. After a new cluster is updated, the information is stored by ID which is a key. This library searches efficiently and locate the update with IDs. The history of local features contain a pair of variables and a cluster's name. When a point on scatter plot is clicked, a pink text box pops up with clustering history of the selected point in figure 4.2. The text box is flexibly located to display all texts wherever the point is. To avoid clipping, the text box is set to a visible location according to a point's location in the plot. To be specific, Figure 4.2(a) has a selected point on the left lower part, so the text box is designed toward a right upper side. As the selected point is on the right lower plot in figure 4.2(b), the text box is positioned on the left upper part. As the same manner, figure 4.2(c) and figure 4.2(d) are composed. There is a pink rectangle in figure 4.3. It clears the text box.

c. Handling potential clutter issue when the data size is large and the scalability issue when the number of attributes grows.

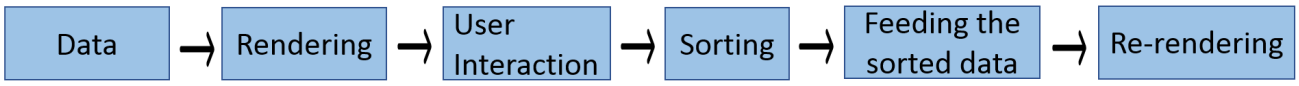


Figure 4.4: A common process to plot data with user interactions.

Figure 4.4 shows a common process to plot data on user interface. However, the larger the size of data grows, the more clutter issues increase. Each process can make performance slow as Figure 4.5.

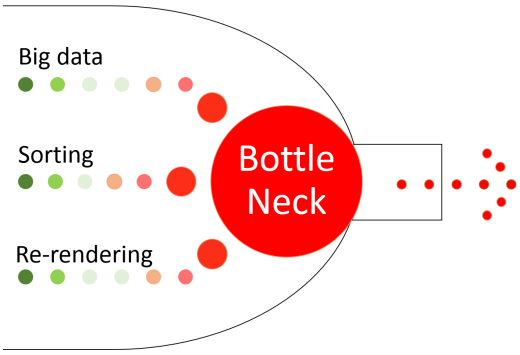


Figure 4.5: Three main causes which make a bad performance on the process in figure 4.4.

Therefore, there are ways to relieve the bottle neck. To begin with, the big size of clustering PHP's data can be reduced by filtering data only for the interesting part. Since this thesis mainly focuses on detecting relationships among variables, plotting the full data might be redundant on clustering PHP. Even the full plot makes interesting axes far away from each other. Therefore, plotting the full data is unnecessary and not efficient. As a solution, a user can select interesting variables, and then only the selected data will be on the data process and render a plot. This shorter version makes the whole process faster. The less data can make a rendering load lighter as well.

Plus, when the number of attributes grows, it is difficult to keep a consistent coding style in D3. In figure 4.6, (a) is an adjusted version from the original dataset with n instances and k variables by adding an ID on each row

from clustering PHP. Plus, a variable makes a dimension of cluster PHP. Each dimension has an axes and they are rendered one by one from top to bottom in this application. Since clustering PHP is based on histograms, the original data is expanded by data binning. There are two ways to extend data of clustering PHP such as figure 4.6(b) and figure 4.6(c). First, figure 4.6(b) is horizontally extended but codes on D3 should be always changed in the manner of the extended columns but it causes errors with FOR loops in D3 and updating plots. Second, the clustering PHP data is vertically expanded in figure 4.6(c). Whenever data has more attributes, the column number is always the same as three. This constant data structure makes a plot in the consistent manner of codding in D3. Therefore, figure 4.6(c) is better to make the code easier and get correct results.

ID	Attribute 1	Attribute 2	...	Attribute k
0				
1				
2				
...				
n-1				

(a) Data with ID

ID	Attribute 1	#Bin	Attribute 2	#Bin	...	Attribute k	#Bin
0							
1							
2							
...							
n-1							

(b) Clustering PHP data in a horizontal style

ID	#Attribute	#Bin
0		
1		
...		
n-1		
0		
1		
...		
n-1		
...		
0		
...		
n-1		

(c) Clustering PHP data in a vertical style

Figure 4.6: (a)Assigning IDs on the original dataset to find and updated information efficiently. (b)After dividing dataset of (a) by bin ranges and binning, the dataset is extended horizontally (c)After dividing dataset of (a) by bin ranges and binning, the dataset is extended vertically. This vertical style is applied on this thesis due to its efficiency of code and correct results.

With user interaction, hovering is a common tool to trigger events. However, it is not an appropriate technique to highlight instances of clustering PHP in this thesis since clustering PHP is built with small rectangles which are associated with instances. For example, when a user wants to highlight a histogram bar and moves a mouse slightly within the histogram bar, the small movement provokes the unnecessary re-rendering and makes performance slow. For this reason, highlighting is performed by clicking which fires an event only when a user wants.

Sorting data is an essential key in this thesis to detect interesting patterns. Reasons for sorting are explained in the following section 4.3.Visualization.a. This thesis is built on JavaScript which is widely used for web applications, but sorting is much slower than the other programs such as Python and C[ANB21]. Hence, this thesis takes the best performance among ways of sorting. More details are explained on the section 6.1.Sorting Speed.

As carrying out user interaction, re-rendering is fundamental but it takes a long time, especially with a large size of data and more attributes. Thus, cutting unnecessary re-rendering chains with a parent component and child components can help the process faster. Accordingly, the application is designed with a certain condition for re-rendering. For example, events trigger rendering the specific child component again. When a related variable changes in a parent component, only the linked child components run again instead of the whole child components. Moreover, using classes with CSS instead of rendering all components again can boost performance to perform a user request for highlighting elements. It is because CSS changes features from existing components. In contrast, re-rendering builds everything from the beginning, so it is more expensive. Accordingly, this

application utilizes CSS with rectangles and lines on clustering PHP and scatter plots.

To sum up, as considering all solutions, the clustering PHP application can handle the clutter issue and the scalability issue with the optimization in figure 4.7.

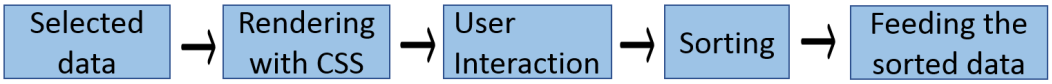


Figure 4.7: An optimized process to plot data with user interactions. It can handle clutter issues and the scalability issues.

4.2 Visualization

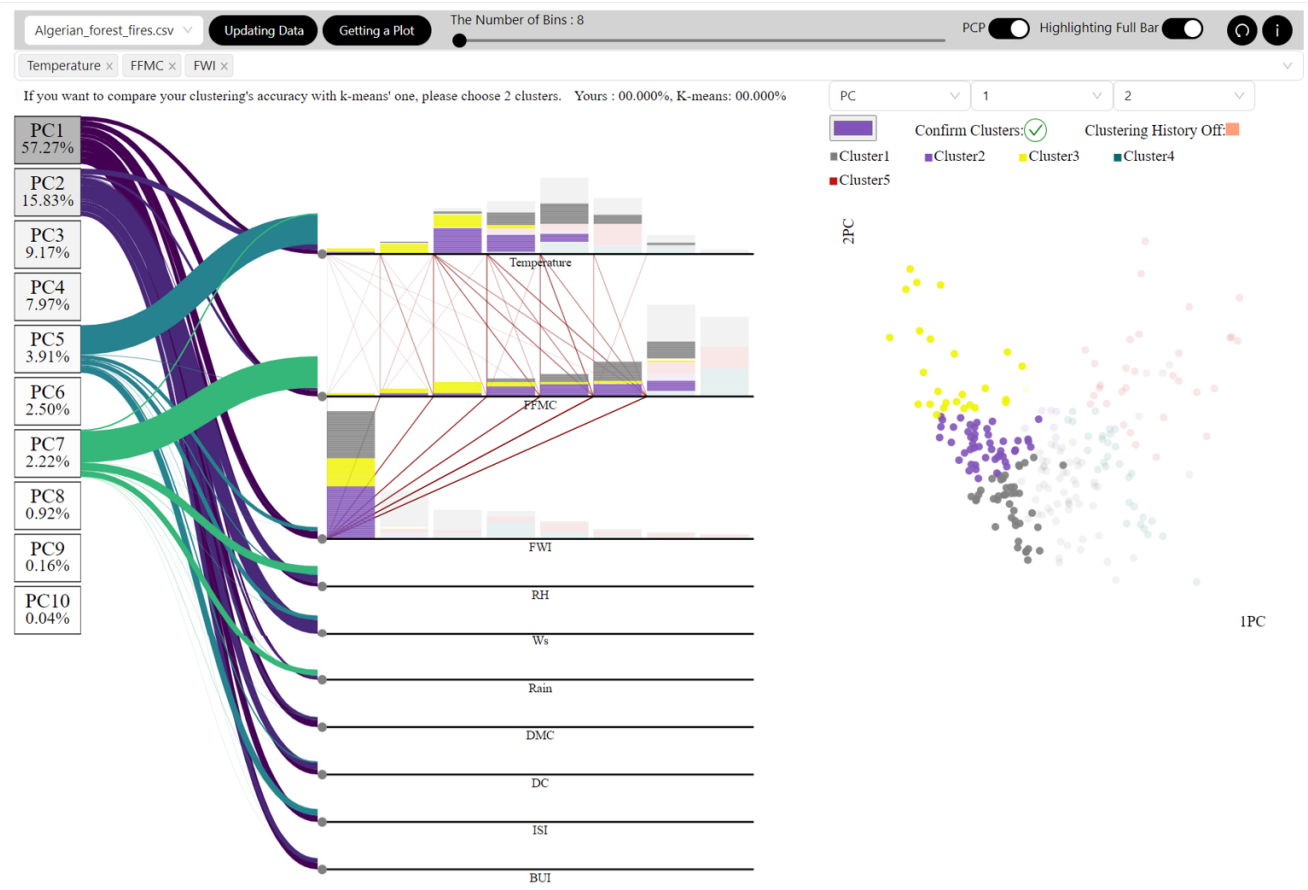


Figure 4.8: Overview of visualization

Figure 4.8 is an overview of a new application for this thesis. There is a grey box which contains setting functions and user interactive functions at the top of figure 4.8. To deep dive into setting functions, there are such as a drop-down selection for choosing a dataset, buttons to communicate between backend and frontend. Next to the setting functions, there are user interactive functions. They consist of a slider to change a bin size for clustering PHP, switches to turn on and off lines of PCP and to swap highlighting options, circle buttons to reset and to give a tour of how to use this application. Under the grey box, there is a selection of variables which will be plotted on clustering PHP. The selected variables can be deleted and added anytime. The change will be updated immediately on the clustering PHP and PCA lines. A user can find information regarding a clustering accuracy below the selection of variables. Under the accuracy text, there are three main plots. From the left side, rectangles

to show explained variance of PCA. The lines are connected to clustering PHP plots. PHP plots can be extended and shrunken by user interaction to overcome space limitation of PHP and PCP. At the right side, a scatter plot is displayed with two principal components or two variables. Clusters can be decided on this scatter plot by a user. To conclude, the functions and plots allow a user to explore data and aid clustering accuracy better.

Figure 4.9: Initial setting

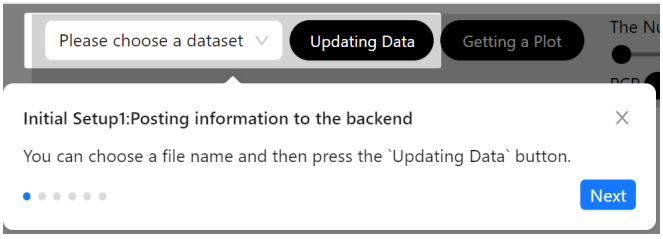


Figure 4.10: After clicking the circle button in the red 'D' in figure 4.9, it offers a tour to explain functions and steps to follow.

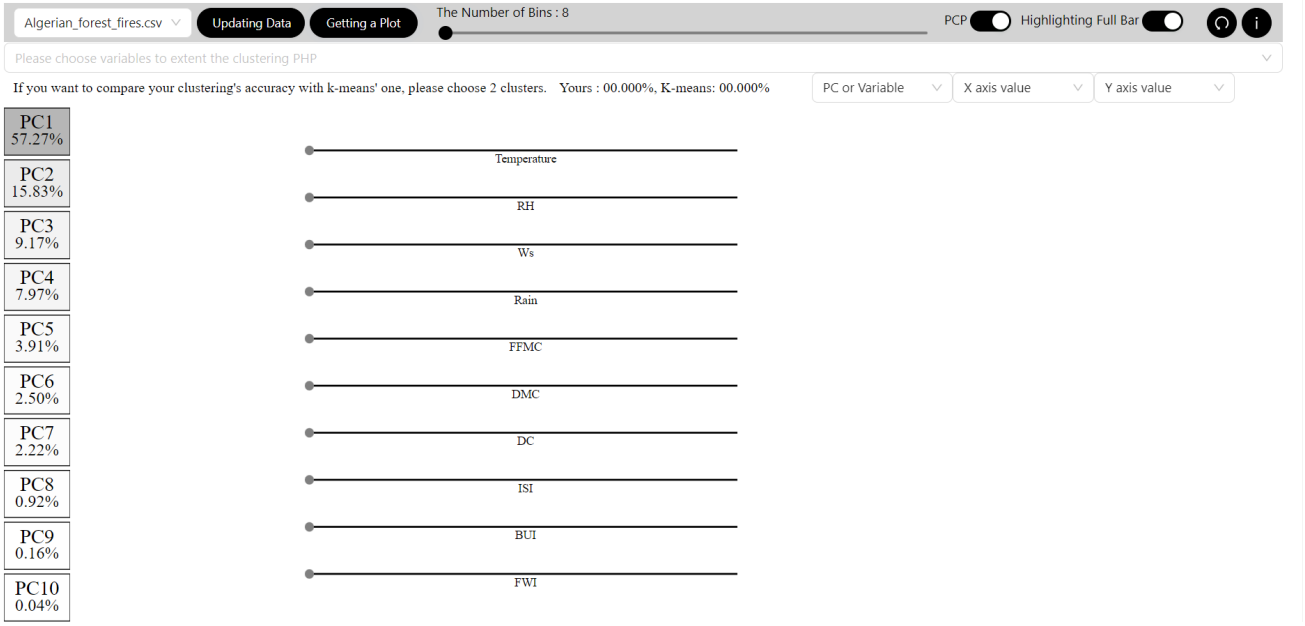


Figure 4.11: Initial plots after pressing 'A','B', and 'C' in figure 4.9.

To start the application, a user should follow a certain order of 'A', 'B', and 'C' with functions in figure 4.11. On 'A', a user should choose a dataset among stored datasets on the backend. Then, the 'Updating Data' button on B, information is posted from frontend to backend such as setups and updates from user interaction. Based on the posted information, the K-means technique and PCA algorithm are executed. On the backend, labels are assigned on clusters and accuracy of algorithms are calculated. After pressing the button 'Getting a Plot' in 'C', the frontend will show plots and accuracy. After pressing 'A', 'B', and 'C' in order, a user will get a initial set up as figure 4.11. The accuracy of Yours and K-means is zero because no clusters are decided yet by a user. Since

certain steps are required, there is a tour guide as figure 4.10 by clicking the 'i' button in the red 'D'. Additionally, at the second last end of the grey box there is a reset button which is a circle shape. It resets all under the grey box.

a. Plotting principal components above the PHP plot and find relationship between any Principal Components and any attribute.

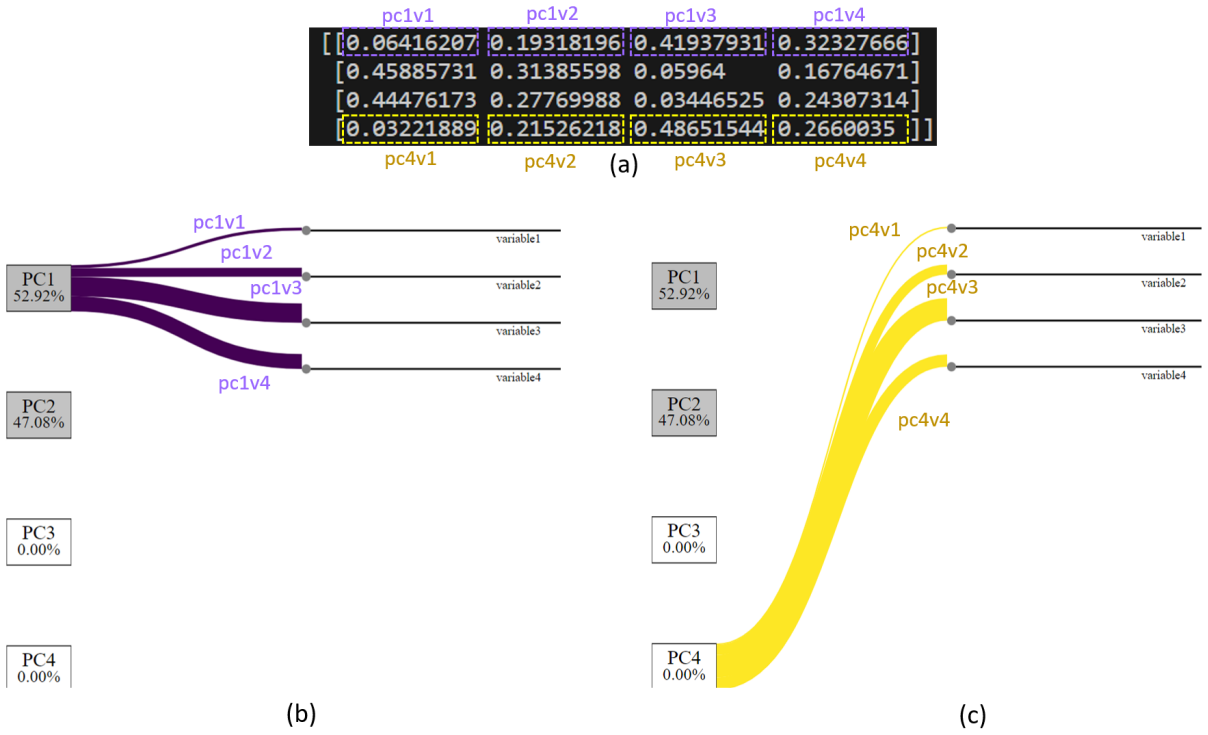


Figure 4.12: An example of contribution of variables to a principal component. The lines are named PCA lines.

On the left side of the application, PCA variance explained ratios in percent are represented inside of rectangles like figure 4.12(b). From the top, principal components are ordered by increasing order such as the first principal component(PC1), the second principal component(PC2) and so on. The PCA variance explained is also shown as percentage and color. When the variance explained is high, the rectangle with the principal component is deep grey. When it is low, the related rectangle is close to white. Plus, the calculated contribution from figure 4.1 is represented on lines named PCA lines in the clustering PHP application. A thickness of a line shows the contribution. When a line is thick, it means the ratio is big, so the contribution is high. Lines color scheme is from purple to yellow starting from the top to bottom. For example, lines from PC1 are purple and lines from the last component are yellow. Similarly, lines from the first variable are purple and lines from the last variable are yellow.

Figure 4.12(a) is an example result from the calculation of contribution on the left side of figure 4.1. It explains the contribution of variables to a principal component(PC). Since the result is a ratio and the sum of each row is 1, lines' thicknesses are by multiplying the result with the height of a rectangle. If the thickness is under 0.1, it will be hard to see. Therefore, the thickness below 0.1 is converted in 0.1 for visibility. The first row of figure 4.12(a) is in the purple notations. They indicate data for PCA lines of the first principal component. Its color scheme is purple and is plotted in figure 4.12(b). The notation means the number of component and variable. For example, pc1v1 means the first component and Variable1. As pc1v1 is the thinnest among purple lines in figure 4.12(b), Variable1's contribution is the lowest on PC1 and the others are similar. In figure 4.12(c), yellow lines show contribution to PC4 which are from the last row of figure 4.12(a). pc4v3 is the thickest, so Variable4 is contributed the most to PC4. On the contrast, Variable1 contributes the least to PC4 because pc4v1

is the thinnest.

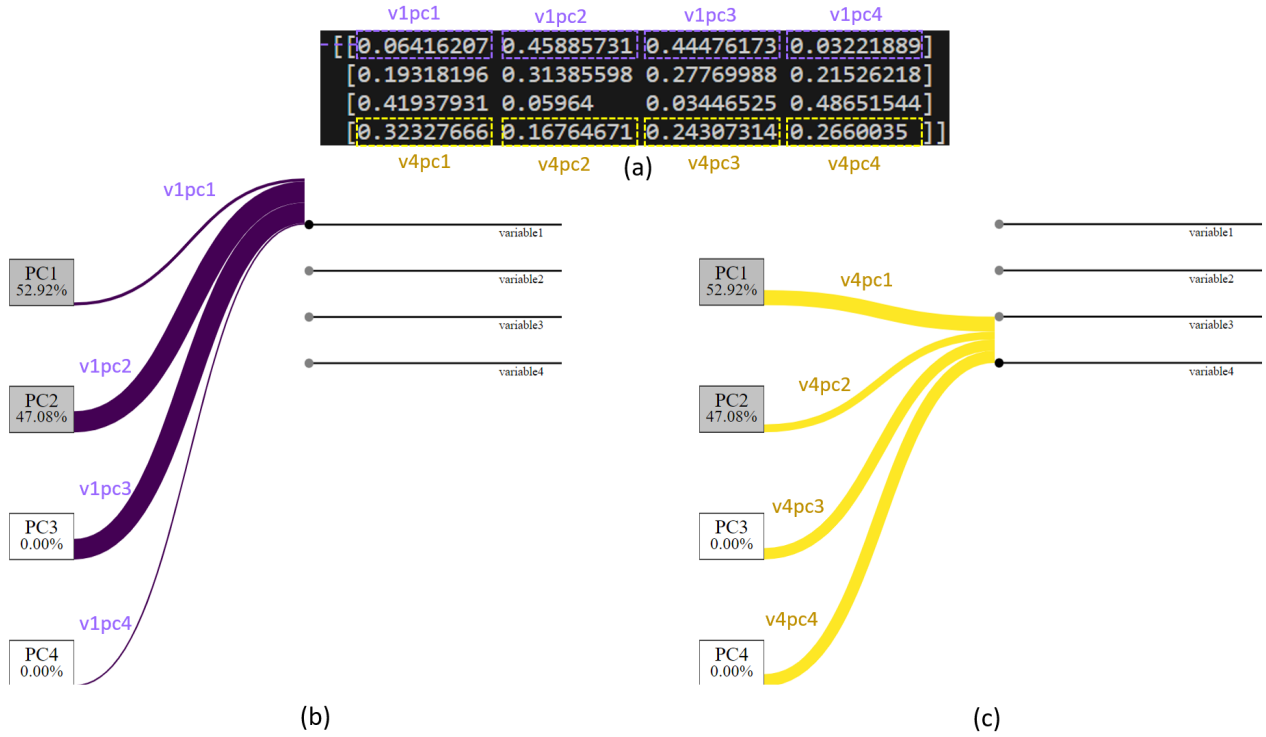


Figure 4.13: An example of contribution of principal components to a variable. The lines are named PCA lines.

Similarly, figure 4.13 shows contribution of principal components to a variable. The result is from the right side of figure 4.1. It has the same color scheme as figure 4.12. The first element in the first row denoted v1pc1 on the first line, then v1pc2 and, so on. v1pc2 and v1pc3 are pretty similar and relatively bigger than the others, so their lines are thick enough and similar on figure 4.13(b). It means that PC2 and PC3 contribute more than the others to Variable1. Since v1pc4 is the smallest among purple numbers, the thinnest line is v1pc4's one on figure 4.13(b). lines on figure 4.13(c) show the yellow number. The lines are made as the same manner in figure 4.13(b). They explain how much PC1, PC2, PC3, and, PC4 contribute to Variable4.

Figure 4.14 is an example of multiple selected PCs. These PCA lines can give hints to a user which variables are interesting since the thickness of lines indicates how big contribution is. With the hint, a user can expand variables on clustering PHP and are plotted on the scatter plot and explore more with variables. When the principal component is highly contributed to the variable, the related line is thick. When the component makes little contribution to the variable, the line is thin. Figure 4.14(a) is a case of well separate lines. With these clear lines, it is better to choose the widest sum of purplish lines which might mean important variables and to remove variables with yellowish lines which might indicate redundant variables due to PCA explained variance. Therefore, a user can have a look first at 'Temperature', 'RH', 'Ws', 'FFMC', and 'Rain' among many variables. Afterward, if clusters are ambiguous and need more information, the rest variables can be useful to be checked. In contrast, if PCA lines are duplicated as figure 4.14(b), it is better to turn off yellowish lines and start to try variables from the thickest variables to the thinnest one. After exploring clustering PHP and a scatter plot, a user can decide variables to explore more. Plus, figure 4.15 is an example, when multiple PCA lines are selected for contribution from variables to principal components.

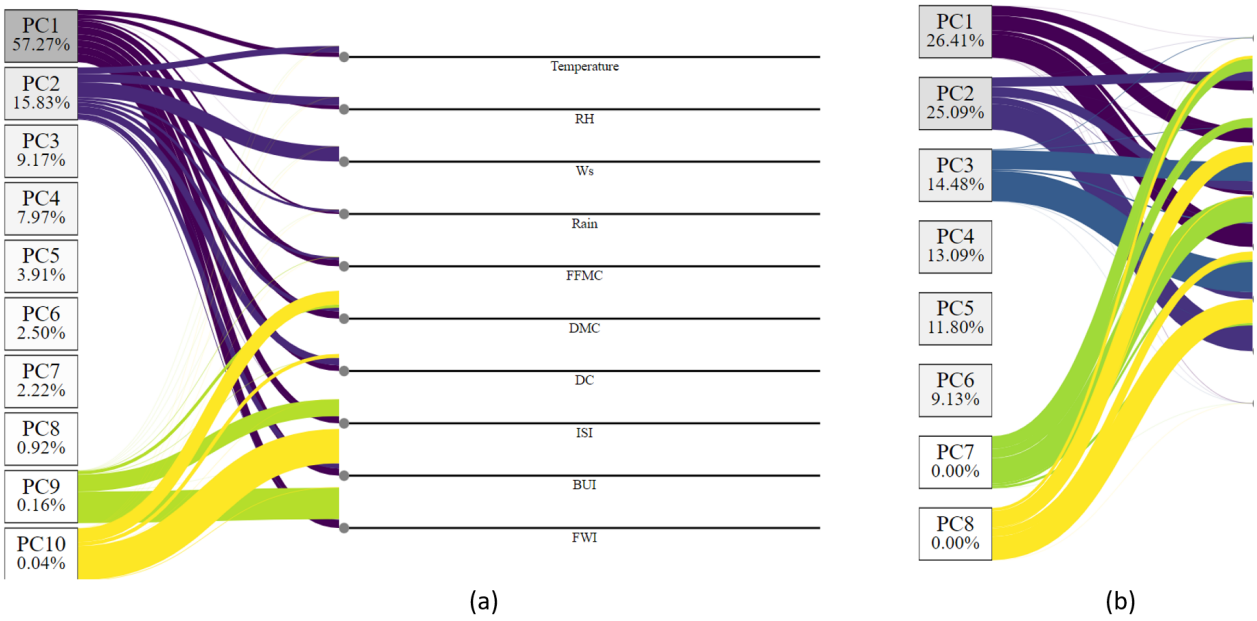
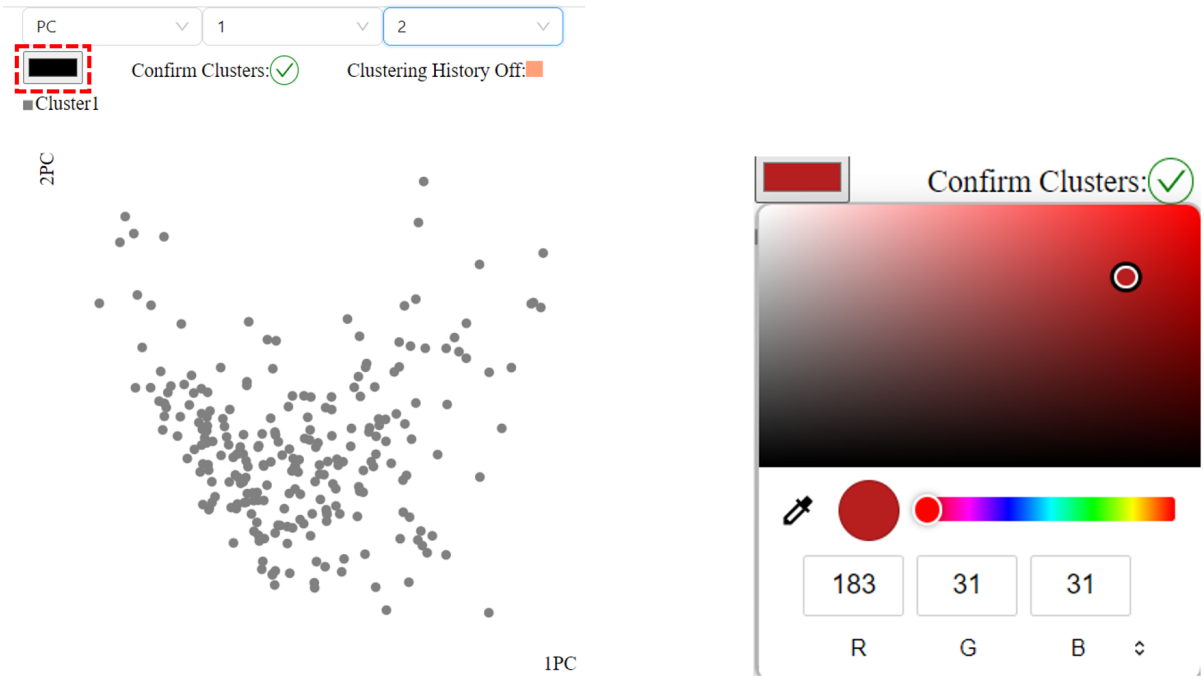


Figure 4.14: An example of multiple PCA lines on PHP plot. However, these PCA lines do not perform variable selection but give hints to a user which variables might be interesting. (a) It is a separate case. A user can start to have a look from the widest sum of purplish lines but not with yellowish lines. It is because purplish lines might mean important variables and yellowish lines are likely to be redundant variables. Therefore, a user can have a look at 'Temperature', 'RH', 'Ws', 'FFMC', and 'Rain' first among many variables. (b) PCA lines are overlapped. In this case, turning off yellowish lines can be helpful. Variables of purple lines can be checked and selected.

Figure 4.15: An example of multiple PCA lines for multi-contribution of principal components to a variable.

b. Adding a scatter plot on the side to visualize the dataset under any two selected PCs or attributes.



(a) (b)

Figure 4.16: (a)A scatter plot with a combination of 'PC', '1', and '2'. It represents the first principal component's data on the x-axis and the second principal component's one on the y-axis. (b)A color palette for color decisions. Each color indicates each cluster.

At the top of figure 4.16, there are three drop-down selections for creating a scatter plot. The First one has two options such as 'PC' and 'Variable'. If a user selects one of the first selections, the corresponding information will be appended sequentially as options in the next selection. The second selection's data is plotted on x-axis wise data and the third selection will form the y-axis' data on the scatter plot. For example, figure 4.16 takes 'PC', '1', and '2', so the scatter plot has the first principal component's data on x-axis and the second principal component's one on y-axis. The red highlight part points a color picker where a user can set the color of dots on the scatter plot. After clicking the picker, a color palette is open like figure 4.16(b). Each color indicates each cluster. With a lasso selection, a user can drag a mouse flexibly and points inside of the grey area are changed to the decided color. Moreover, the color decision also reflects on clustering PHP as figure 4.16. Furthermore, a user can take the same color previously assigned again after clicking a rectangle in front of a cluster's label. Plus, any colors can override old color. The changes of color are recorded as a history in figure 4.2. Additionally, clusters can be empty since some points are all overridden by other clusters. In this case, the empty cluster remains the label in the scatter plot but the label is crossed out in figure 4.18. It is because it is better to keep the empty cluster to understand how points are changed on the clustering history. The crossed cluster does not count on the number of clusters to get accuracy. For example, the number of clusters is three in figure 4.18.

c. Color-coded PHP : Extend each attribute of the PCP into histograms, with colors indicating clusters.

Bok et al.[BKS22] introduced PHP which is a powerful technique to find relationship visually among variables in high-dimensional data. However, it is not proper to show clustering methods since it already has ten colors for histogram bars. To indicate clusters, adding more colors for each cluster would lead to further complicate

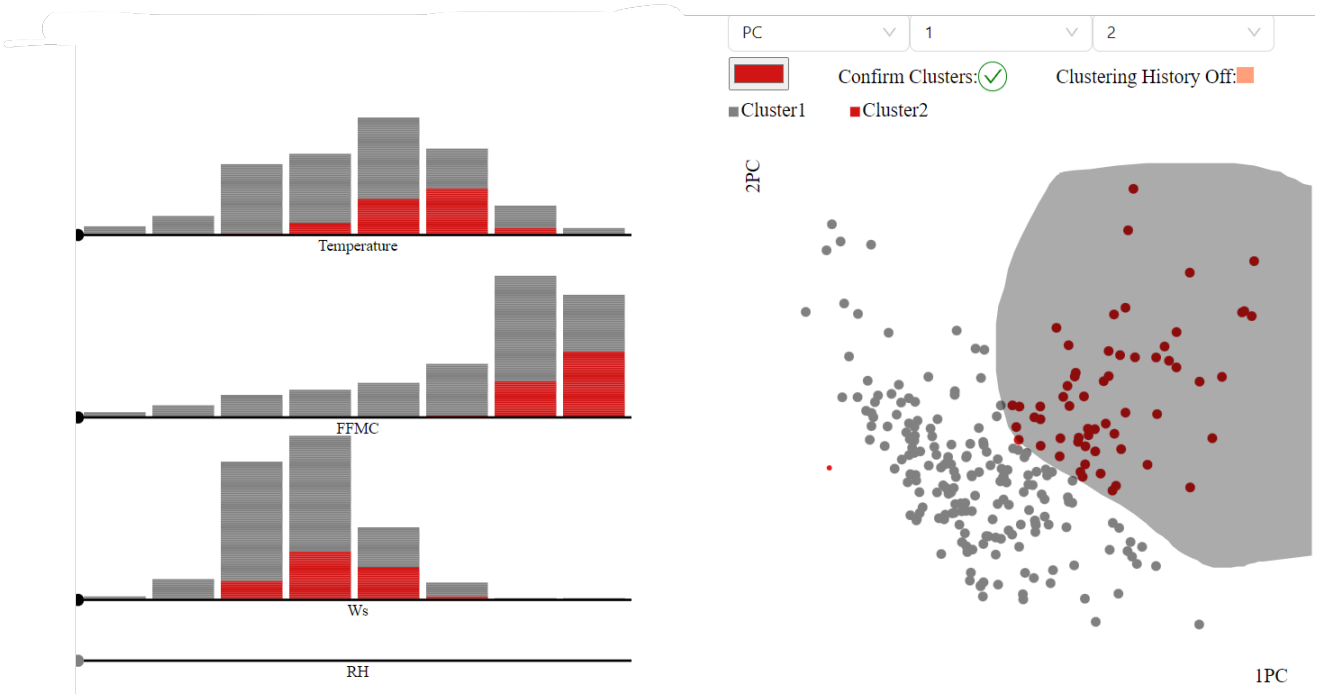


Figure 4.17: After dragging dots, related data points are changed into red on the scatter plot and the clustering PHP.

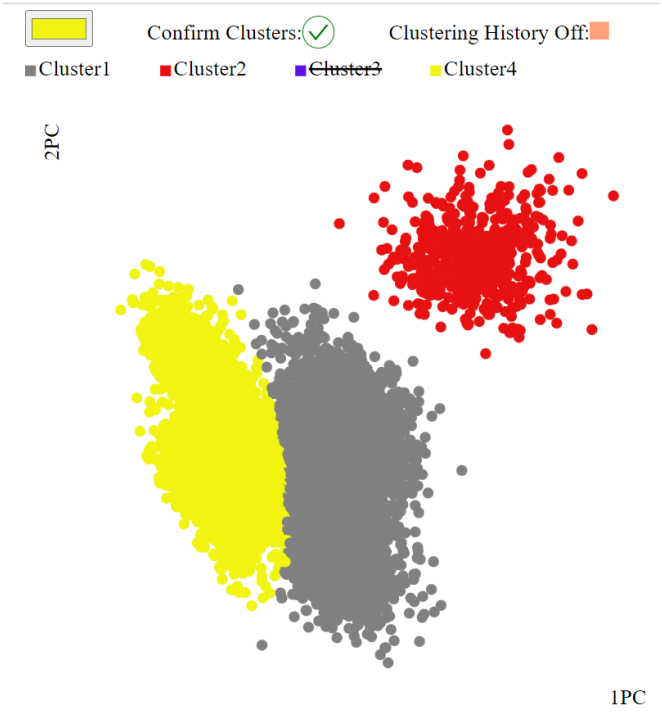


Figure 4.18: If there are no points anymore in the previous cluster, the text of the previous cluster is crossed out.

the visualization. Moreover, the complication distracts potentially users from identifying interesting patterns of variables' relationship. Therefore, this application adjusts the original color scheme to a clustering color scheme. As a result, the application employs advantages from PHP to reveal relationship among variables for clustering.

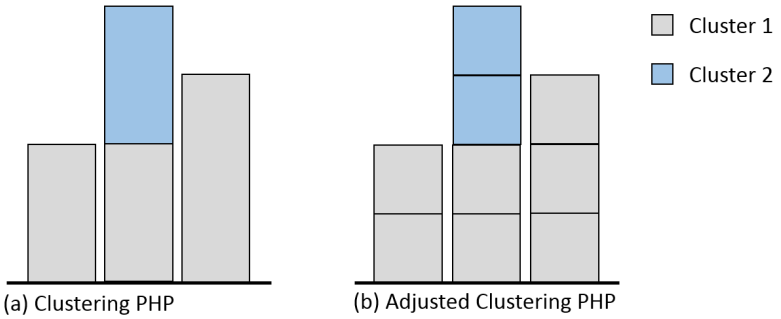


Figure 4.19: (a)A sample example with one variable to explain the 4.3.c solution easily. It is named clustering PHP which takes a modified color scheme from the original PHP. The color scheme represents clustering groups. (b)It is an improved version of (a) to explore variables' relevance. It is especially useful with multivariate data

To begin with, here is a simple sample dataset to explain this thesis' solution. The data has nine instances, one attribute, and two clusters. The PHP's color scheme is modified for representation of clustering groups in figure 4.19(a). However, it is not enough to explore variables' relevance, so clustering PHP is adjusted as figure 4.19(b). Furthermore, figure 4.20 explains the solution clearly with two variables. The full stacked bar chart can not see variables' relationship in the other dimensions in figure 4.20(a). When one full histogram bar on Variable 1 is highlighted, all bars are wrongly highlighted which contain the selected instances. To solve the problem, the full histogram bar is split into small rectangles with the same color scheme by clusters and each small rectangle indicates an instance. As a result, the correct instances are highlighted in figure 4.20(b). However, the plot is difficult to represent interesting patterns since clusters are separate and mixed up. Therefore, sorting instances by cluster on each histogram bar is needed. Finally, highlights are displayed on the right instances and patterns are easily detected with the organized plot in figure 4.20(c). The sorted separate PHP is implemented in the cluster PHP application.

In addition, clustering PHP can be extended by a user. The red part in figure 4.21(a) opens variables selections as figure 4.21(b). By selecting the variable's name, a user can compare the desired variables closely. Clustering PHP is drawn from the top in the existing clustering PHP on figure 4.21(c). After a user clicks the 'X' sign next to the variable's name of figure 4.21(b), the variable's bar on the histogram disappears. The original PHP does not allow a user to change variables' order, so it is difficult to compare far away variables. Moreover, multivariate data extends histograms of PHP infinitely, so the increased space is difficult to see at a glance. If a specific space is defined for viewing at a glance, each histogram will be compressed to a very small size to represent all variables in it. The compression interrupts observation due to very tiny histograms. However, thanks to the deleting and adding variable function in the application, clustering PHP can overcome a limitation of the original PHP. Clustering PHP can be small and big up to the selecting option, so the histograms have enough space to be observed. Plus, any variables can be located adjacently to each other to compare variables simply and detect interesting relationships easily.

After deleting a variable from figure 4.21(b), the corresponding PCA line and the variable' axis of the deleted variable return to its original order. The positions of the corresponding PCA lines also move accordingly immediately. Plus, there is a slider above the red dashed box in figure 4.21(a). With the slider, a user can change the bin size of the clustering PHP. This function offers flexible exploration in the application. The flexibility is especially useful with big size of datasets because it allows a user to investigate data closely and to focus on interesting parts

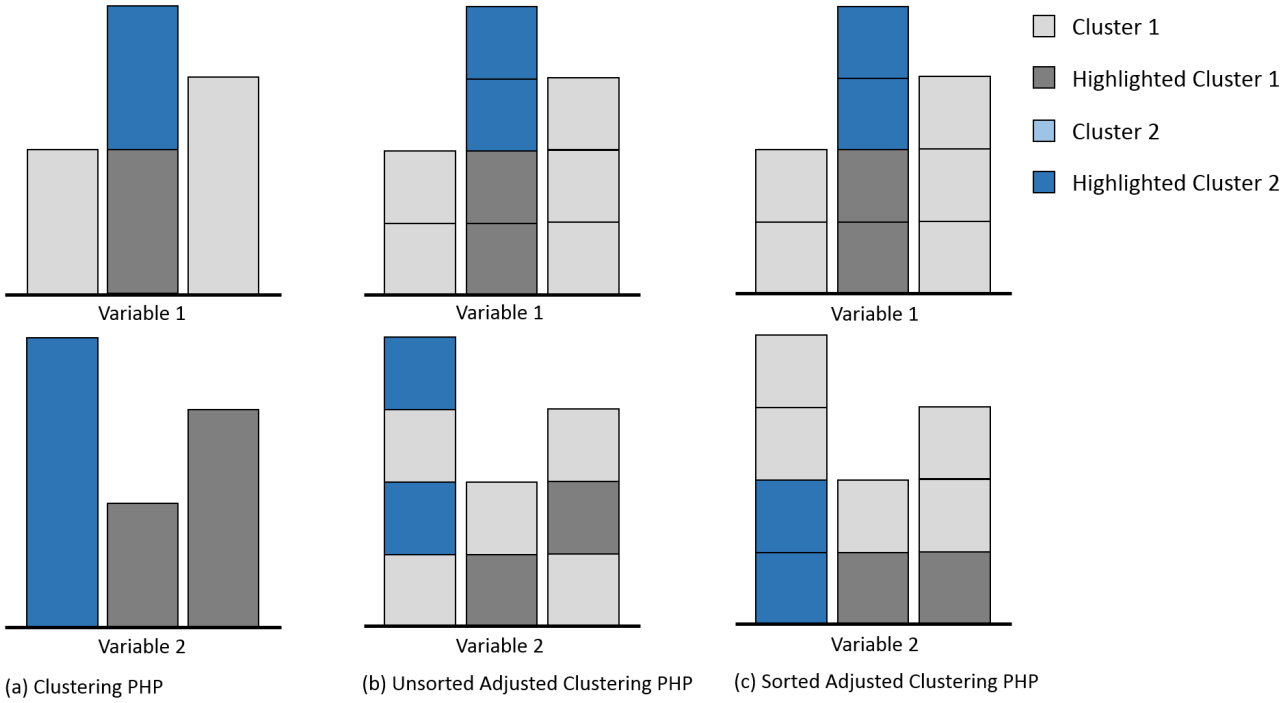


Figure 4.20: A sample example with two variables to explain the 4.3.c solution easily. Each variable takes one dimension. (a)The full stacked bar highlights the other dimension's bars wrongly since it counts all bars which contain the selected instances. (b) Unsorted clustering PHP. It is from splitting the full clustering PHP with the same color scheme but colors are mixed up. (c) Sorted clustering PHP is implemented in this thesis. highlighting features are on the target instances and patterns are easily detected with the organized plot.

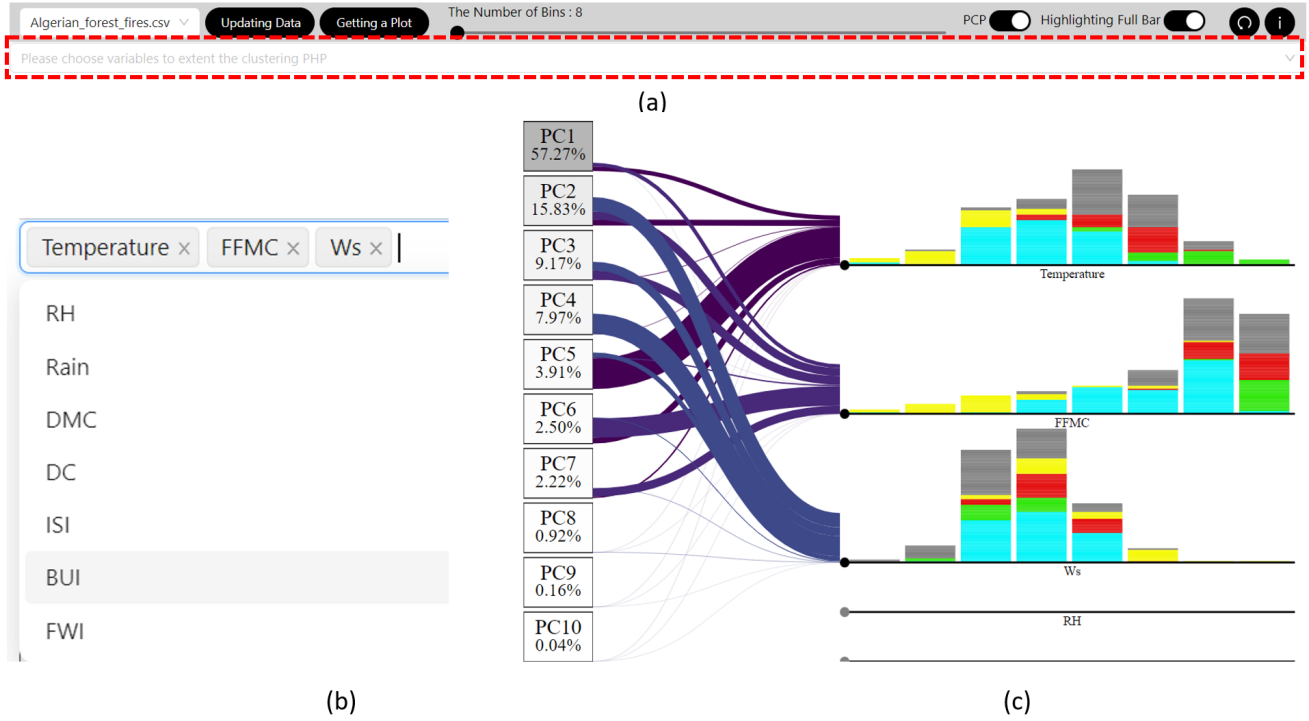


Figure 4.21: (a)A function to extend clustering PHP. (b)After clicking the red part on (a), variable selections will be offered. (c) A result of selecting the variable's name. Clustering PHP is drawn from the top in the existing clustering PHP, so the desired variables can be observed closely adjacently.

efficiently. When a single stacked rectangle's height is too small in clustering PHP, it leads errors. Therefore, here is a function to give an alert with the height under 0.02 in figure 4.22. After closing the alert message and putting up the bin's number, a user gets a correct result in the application.

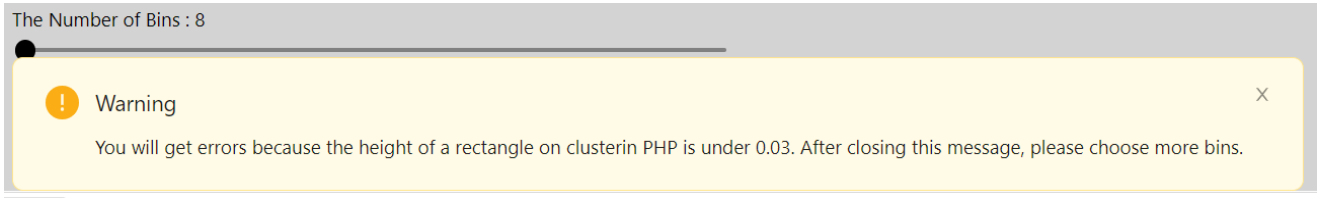


Figure 4.22: A warning alert pops up when a single stacked rectangle's height is under 0.02 in clustering PHP. It guides a user to adjust a right size of bins without errors.

d. Building a PCP to visualize multidimensional data.

Parallel coordinates plot (PCP) is mapped on clustering PHP. When a histogram bar is clicked, related IDs are searched and connected among other dimensions. Frequency can be presented as transparency. High frequency means that many instances move to the same bin between two adjacent variables. The high frequency are represented with a high opacity. In contrast, when the line's opacity is weak, it indicates the frequency is low.

In figure 4.23, the red box points two switches which affect PCP and clustering PHP. One is the 'PCP' switch and the other one is the 'Highlighting Full Bar' switch. In figure 4.23, both switches are on and a user clicks a histogram bar on 'Temperature'. Consequently, the full stacked bar on 'Temperature' and related instances on the other variables are highlighted. PCP is also built on the clustering PHP. A user can detect variables' relation-

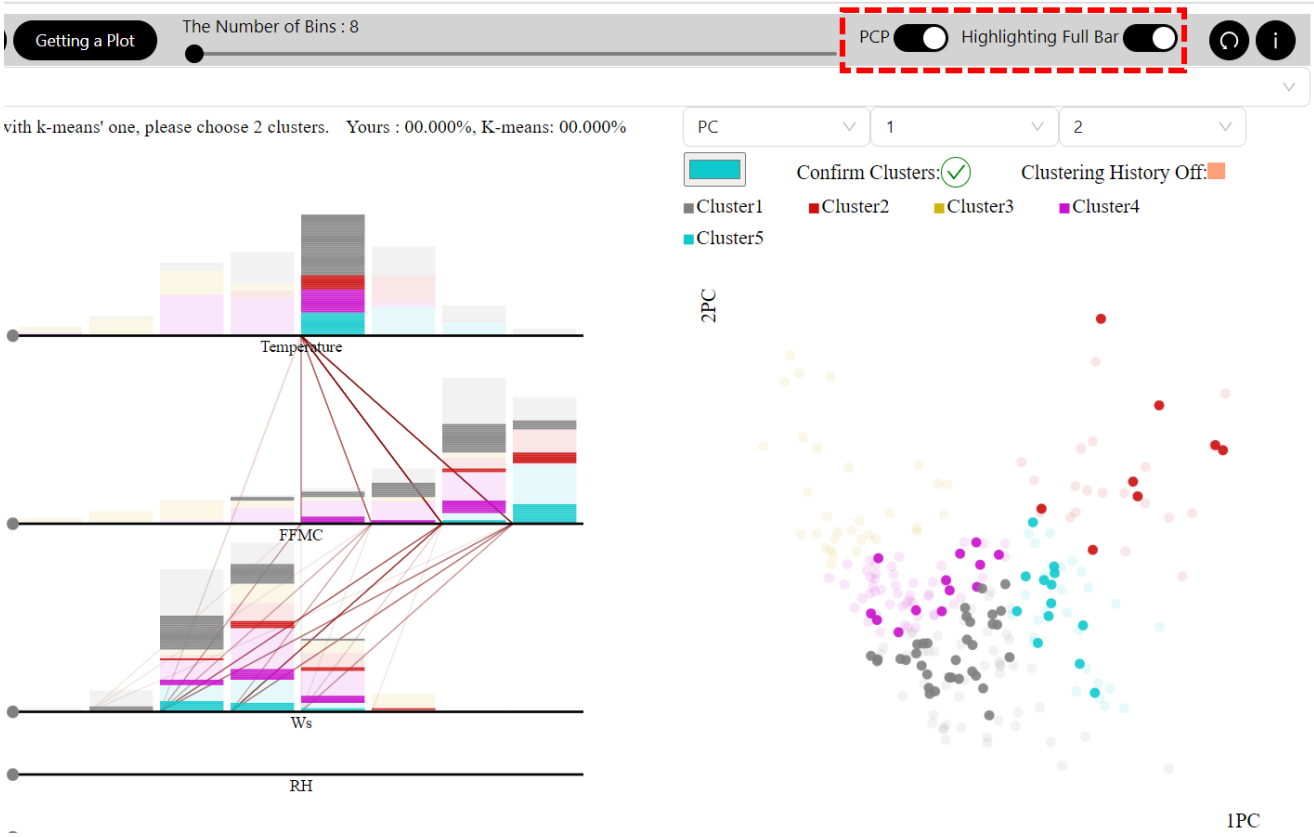


Figure 4.23: The 'PCP' switch is on and the 'Highlighting Full Bar' switch is on.

Figure 4.24: The 'PCP' switch is on and the 'Highlighting Full Bar' switch is off.

ship with a full view of all items of the selected dimension into the other dimensions. The related data is also highlighted on the scatter plot. When the 'Highlighting Full Bar' switch is turned off and 'PCP' is on, the full histogram bar is divided into small bars by color in figure 4.24 to look closely.

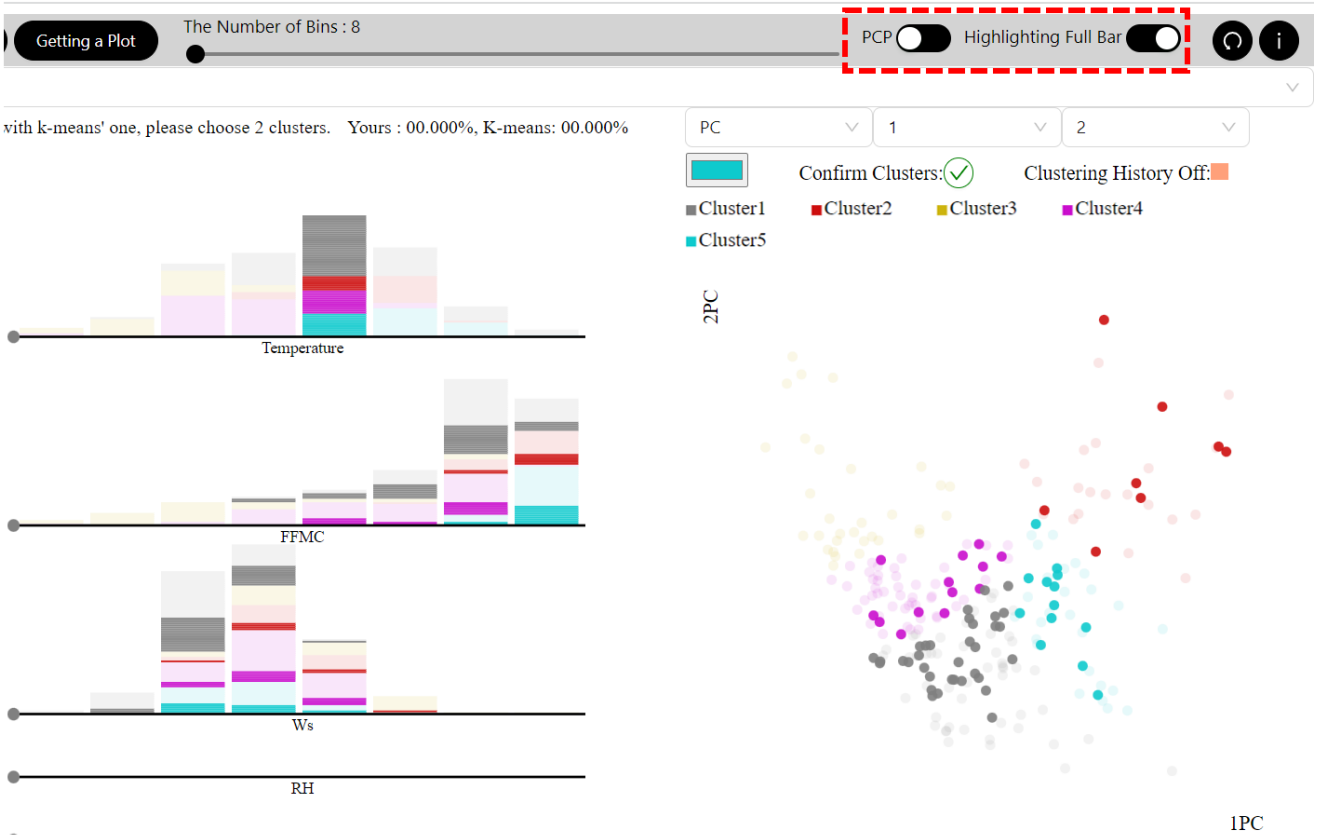


Figure 4.25: The 'PCP' switch is off and the 'Highlighting Full Bar' switch is on.

If many lines of PCP obstruct observation, a user can swap the 'PCP' switch to be off in figure 4.25 and figure 4.26. Highlighting parts of figure 4.25 are the same as figure 4.23 and figure 4.26's one is the same from figure 4.24's one. It's the same spotlights and color scheme, but a user can focus neatly only on clustering PHP without distraction from PCP. As turning off PCP, the application has a lighter load than the turning on because of not making data of PCA and not rendering lines. As a result, the application can plot only clustering PHP quickly and assist a user to detect variables' relevancy and tendency neatly.

Figure 4.26: The 'PCP' switch is off and the 'Highlighting Full Bar' switch is on.

4.3 Accuracy

Basically, as this thesis introduces an interactive application, it does not limit the number of classes for any datasets. A user can explore data and assign any number of clusters as much as the user wants. However, for calculating accuracy, ground truth is needed. Therefore, the application offers a user how many clusters are required based on the ground truth. Consequently, once a user assigns the same number of clusters as much as the number of clusters in the ground truth, the application calculates accuracy from backend. However, before the calculation, certain order of buttons should be clicked. First, the green check mark next to the text 'Confirm clusters' should be clicked, and then the 'Updating Data' button and the 'Getting a Plot' button should be clicked sequentially. Finally, the results are in percent rounded to three decimal places in figure 4.32. 'Yours' means the user's interaction-based clustering algorithm and 'K-means' takes a K-means algorithm. K-means algorithm is chosen for the comparison since K-means method is one of the popular clustering methods due to its simple and fast technique[KM+13].

There are plenty of ways to explore the application and set clusters but here is a tip to get better accuracy with this application than K-means' accuracy. It can be also a guide to choose how to find interesting variables and to get useful subspaces. A user can follow steps from figure 4.27 to figure 4.31. This guide is with data from the section 6.3 Application Scenario1: Algerian forest fires dataset. To begin with, it is better to start from a scatter plot as figure 4.27 with the first and second principal components as Step1. When some possible clusters are detected, they can be marked as clusters with colors in Step2.

In Step3, PCA lines for distribution of variables to a principal component can give a meaningful hint to find useful variables. The rectangles with principal components can be clicked from the first principal component to

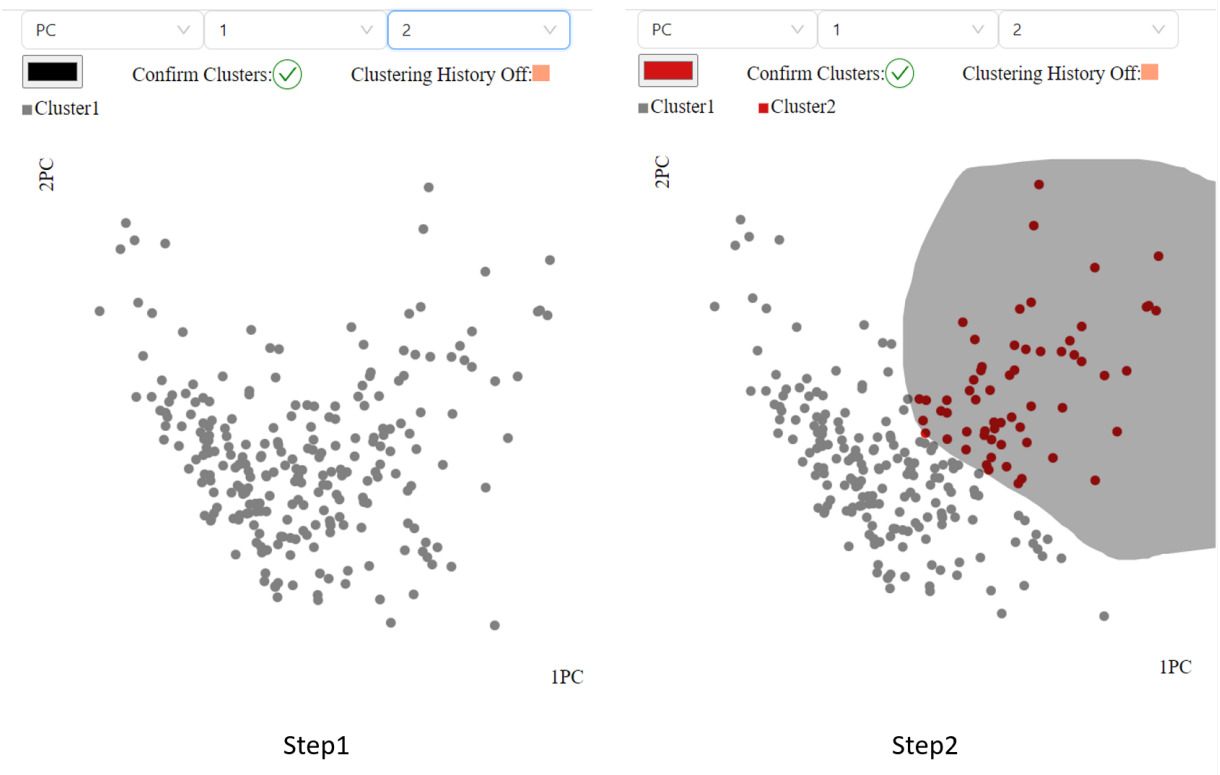


Figure 4.27: It is a tip to get better accuracy and find relevancy among multi-variables. The tip is explained in steps. Step1: starting from the scatter plot with the first and second principal components. Step2: finding clusters and coloring them.

the following components till the sum is approximately 70 percents. In this case, PC1 and PC2 are selected. The sum of purplish lines' thickness in figure 4.28 indicates possible important variables. When the thickness is big, it shows the variable might be an important clustering key among data. Afterward, a user can click rectangles of the last principal component and the second last principal component. The yellowish lines can get rid of variables because variables with the thick yellowish lines might be redundant or make a noise for clustering. However, if the yellowish lines are overlapped on purplish lines a lot, it might be better to ignore the yellowish lines and a user should explore variables on purplish lines more. To consider these lines, 'Temperature', 'FFMC', 'Rain', 'RH', and 'Ws' are picked up and the related clustering PHPs are built up with the color scheme from Step2 in figure 4.27. To dive into variables, a scatter plot can be utilized in Step4. A user can explore the scatter plot with various combinations of the variables and find new clusters.

In figure 4.29, clustering PHP is expanded with 'Temperature', 'FFMC', and 'Rain'. A negative correlation is observed between 'FFMC' and 'Rain' on the clustering PHP and lines in Step5. Therefore, it might be interesting to look into 'FFMC' and 'Rain' more than the other variables, so a scatter plot is created with them. There is a mixed part with grey and red points which means clusters are mixed up. To find a right cluster for the mixture, a user can click histogram bars by putting the 'PCP' and 'Highlighting Full Bar' switches on. The related data points are highlighted on clustering PHP and the scatter plot, so the user can see how variables are related and decide more cluster groups delicately.

Even the application offers more deep inside of view in Step6 in figure 4.30. As swapping the 'Highlighting Full Bar' switch off, the full histogram bar is divided into small bars by color. If a user selects a grey part on 'FFMC', related data has a spotlight on 'Temperature' and 'Rain'. PCP lines are made accordingly as well. Based

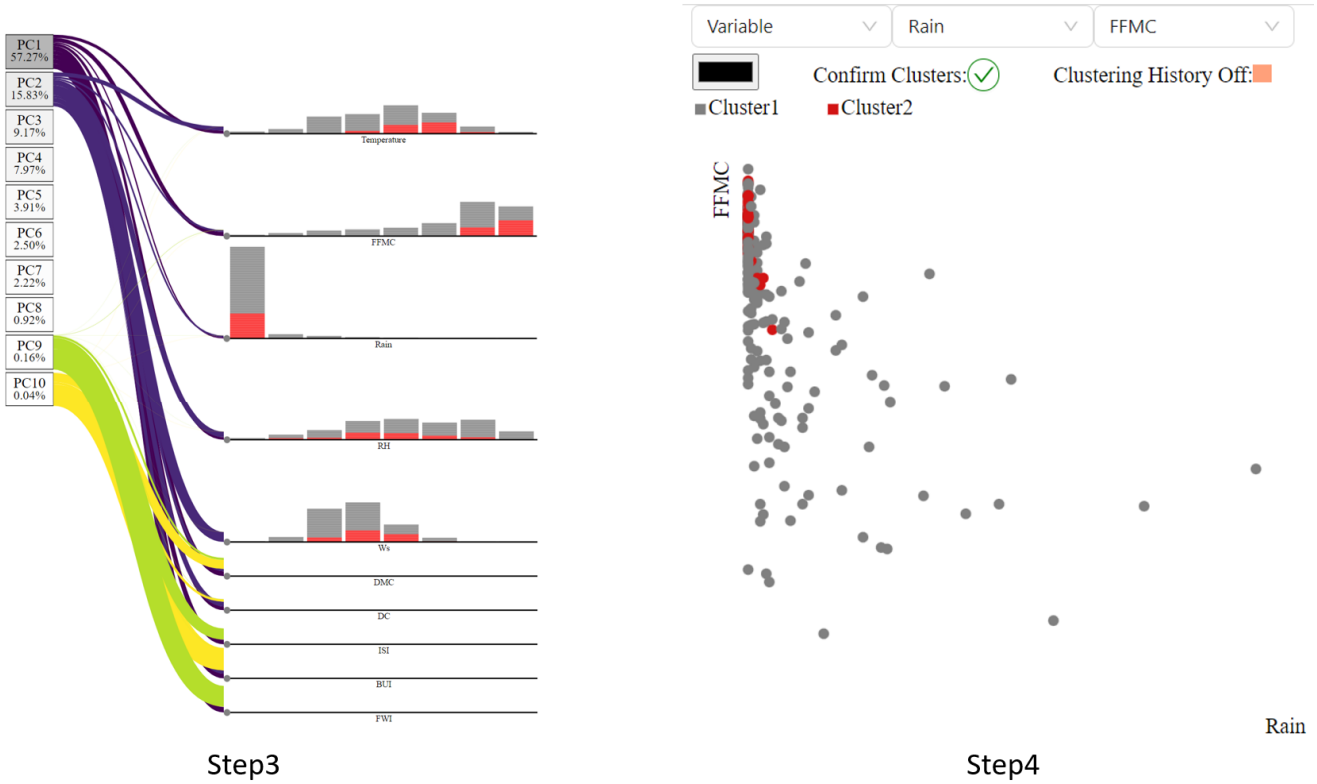


Figure 4.28: Step3: PCA lines from principal components to variables can be used to find meaningful variables. When the sum of purplish lines' thickness is big, the one can be a possible important variable for clustering. The yellowish lines can get rid of variables because they might be redundant or make a noise for clustering. However, if the yellowish lines are overlapped on purplish lines a lot, it might be better to turn off the yellowish lines and a user should explore variables on purplish lines more. With consideration of lines, variables can be chosen and their clustering PHPs are drawn. Step4: exploring the scatter plot with combinations of the variables and detecting new clusters.

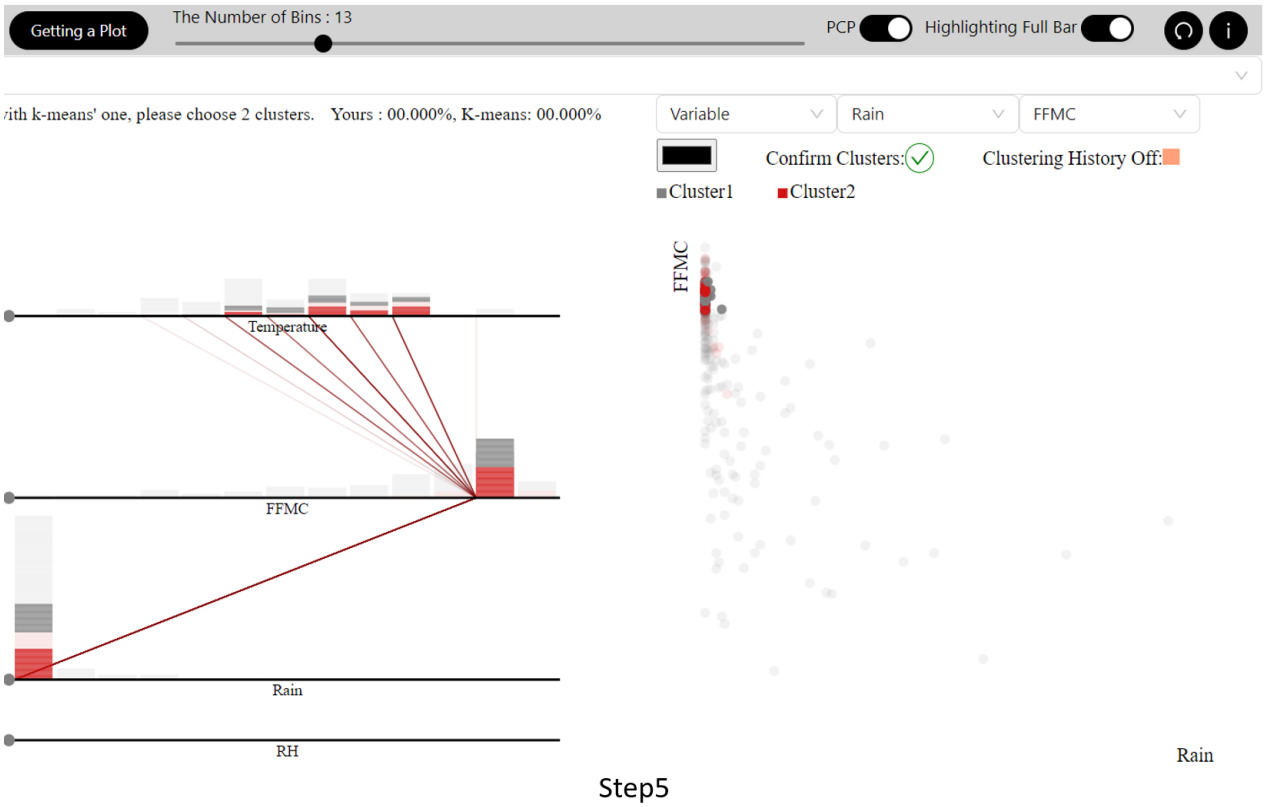


Figure 4.29: As clicking a highlighting function with the full bar, negative correlation is detected between 'FFMC' and 'Rain' to each other. Mixed points can be correctly assigned into right clusters after exploring the clustering PHP and the scatter plot.

on the observation, the selected grey part might be in the red group(Cluster2). Before changing the cluster, it is better to have a look more to confirm the new decision. The suspicious part can be investigated deeply on Step7 choosing the other histograms. A user also can check dubious points' clustering history and make a decision which cluster group the points should belong to. As a result, the suspicious part becomes a red group(Cluster2).

Figure 4.30: A detailed view by turning the 'Highlighting Full Bar' switch off. The full histogram bar is divided into small bars by color, so a user can focus on the specific part.

After the number of cluster groups reaches to the same number of the ground truth's cluster groups, a user can request to get accuracy. In this case, they are two clusters. To get the accuracy, a user clicks the green mark next to the 'Confirm clusters' text, then the 'Updating Data' button and the 'Getting a Plot' button. Eventually, the accuracy results are out in figure 4.32. The K-means' accuracy is 62.964% and the user interaction one is 88.889%. Therefore, the application improves more accurate clustering results with the visual design.

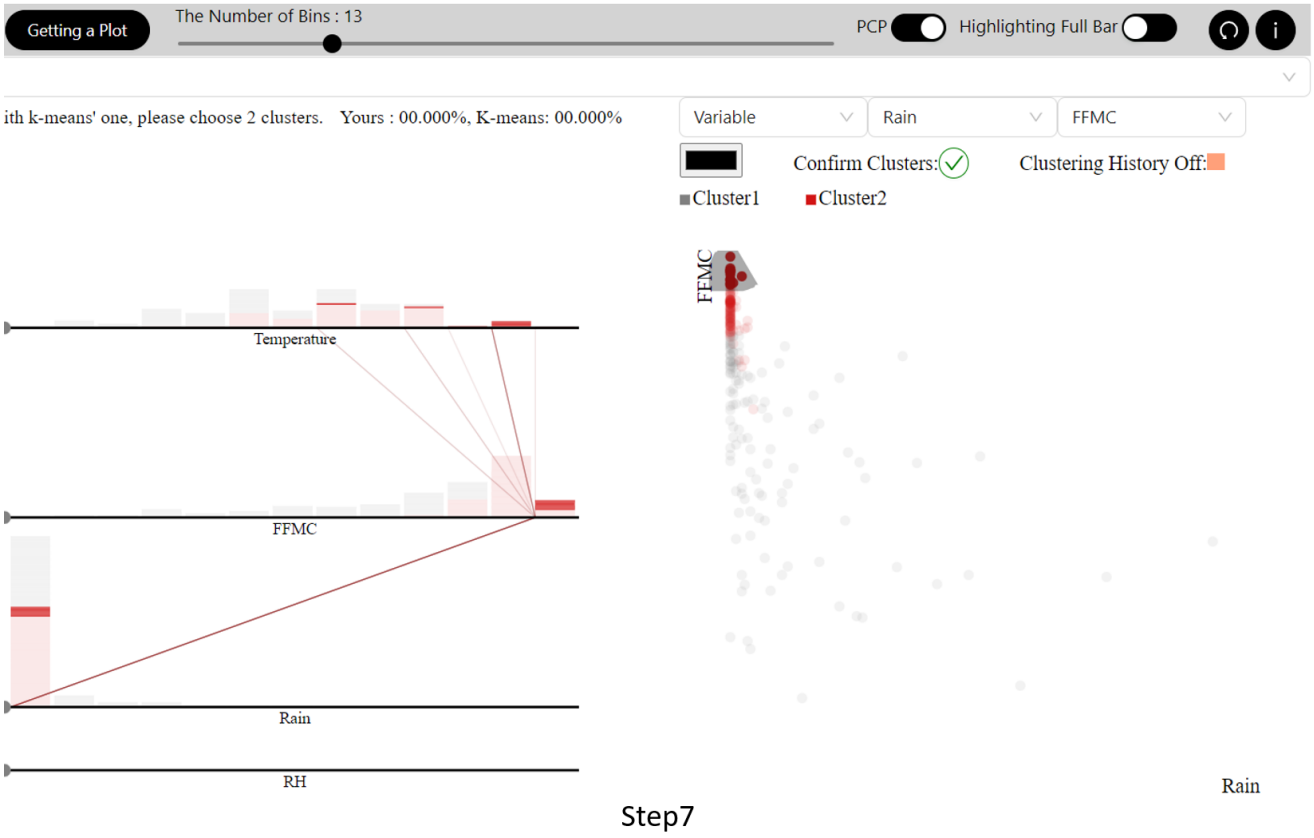


Figure 4.31: Suspicious parts can be investigated more by choosing the other histograms. In this case, the suspicious part is decided to belong to the red group.

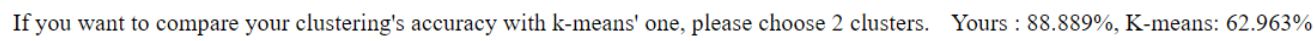


Figure 4.32: After the number of cluster groups reaches two, the green check mark next to the 'Confirm clusters' text, then the 'Updating Data' button and the 'Getting a Plot' button should be clicked to get accuracy. The application improves accurate clustering results.

5 Implementation

5.1 Overview

Figure 5.1 explains main pipelines of the clustering PHP application. There are two main part of the application such as backend and frontend. The purpose of this application is helping a user to conduct a more accurate clustering result with a visual explorative framework. To magnify visual exploration, using user interaction on frontend can be a clever idea. However, it might bring bad performance, if the application conducts all works such as performing data processing, executing machine learning algorithms, and reacting user interaction. Speedy reaction is crucial to this application because a user explore all connected plots and visually detect clusters from meaningful subspaces. To booster the speed, this application separates work loads into backend and frontend. Therefore, time consuming tasks are executed on backend instead of frontend except for necessary parts such as visualization and events from a user.

Backend with Python is in charge of parts which do not need to react user interaction immediately. For example, data processing and running machine learning methods are not needed right after user interaction, so the tasks are done in backend. For data processing, a ID is assigned on each instance to search and locate efficiently. Results from backend can be sent to frontend via Flask in backend, and received by Axios in frontend. Huge loads of Communication between backend and frontend can be a reason to make the performance slow, so this application minimizes unnecessary deliveries. For example, the full data is sent only once when the 'Get' Button clicked instead of sending the full data with all events. Plus, the result of clustering is posted from frontend to backend with only one dimensional array instead of the full data with the clustering result. For example, when the full array has A instances and B variables, sending the full array with the result takes A*(B+1) size but sending only clustering result is A*1 size which causes smaller loads.

As mentioned on the section 4.3.c, sorting is a key function of this application, but sorting is slow with the frontend development. Therefore, putting sorting part on backend was considered in this application. However, the consideration is not proper because communications between backend and frontend need a trigger, so user have to click a button every single time after events. The trigger also makes re-rendering which produces bad performance. Therefore, the application keeps the sorting part on frontend but the best performance sorting is implemented from many ways in the section 6.1 Sorting Speed. As shown on the frontend side in figure 5.1, parent component and child components are divided to prevent unnecessary re-rendering. Events control only the specific part to be rendered again or minimize loads of sending information between functions. This application is optimized on Chrome with the window's height 661 pixels and width 828 pixels. If the application does not fit in a user's screen, a user can scroll up or down with a mouse to adjust the screen size.

5.2 Dependency

An overview of dependency is demonstrated in figure 5.2. On backend, Python3 is implemented and it takes care of data processing, assigning labels on clustering groups, and calculating accuracy. For libraries in Python, pandas and numpy are used for data process. Scikit-learn is also employed for performing PCA, assigning labels on clustering groups, and calculating accuracy. scipy.optimize is also applied together for assigning labels on clustering groups. Plus, make_classification from scikit-learn generates a sample dataset for the section 6.Experimental Results. Flask in backend and Axios in frontend are utilized to communicate with backend and frontend.

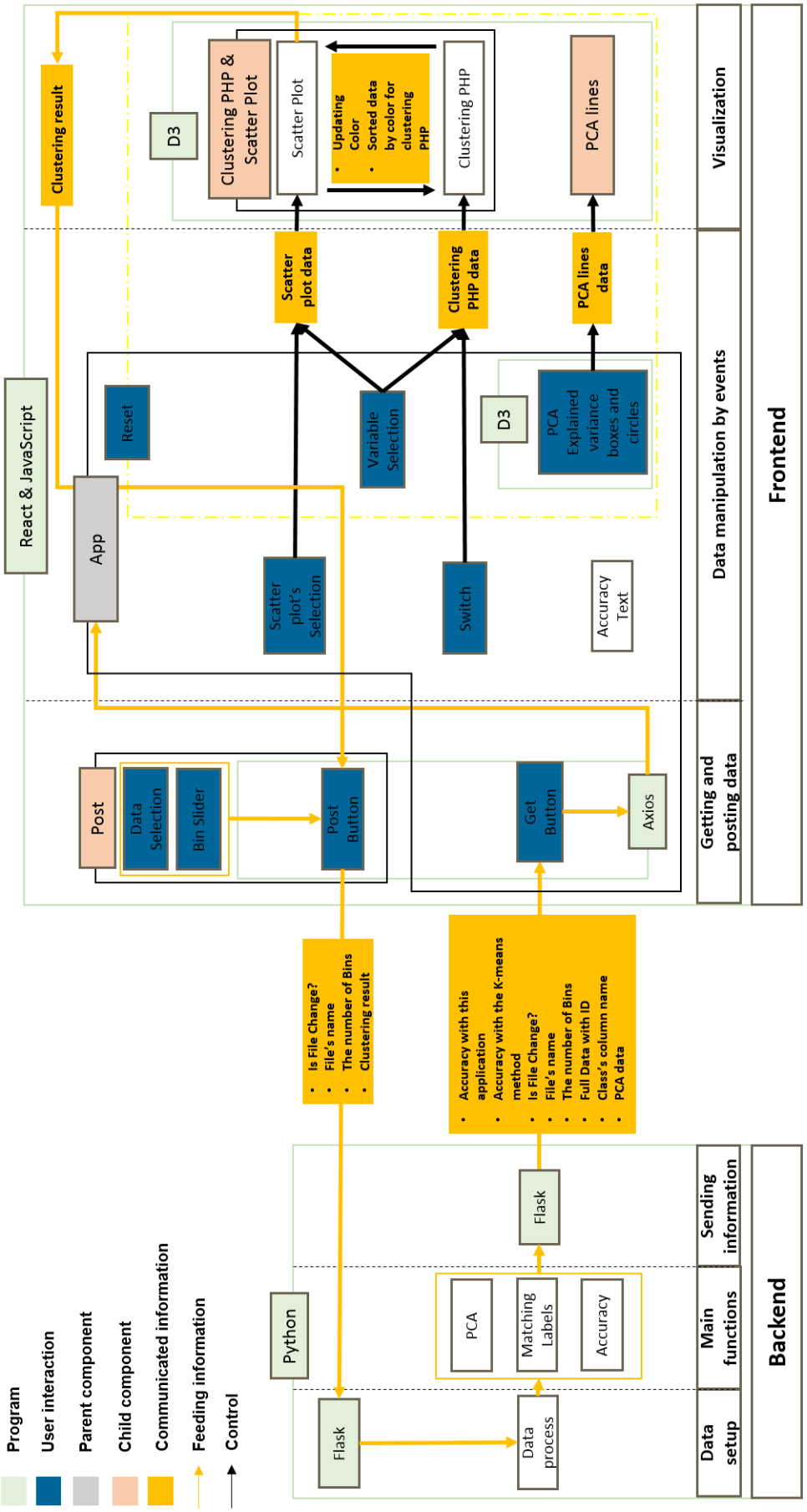


Figure 5.1: Overview of implementation

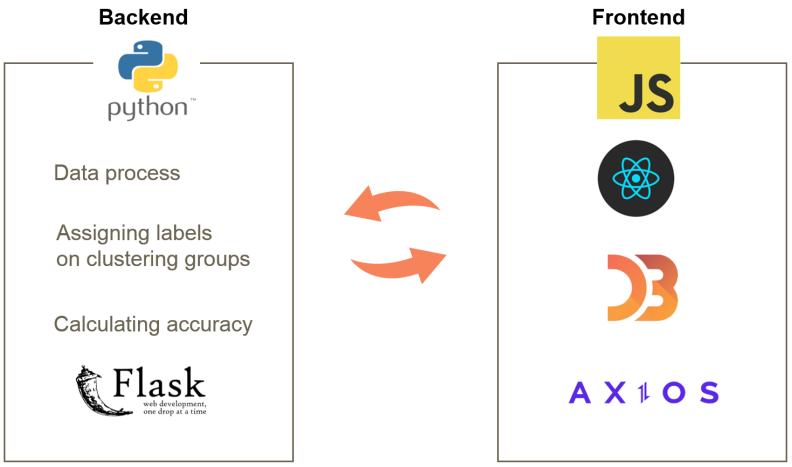


Figure 5.2: Dependency on backend and frontend

They allow the application to post and set data smoothly. In frontend, JavaScript is used with React, D3 and Axios. To be specific, React builds frontend and plots are created by D3.

5.3 Code

This section explains important parts in depth with code. In backend, it is critical to assign correct labels on the right clusters and to calculate accuracy. In frontend, adjusting data for the clustering PHP is important.

Backend

This application uses accuracy to evaluate algorithms' performance. Before calculating accuracy, it is essential to find the best match between labels from a user and ground truth labels. Therefore, listing 5.1 can optimally match labels based on the linear sum assignment and returns accuracy. This code refers to Ouali et al.'s code [OHT20].

Listing 5.1: Finding correct labels on clusters in backend with Python

```
def find_indices(list_to_check, item_to_find):
    indices = []
    for idx, value in enumerate(list_to_check):
        if value == item_to_find:
            indices.append(idx)
    return indices
def calculateAccuracy(y,y_pred):
    y_adj=y_pred.copy()
    #Optimized matching labels based on linear sum assignment.
    confusion = confusion_matrix(y,y_pred)
    old_ind, new_ind = linear_sum_assignment(np.max(confusion)-confusion)
    new_ind.tolist()
    old_ind.tolist()

   #assigning correct labels on clusters.
    adj_list=[]
    for i in range(len(old_ind)):
        adj_ind=find_indices(y_adj.tolist(),new_ind[i])
        adj_list.append([adj_ind,old_ind[i]])
    for i in range(len(adj_list)):
        y_adj[adj_list[i][0]]=adj_list[i][1]
   #Calculating accuracy
    return round(accuracy_score(y.tolist(), y_adj.tolist())*100,3)
```

Frontend

This function in listing 5.2 generates a new data. The data takes only necessary information such as ID and the location of bin and variable. The function is in a parent component, so when the data is updated, the result goes to related child components instead of entire child components. The function runs after changing a bin's size and updating variables on clustering PHP.

Listing 5.2: Making data for clustering PHP

```
function BinMaker(n,dimN,originaldata,name,DimList,BinNum1) {
    const fulldata=originaldata.map((x) => x)
    let dimNameInd=name.indexOf(dimN)
    let FullD=fulldata.map((e) => { return e[dimNameInd] })
    const FullMax=Math.max(...FullD)
    const FullMin=Math.min(...FullD)
    //Making a range according to the bin size
    let binRange=[]
    for (let ii=0;ii<BinNum1;ii++){
      binRange.push(FullMin + ((ii)/BinNum1 ) * (FullMax - FullMin))
    }
    //Making clustering PHP data : d[0]=ID, d[1]=Dim Number, d[2]=State(Bin Number)
    let IndexID=name.indexOf('ID')
    let filteredData=[]
    for (var i=0; i<fulldata.length;i++){
      for (var j=0;j<binRange.length-1;j++){
        if (fulldata[i][dimNameInd]>=binRange[j] && fulldata[i][dimNameInd]<binRange[j
            +1]){
          filteredData.push([fulldata[i][IndexID], n, j]);
        }
      }
      if (fulldata[i][dimNameInd]>=binRange[binRange.length-1]){
        filteredData.push([fulldata[i][IndexID], n, binRange.length-1]);
      }
    }

    //sorting data : dimensional => bin number
    filteredData.sort((a, b)=> {
      if (a[1] === b[1]){
        if (a[2]===b[2]){ return clusteringList[a[0]]<clusteringList[b[0]]? -1:1}
        else{ return a[2] < b[2] ? -1 : 1}

      } else {
        return a[1] < b[1] ? -1 : 1
      }
    })
return filteredData
}
```

5.4 PCA lines : contribution

The following explanation is according to the papers [TLJ+19] and [AW10].

Let data $\mathbf{D} \in \mathbb{R}^{n \times k}$ be. The data has n instances and k variables. $\mathbf{M}$ is the mean of $\mathbf{D}$ in a vector-wise direction with $\mathbf{M} \in \mathbb{R}^{n \times k}$. The result is $\mathbf{X} \in \mathbb{R}^{n \times k}$, and the covariance $\mathbf{A} \in \mathbb{R}^{k \times k}$.

$$\mathbf{X} = \mathbf{D} - \mathbf{M},$$
$$\mathbf{A} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X},$$

Eigendecomposition of $\mathbf{A}$ is executed.

$$\mathbf{A} = \mathbf{W}\Lambda\mathbf{W}^T$$

$\Lambda \in \mathbb{R}^{k\times k}$ is a diagonal matrix with eigenvalues λ_i, i = 1,2,...,k. $\mathbf{W} \in \mathbb{R}^{k\times k}$ is a set of eigenvectors with $\boldsymbol{w}_i$ with i = 1,2,...,k corresponding eigenvalues λ_i.
Then, $\Sigma = \sqrt{\Lambda}$ by $\sigma_i = \sqrt{\lambda_i}$ with i = 1,2,...,k and $\Sigma \in \mathbb{R}^{k\times k}$. Loading as a factor score is $\mathbf{F} \in \mathbb{R}^{k\times k}$.

$$\mathbf{F} = \mathbf{W}\Sigma$$

The squared cosine(cos^2) indicates the importance of a component for a given observation. It squares the distance of the observation to the origin. Let n denote the number of principal components and m denote the number of variables with n= 1,2,...,k and m= 1,2,...,k. $\boldsymbol{f}$ indicates an element of $\mathbf{F}$.
Therefore, contribution of n variables to the principal component m is the following:

$${cos_n}^2 = \frac{\boldsymbol{f}_{nm}^2}{\sum_{m'=1}^{k}\boldsymbol{f}_{nm'}^2}$$

In the clustering PHP application, contribution of m principal components to the variable n is the transpose of contribution of n variables to the principal component m. A large value of ${cos_n}^2$ indicates that it contributes a relatively big portion to the total distance. In other words, when the result is high, variables contribute highly to a principal component.

$${cos_m}^2 = \left({cos_n}^2\right)^T = \frac{\boldsymbol{f}_{mn}^2}{\sum_{m'=1}^{k}\boldsymbol{f}_{m'n}^2}$$

${cos_m}^2$ represents the contribution of components to a variable. Is is a transpose of ${cos_n}^2$. A big value of ${cos_m}^2$ means that the component is important for the variable. The sum of ${cos_n}^2$ and ${cos_m}^2$ across each component or each variable equals to one.

6 Experimental Results

6.1 Sorting Speed

As mentioned in the section 4.3.c, sorting data is crucial for user interaction. However, sorting data with React and JavaScript can be not fast and efficient as much as the other backend development because the frontend web development focuses on user interface. However, sorting data has to be implemented in fronted in the clustering PHP application for dynamic user interaction. Therefore, making the sorting process faster can be a main key for better performance in the clustering PHP application. This thesis suggests five ways to improve speed of the application. Data is sorted by selected IDs. After testing their performances, the best one will be applied in this application. To begin with, there are functions from JavaScript. 'sort()' means that it converts elements as string, and then compares UTF-16 code of the elements to sort. 'Array.from()' is a static property of Array object. 'filter()' runs all dataset and filter instances by condition. 'slice()' takes elements by index numbers.

- way1: executing 'sort()' on the full two dimensional data.
- way2: with 'filter()', dividing the the full two dimensional data by selected IDs into data to be sorted and data not to be sorted. On data to be sorted, filtering bins included selected ID with 'filter()'. Sorting the bins which have selected IDs in each dimension with 'sort()'. Combining the sorted data from all dimensions.
- way3: applying 'Array.from()', and then 'sort()' on the full two dimensional array.
- way4: it is the same as way2 but it takes 'Array.from()' before 'sort()'
- way5: with 'filter()', dividing the the full two dimensional data by selected IDs into data to be sorted and data not to be sorted. On data to be sorted, filtering bins included selected IDs with 'slice()'. Applying 'Array.from()' and sorting the bins which have selected IDs in each dimension with 'sort()'. Combining the sorted data from all dimensions.

Figure 6.1: Left : speed of data size 500 by the clustering PHP data size, Right : speed of data size 5000 by the clustering PHP data size

This thesis evaluates speed of sorting with Windows 8.1 and Google Chrome. Sample datasets are generated by make_classification from scikit-learn. All generated datasets' random states are zero for a fair comparison to

Figure 6.2: Left : speed of data size 10000 by the clustering PHP data size, Right : accuracy with increasing the number of variables

different conditions and 'class_sep' is three to spread out clusters slightly. The initial class is cluster1 in grey. Clustering PHP data size increases by adding variables from variable1 to the last variable. When a user clicks a histogram bar in a dimension, data on the other dimensions are sorted, so time is checked from twice of the original data size. On each variable, a user clicks the highest histogram bar. The time is the average of all variables' time.

Figure 6.1 and figure 6.2 Left are comparisons on different data sizes by specific conditions. They have total eight variables, four useful variables which mean the half of variables makes meaningful clustering features, and two classes. The size of data is different such as 500, 5000, and 10000. Three plots seem a similar tendency with the ways. Even though way2 and way4 have an advantage not to sort the full data and sort only data to be sorted, way2 and way4 take longer time than others. It is because they have loops to find bins and combine selected dimension's data and the rest data. Plus, way5 is faster than way2 and way4 because way5 takes only needed part instead of the full data. To be specific, way2 and way4 are built with 'filter()' to split data. It executes all data to fulfill conditions. In contrast, way5 uses 'slice()' to split data. It does not run through all data to filter the data and changes data by index directly. As a result, way5 is quicker than way2 and way4. Furthermore, way3 and way1 are quicker than others since 'sort()' optimizes a sorting process. Even if they take the full data, they consume less time than dividing data with loops or filters. The difference between way3 and way1 is that way1 converts an integer array to a string array inside of 'sort()' and way3 converts an integer array to a string array before 'sort()'.

To check speed of multivariate data, the number of variables increases in the right side of figure 6.2. The dataset size is 5000 and useful variables are the half of variables. All ways run slower with 20 variables' dataset than others but they become relatively faster with 30variables' dataset. way1 and way3 are the most fastest with all conditions. The result is similar to the previous plots' results. Therefore, way1 and way 3 can be the most efficient options to sort data on this application. In this thesis, way1 is applied on the clustering PHP application because it performs better with big dataset and multi-dimensional data on figure 6.2(a) and (b).

6.2 Accuracy

The goal of the application is maximizing advantages of clustering PHP and build a visual explorative framework to conduct subspace clustering tasks with an interactive user interface. Furthermore, the application can offer better accuracy than K-means' one. Based on Swets' paper [Swe88], accuracy can be calculated as the equation 6.1.

$$Accuracy = \frac{True\ positive + True\ negative}{True\ positive + False\ positive + True\ negative + False\ negative} \tag{6.1}$$

The application's accuracy comes from clusters decided by user interaction. Therefore, the accuracy can be affected and changed by what a user makes a decision while exploring the application. To get a fair result, the accuracy is the average of five times attempts' accuracy. Datasets are made by make_classification from scikit-learn. The generated datasets are with the following conditions. The number of useful variables is half of the number of total variables. The 'random_state' is zero for a fair comparison and 'class_sep' is 3. 'Yours' denotes the average of five attempts' accuracy with the thesis's application and 'K-means' indicates the average of five attempts' accuracy with the K-means method.

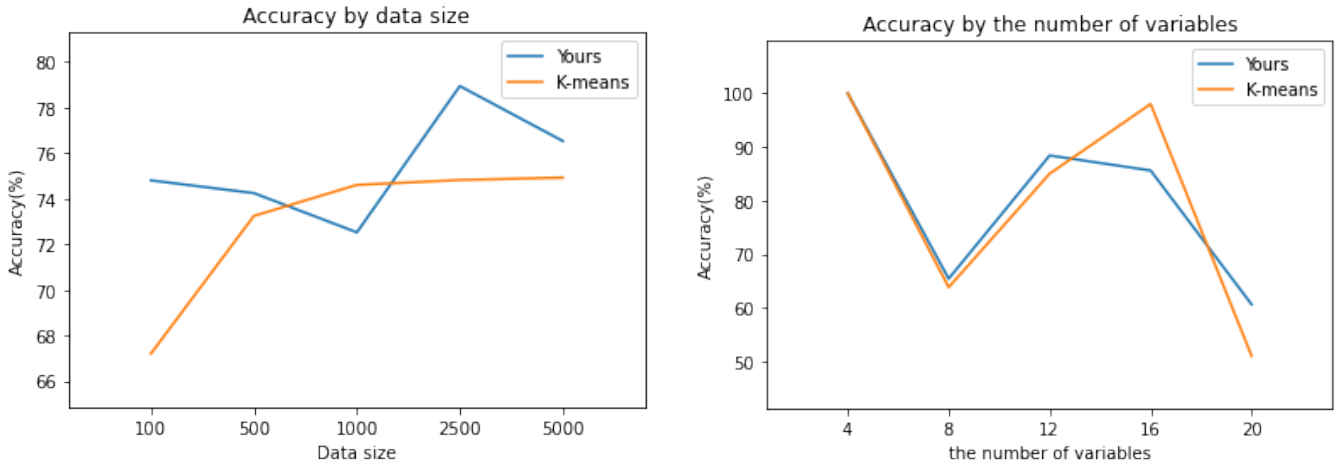


Figure 6.3: Left : accuracy by data size, Right : accuracy by the number of variables

A dataset in figure 6.3 Left has total eight variables, and two classes. Its data size is from 100 to 5000. Both algorithms are likely to make better results when the size of data becomes bigger. In the many cases, using the application('Yours') makes accurate results than K-means. in figure 6.3 Right, the data has 500 data size and two classes. The number of variables changes from four to 20. 'Yours' and 'K-means' return similar accuracy but 'K-means' performs very bad on twenty variables.

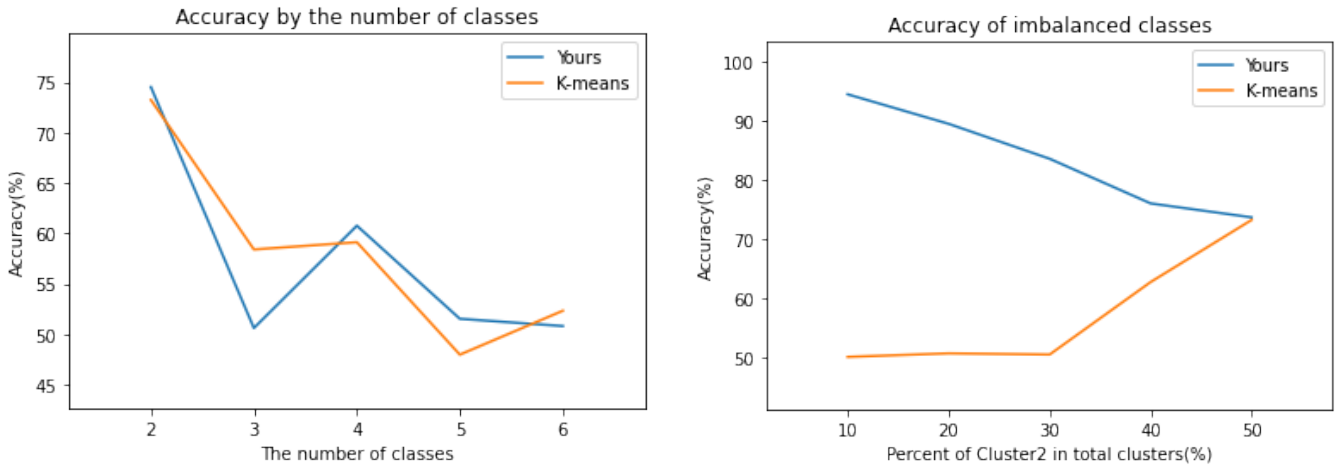


Figure 6.4: Left : accuracy by the number of classes, Right : accuracy of imbalanced classes

A dataset in figure 6.4 Right has 500 data size, and the total eight variables. The number of classes is ranged from two to six. When the number of classes increases, the accuracy of both algorithms performs very poorly

around 50 percent. Plus, a dataset in figure 6.4 Right has 500 data size, and eight variables in total. There are two classes but with different weights from 10 percent to 50 percent. 10 percent means Cluster1 is 90percent and Cluster2 is 10percent. 'Yours' operates much better when the data set is not balanced. To conclude, using an interactive user interface in the application can conduct accurate results especially with big datasets and imbalanced datasets.

6.3 Application Scenario1: Algerian forest fires dataset

Variable	Detail
Temp	Temperature at noon (temperature max) in Celsius degrees from 22 to 42
RH	Relative Humidity from 21% to 90%
Ws	Wind speed from 6 km/h to 29km/h
Rain	Total day from 0 to 16.8mm
FFMC	Fine Fuel Moisture Code (FFMC) index from the FWI system from 28.6 to 92.5
DMC	Duff Moisture Code (DMC) index from the FWI system from 1.1 to 65.9
DC	Drought Code (DC) index from the FWI system from 7 to 220.4
ISI	Initial Spread Index (ISI) index from the FWI system from 0 to 18.5
BUI	Buildup Index (BUI) index from the FWI system from 1.1 to 68
FWI	Fire Weather Index (FWI) Index from 0 to 31.1
Class	1: Fire, 0: Not Fire

Figure 6.5: Attribute information of Algerian forest fires dataset.

Here is a real data set with 244 instances and 12 Attributes from the UC Irvine Machine Learning Repository. It is a classification dataset with classes such as 'Fire' and 'Not Fire' in the Algeria region. The original data is adjusted by deleting the 'Date' column, dropping an invalid row and labeling 'Fire' as 1 and 'Not Fire' as 0. The result is in figure 6.5 with 243 instances and 11 Attributes. Plus, the number of class 0 (Not Fire) is 106 and 1(Fire) is 137, so this data set is pretty balanced. This scenario is explained in the section 4.4.Accuracy with steps. After following the steps with the clustering PHP application, it returns a better cluster result(88.889%) than k-means(62.963%) in figure 4.32.

6.4 Application Scenario2: Breast Cancer Wisconsin dataset

To evaluate how to deal with multivariate data, this scenario takes more variables than the scenario1. The dataset is Breast Cancer Wisconsin (Diagnostic) from the UC Irvine Machine Learning Repository. This classification dataset has 569 instances and 30 Attributes for diagnosis of breast cancer in three planes. There are two classes 'Malignant' and 'Benign'. In this scenario, the dataset is edited such as dropping Patient ID and changing labels of class. 'Malignant' becomes 1(class1) and 'Benign' is 0(class0). The number of class1 is 357 and class0 is 212, so it is an imbalanced dataset. Because variables are from a digitized image of a fine needle aspirate(FNA) of a breast mass, variables are in each separating plane except for the 'Diagnosis' variable. For example, radius1 is a radius in the first dimension, radius2 in the second dimension, and radius3 in the third dimension. The denotation style is applied on the other variables in figure 6.6.

After setting up the dataset on the clustering PHP application, figure 6.7(a) is a scatter plot with the first component and the second component with selecting PC, 1, and 2 in the order. From this components' scatter plot, it is a great start to assign clusters. The concentrated part can remain Cluster1 and sparse points can be

Variable	Detail
Diagnosis	1: Malignant, 0: Benign
radius1	Mean of distances from center to points on the perimeter in the 1 dimension
texture1	Standard deviation of gray-scale values in the 1 dimension
perimeter1	Perimeter of breast tumor in the 1 dimension
area1	Area of breast tumor in the 1 dimension
smoothness1	Local variation in radius lengths in the 1 dimension
compactness1	Perimeter^2 / area - 1 in the 1 dimension
concavity 1	Severity of concave portions of the contour in the 1 dimension
concave points1	Number of concave portions of the contour in the 1 dimension
symmetry 1	Symmetry of breast tumor in the 1 dimension
fractal dimension1	Coastline approximation – 1 in the 1 dimension
radius2	Mean of distances from center to points on the perimeter in the 2 dimension
texture2	Standard deviation of gray-scale values in the 2 dimension
...	...
symmetry 3	Symmetry of breast tumor in the 3 dimension
fractal dimension3	Coastline approximation – 1 in the 3 dimension

Figure 6.6: Attribute information of Breast Cancer Wisconsin dataset.

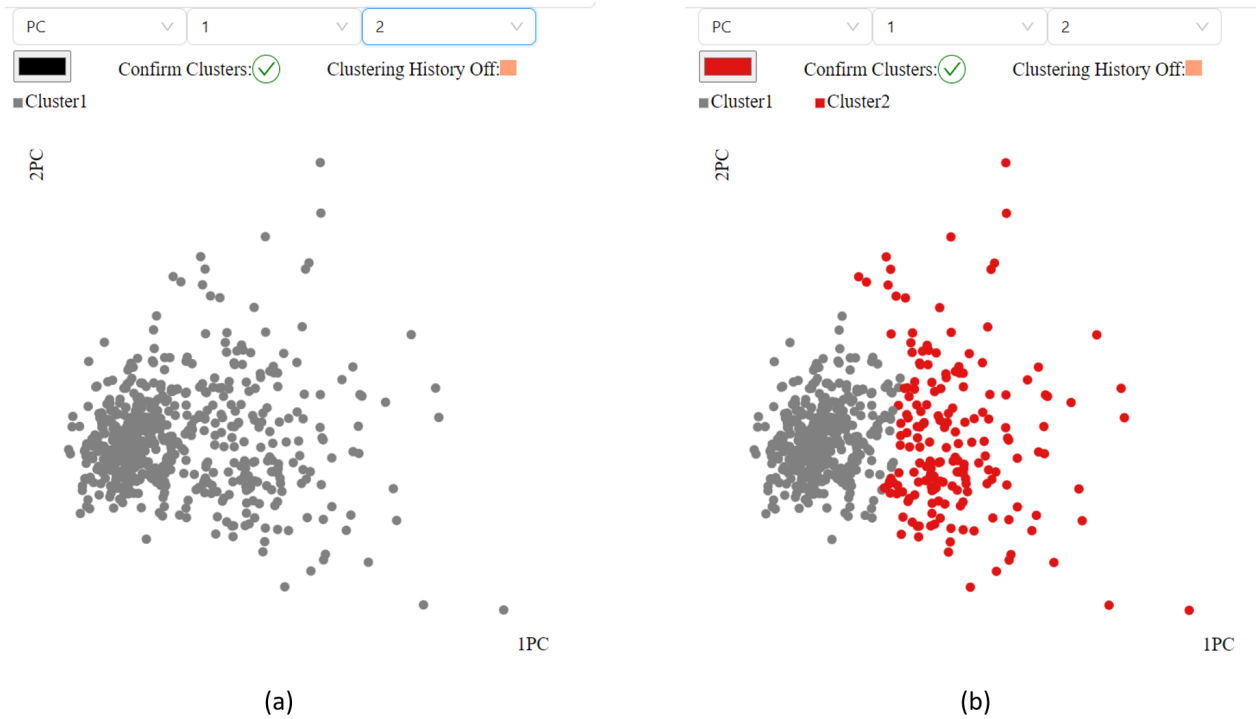


Figure 6.7: (a)Making scatter plot by selecting PC, 1, and 2 , (b) new cluster is created with sparse points.

Cluster2 in figure 6.7(b).

As assigning variables from the components, PCA lines can offer ideas about which variables are interesting

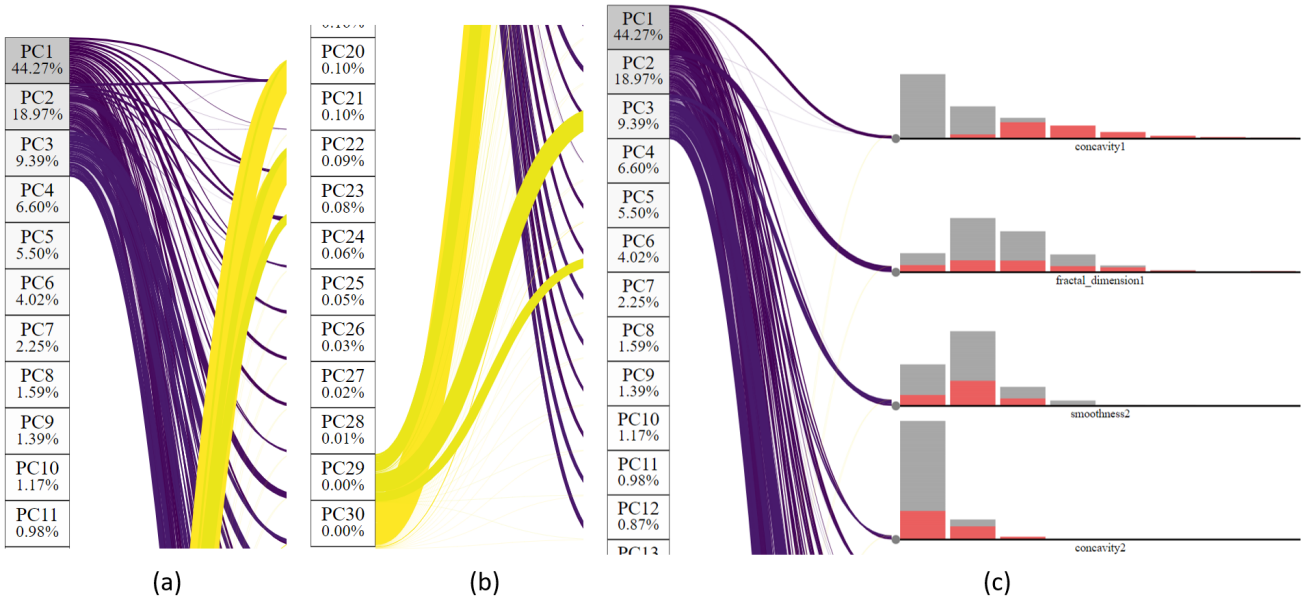


Figure 6.8: (a)As clicking rectangles which contain PCA explained variance ratio on PC1, PC2 and PC3, purple PCA lines are created, (b) Yellow PCA lines are drawn by selecting PC29 and PC30.(c) Based on PCA lines, variables with purple lines and not yellow lines are picked and they are plotted on clustering PHP.

and should be chosen for clustering PHP. in figure 6.8, since the sum of PCA explained variance ratios of PC1, PC2, and PC3 is around 70%, the three boxes are clicked and corresponding purple PCA lines are drawn on (a). When a variable has a big sum of purple lines' width, the variable is likely to be an interesting variable. However, the widths spread pretty equally on (a). Therefore, it would be a great idea to use yellow lines which indicate the last two components on (b). When the yellow lines are thick on the variable, the variable might be redundant for data. Therefore, after considering PCA lines, there are four variables are picked and plotted on clustering PHP.

A user can explore the user interface freely and make combinations among interesting variables from figure 6.8. In the case in figure 6.9, 'concavity1' and 'fractal_dimension1' create the scatter plot. The plot on (a) seems like more likely a circle-shaped cluster in the left bottom corner. Therefore, a user assigns points above the condensed circle shape to cluster2.

Clusters can be finely determined with other combinations of variables. The scatter plot in figure 6.10 takes 'smoothness1' and 'concave_points1'. Two grey points are detected inside of red points. Since the two points are far away from the grey group, so they should belong to red points and are changed into cluster2.

Finally, after confirming clusters, the accuracy is in figure 6.11. 'Yours' indicates accuracy with the clustering PHP application. The result is 86.995%. K-means' accuracy is 85.413%. As a result, the clustering PHP application makes a more accurate result.

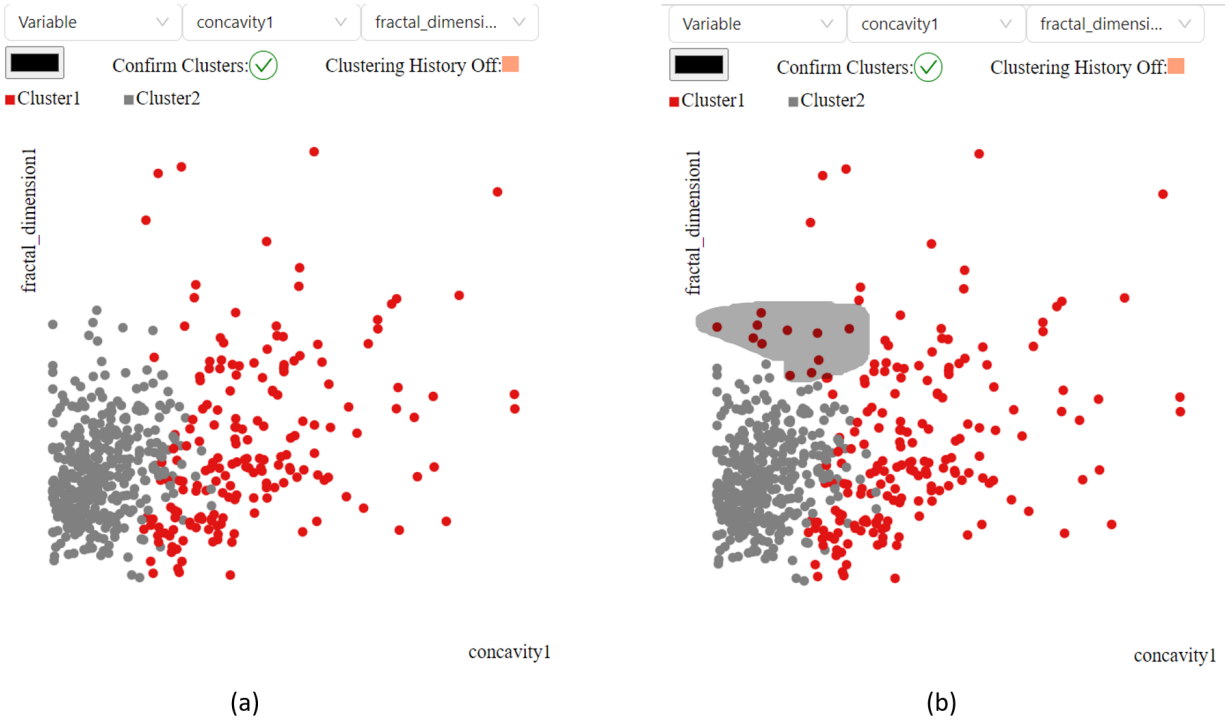


Figure 6.9: (a)selecting interesting subspaces based on clustering PHP's patterns from figure 6.8. In this case, 'concavity1' and 'fractal_dimension1' are plotted on the scatter plot, (b) Assigning points in the lasso selection to Cluster2.

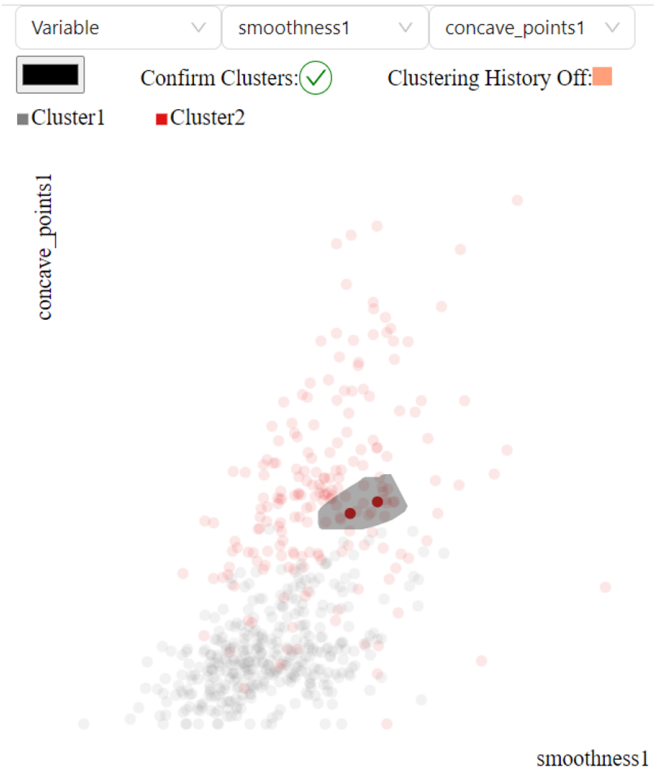


Figure 6.10: Clusters can be finely determined by changing combinations of variables.

If you want to compare your clustering's accuracy with k-means' one, please choose 2 clusters. Yours : 86.995%, K-means: 85.413%

Figure 6.11: It is the result of scenario2. The clustering PHP application helps accuracy to be more accurate.

6.5 Application Scenario3: Dry Bean dataset

Variable	Detail
Area	The area of a bean zone and the number of pixels within its boundaries.
Perimeter	Bean circumference is defined as the length of its border
Major axis length	The distance between the ends of the longest line that can be drawn from a bean.
Minor axis length	The longest line that can be drawn from the bean while standing perpendicular to the main axis.
Aspect ratio	Defines the relationship between Major axis length and Minor axis length
Eccentricity	Eccentricity of the ellipse having the same moments as the region.
Convex area	Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
Equivalent diameter	The diameter of a circle having the same area as a bean seed area.
Extent	The ratio of the pixels in the bounding box to the bean area.
Solidity	Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.
Roundness	Calculated with the following formula: (4piArea)/(Perimeter^2)
Compactness	Measures the roundness of an object: Equivalent diameter/Major axis length
ShapeFactor1,2,3,4	shape form of grain
Class	Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira

Figure 6.12: Attribute information of dry beans dataset.

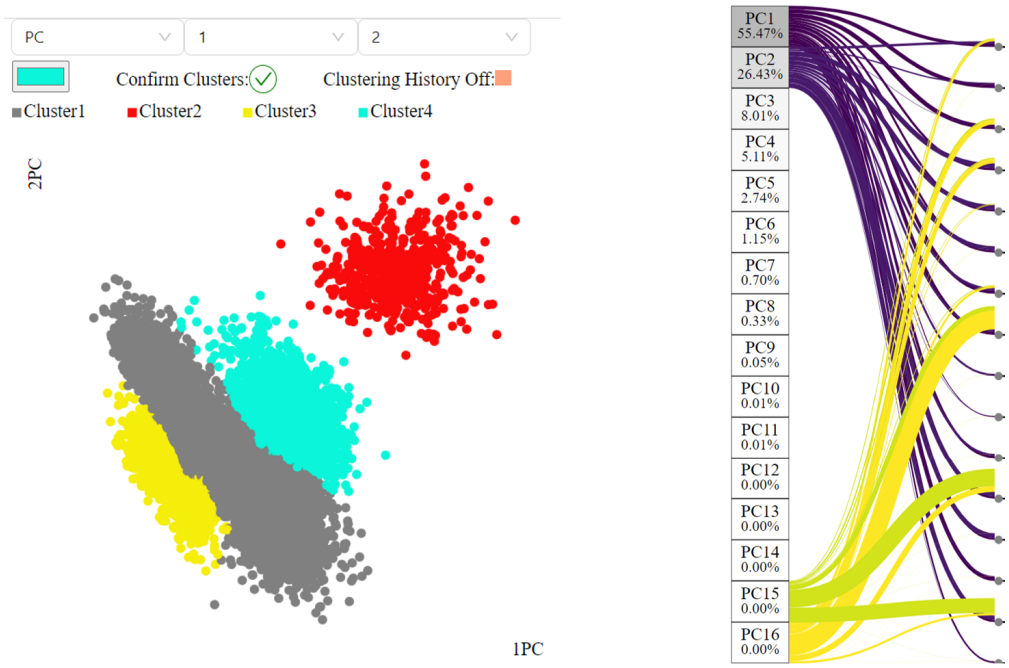
To check whether this application works well or not with big data, this scenario takes a bigger dataset than the previous datasets. The dataset is about different types of dry beans with 13,611 instances and 16 Attributes from the UC Irvine Machine Learning Repository. There are seven classes such as 'Seker', 'Barbunya', 'Bombay', 'Cali', 'Dermonsan', 'Horoz' and 'Sira'. The number of 'Seker' is 2027, 'Barbunya' is 1322, 'Bombay' is 522, Cali is 1630, 'Dermonsan' is 1928, 'Horoz' is 2636, and 'Sira' is 3546. Therefore, the classes are imbalanced. Variables are from images of beans. Variables are in 12 dimensions and four shape forms.

The dry beans dataset is set up on the clustering PHP application. Then, a scatter plot is created with the first principal and the second principal. Four clusters are assigned by a user in figure 6.13(a). After that, a user extend PCA lines in figure 6.13(b). These lines give a hint to choose variables on the clustering PHP.

Previous scenarios have two classes but this dataset has seven classes. To deal with multi-class, a different approach is needed. The new approach is as follows:

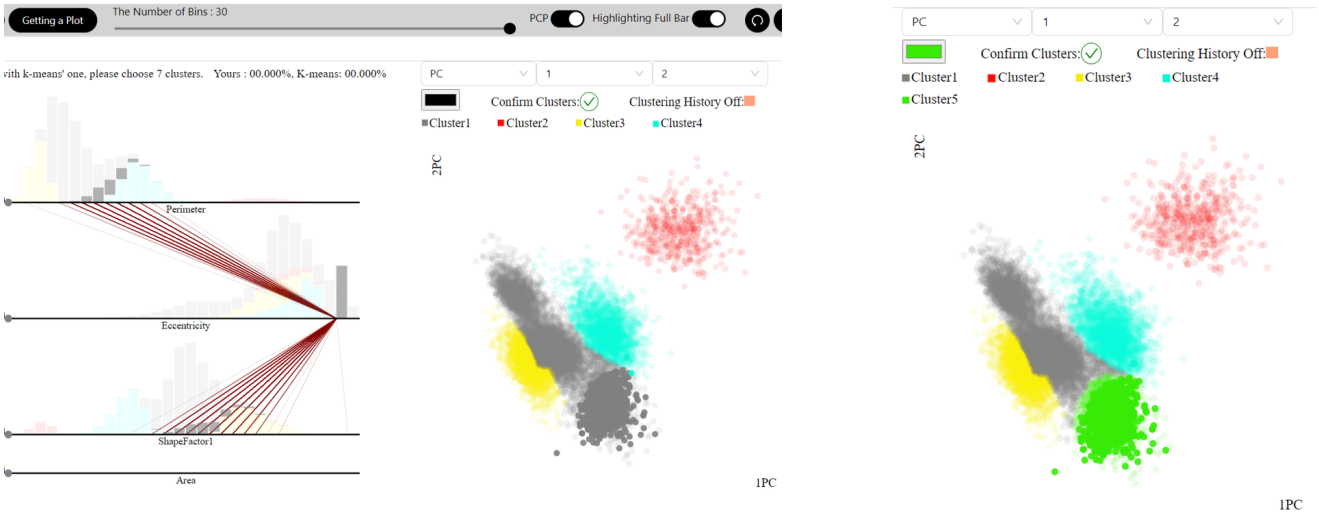
- 1. keeping the scatter plot with the first principal component and the second principal component.
- 2. extending clustering PHP with consideration from PCA lines.
- 3. exploring clustering PHP till new clusters are detected.

Figure 6.14 shows how to follow the new approach. A user keeps the scatter plot of the first principal and the second principal. After that, a user can pick variables for clustering PHP and explore them with the user interface as figure 6.14(a). Plus, the slider is changed to 30 to investigate the cluster PHP in depth. A user puts the size of bins to 30 and clicks a histogram bar on 'Eccentricity'. Related data are highlighted on 'Perimeter', 'Eccentricity', and 'ShapeFactor1'. Corresponding points are also highlighted on the scatter plot. From exploration, the points seem like the other new cluster since they are concentrated and away from the other clusters. On the clustering PHP, their patterns on 'Perimeter' and 'ShapeFactor1' look fine to be a new cluster. As a result, the points create



(a) (b)

Figure 6.13: (a)Points are grouped in four clusters on the scatter plot with the first principal and the second principal by a user. (b)PCA lines are open for choosing variables.



(a) (b)

Figure 6.14: (a)this dataset has seven classes. A different approach is implemented which is useful for multi-classes. The new approach is as follows: While keeping the scatter plot with the first principal component and the second principal component, a user can pick variables for clustering PHP according to PCA lines. Then, as exploring clustering PHP, new clusters are detected. (b) a new Cluster5 is created.

Cluster5 on 6.14(b).

Five clusters are assigned so far, so two more clusters should be found. To find more clusters, a user explores more histogram bars. In figure 6.15(a) and figure 6.15(b), highlighted parts on the scatter plot can be a new class because they are gathered together and highlighted stacked histograms look right to create a new class. Consequently, the highlighted points are in the new cluster6 in figure 6.15(c).

Afterward, variables such as 'Perimeter' and 'Eccentricity' are chosen in the scatter plot to find one more cluster in figure 6.16. No obvious new group is detected but it helps to correct clusters. The highlighted grey points should belong to green group(Cluster5). After considering relationship highlighted parts on the other variables, the points are assigned in Cluster5. Furthermore, the clustering history can help a user to determine clusters correctly as well. In figure 6.17(a), a grey point is detected between green and blue groups. A user might guess this point should belong to one of green or blue one. In this case, clustering history can be checked as 6.17(b). Since the point is the initial setting, changing its cluster is better. After checking the point on clustering PHP, the point is allocated in green(Cluster5) in 6.17(c). As the similar way, more points can be correctly assigned in figure 6.18.

It needs one more cluster to get accuracy but the new cluster is not out yet with possible variables from Figure 6.13(b). Therefore, some variables which have yellow lines can be considered. In figure 6.19, the new cluster is not found but some points are adjusted into a right cluster.

Finally, swapping variables on the visual interaction, a new cluster is observed in figure 6.20(a) and (b), so the part belongs to Cluster7.

As a result, the accuracy with clustering PHP returns much better result as 80.736% than K-mean as 54.478%. It is because many combinations of variables were considered in this scenario. Detailed clustering correction were also done. Furthermore, The dataset on this scenario is imbalanced and big. According to 6.2 Accuracy, the clustering PHP application works well with imbalanced big datasets.

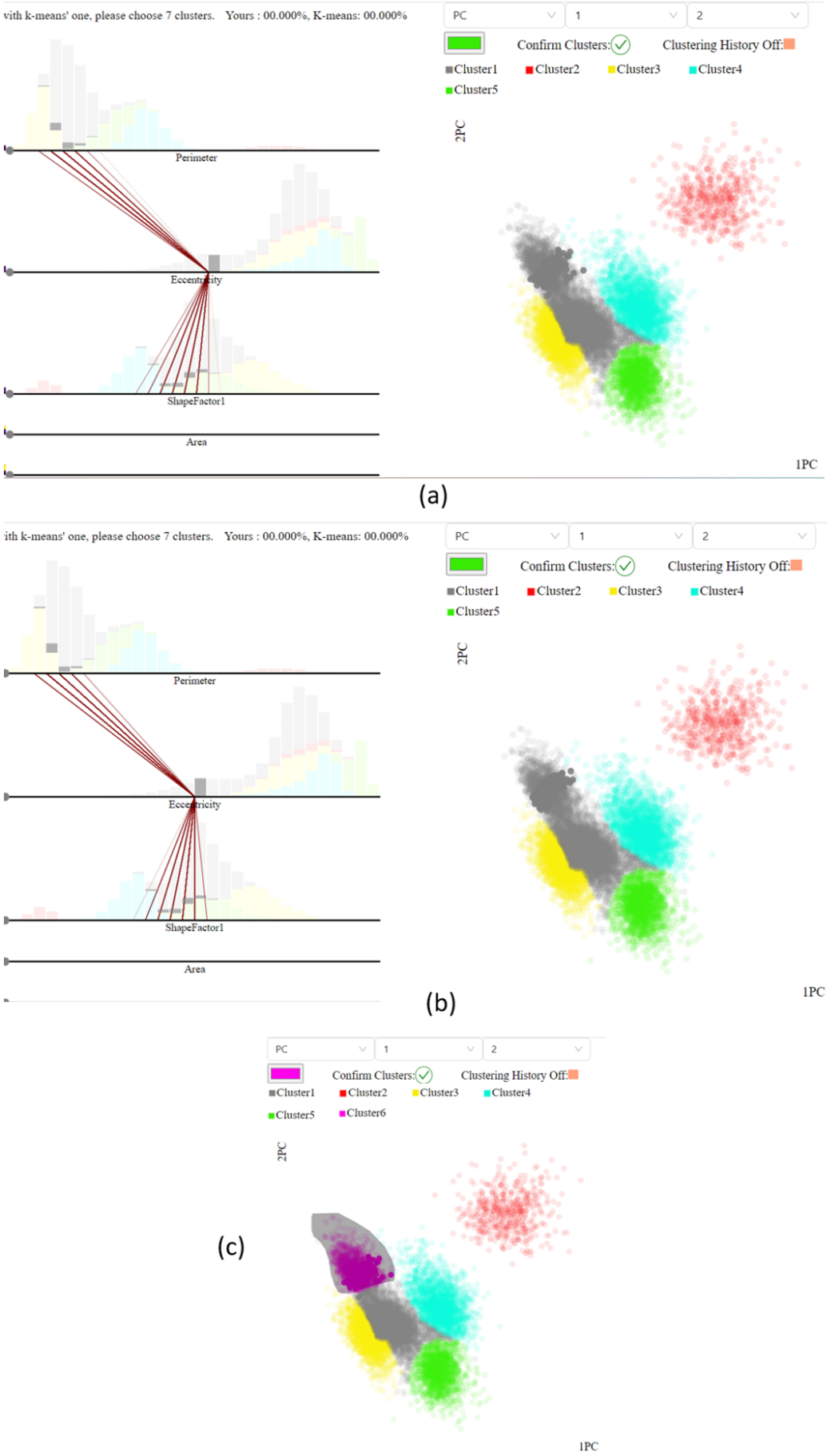


Figure 6.15: (a)highlighted parts on the scatter plot are concentrated. (b) Also, the highlighting parts are next to the points from(a) on the scatter plot. It is a sign to be a new class. (c) Cluster6 is created.

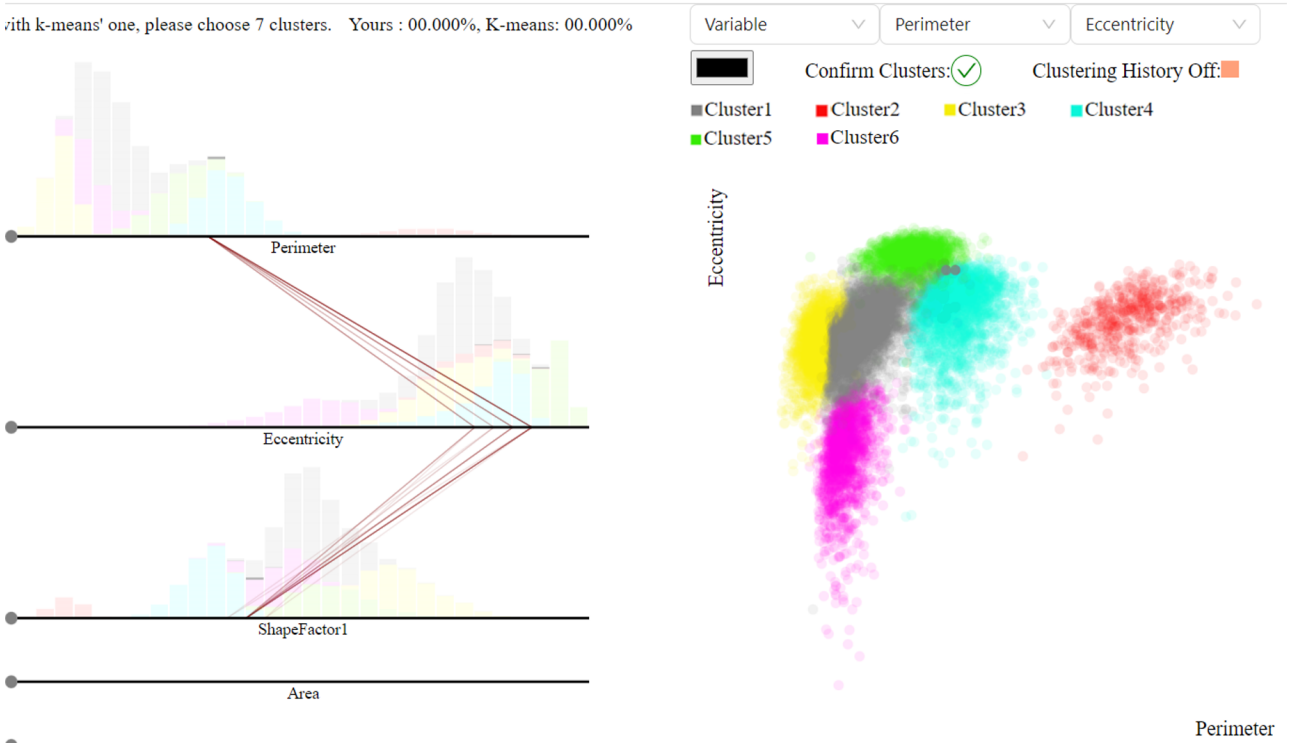
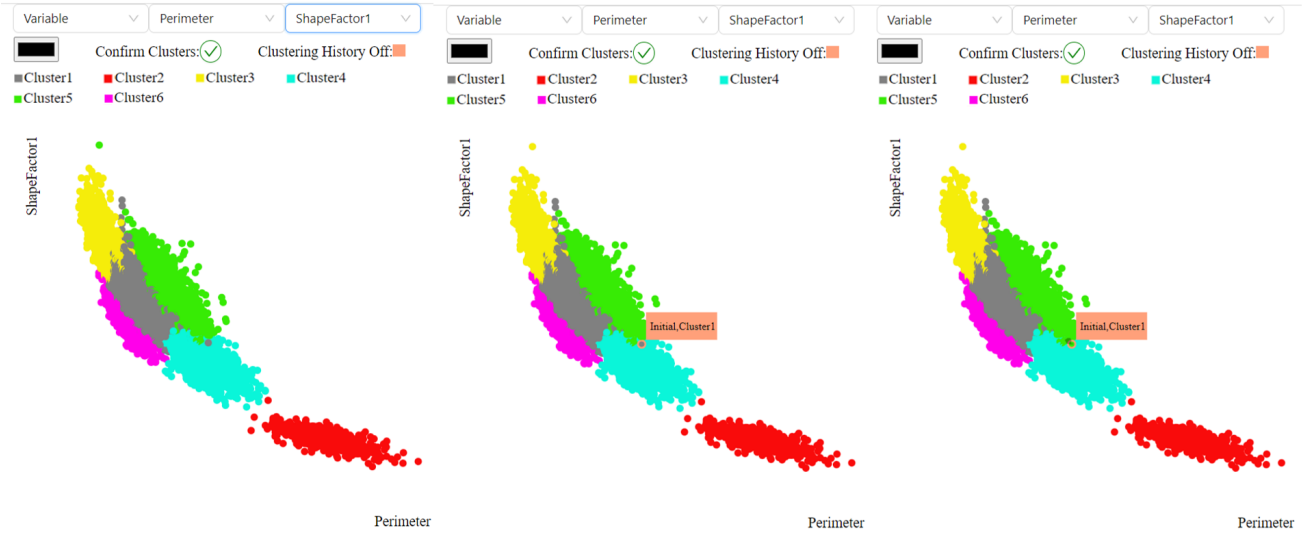


Figure 6.16: To detect one more cluster, the other variables are chosen in the scatter plot. After observing patterns of highlighted parts on the clustering PHP and the scatter plot, the highlighted grey points should be changed from Cluster1 to Cluster5.



(a) (b) (c)

Figure 6.17: (a)There is a grey point in Cluster1 between green group(Cluster5) and blue group(Cluster4). (b)On the clustering history, the Cluster 1 is the initial setting on the point. (c)A user determines the point not into Cluster1. After checking the point on the cluster PHP, the point is assigned to Cluster5.

Figure 6.18: Changing variables can help a user to determine clustering correctly.

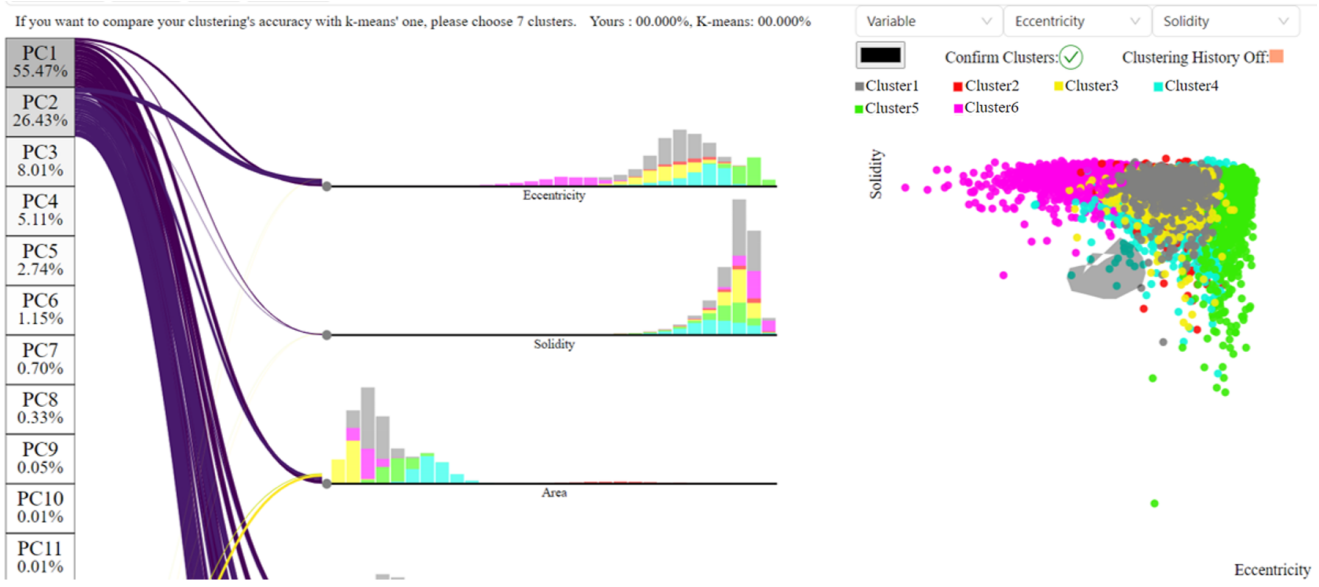


Figure 6.19: Detailed correction of clustering is possible with the application.

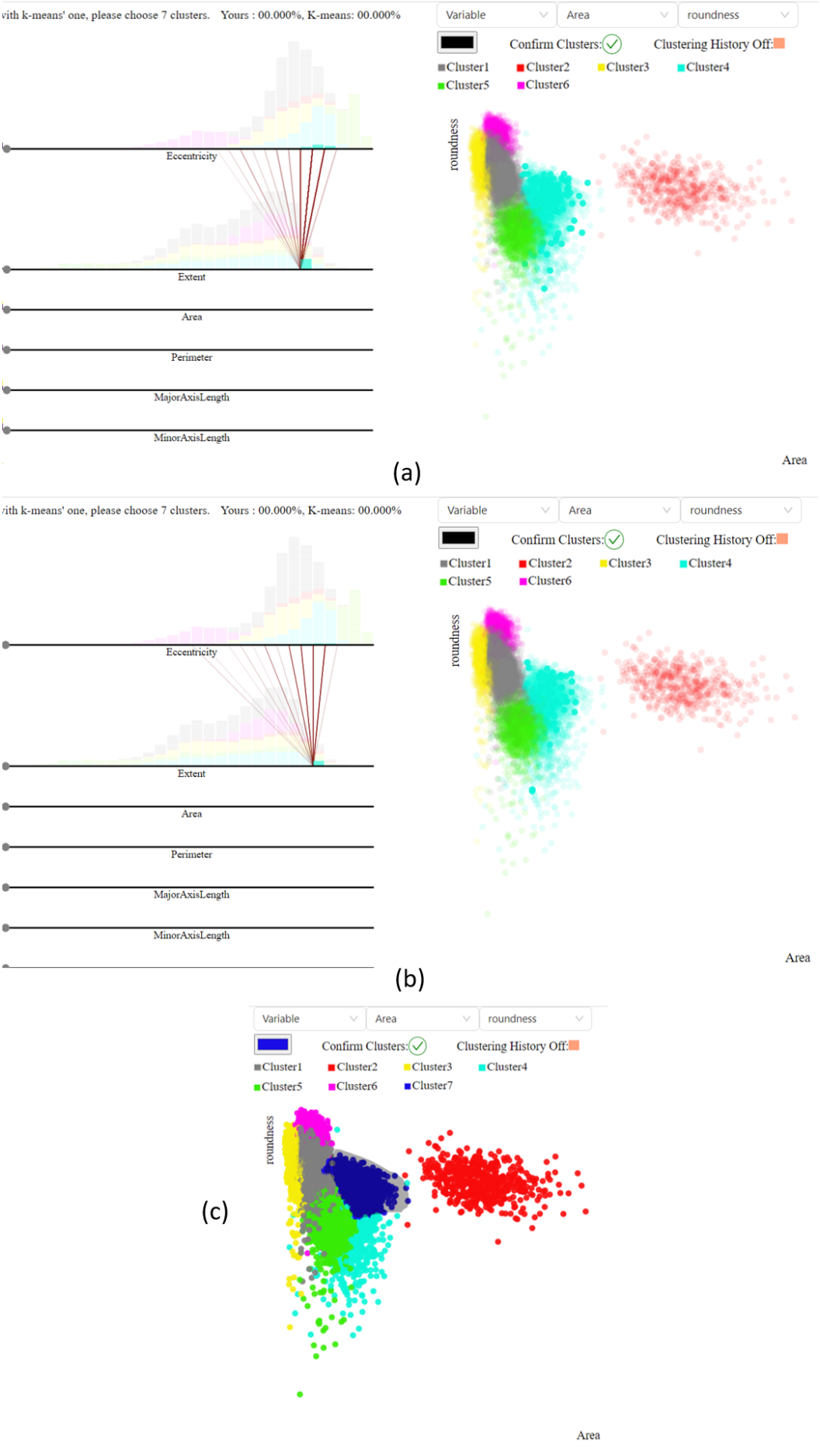


Figure 6.20: The last cluster is found by swapping variables.

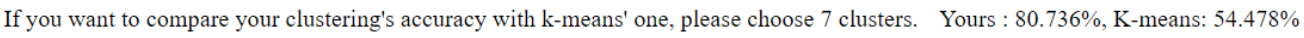


Figure 6.21: The result from the scenario3. Using the clustering PHP application improves accuracy.

7 Conclusion and Discussion

The ”clustering PHP application” helps users to conduct subspace clustering of multivariate data more easily and more efficiently. The application creates synergy by combining Parallel Histogram Plot, Principal Component analysis, a scatter plot with the ability for user interaction. PCA lines help users select variables by showing their respective contribution. Clustering PHP help users detect interesting relationship among variables. Highlighting and coloring events are immediately synchronized between clustering PHP and the scatter plot. With the interface, users can visually find relationships among variables and detect clustering groups with interactions on subspace. As a result, clustering PHP application can be helpful to explore data for clustering with multi-dimensional data. The application especially performs well with imbalanced big data.

The ”clustering PHP application” has several limitations and several aspects around the ”clustering PHP application” should be improved further in the future. Since clustering is decided by a user, the user's understanding of the data and the level of clustering knowledge are important. If a user does not have a background of these theme, using this application might be challenging and the result may not be accurate. Thus, if a dataset has so many variables that the PCA lines are overlapping, selecting variables might be difficult. In that case, users should try several combinations of variables on the application to get a meaningful result. This can be tedious and could be automated in the future. The user interaction can also be improved by adding functions. This application optimizes the sorting process. For very big dataset this sorting may take a few seconds. If the response becomes quicker in the future, this could improve user experience.

Acknowledgements

I sincerely appreciate Haiyan Yang. She dedicated her time and gave me clear feedback on my thesis. She also always replied to my questions very quickly and supervised my master's thesis on a weekly basis. Thanks to Prof. Dr. Renato Pajarola, I had a chance to study in depth about data visualization. On 6.Experimental Result, datasets of all scenarios and attributes information of the datasets are provided from the UC Irvine Machine Learning Repository(https://archive.ics.uci.edu/).

Bibliography

[ANB21] Syed Muqeet Aqib, Haque Nawaz, and Shah Muhammad Butt. Analysis of merge sort and bubble sort in python, php, javascript, and c language. *International Journal*, 10(2), 2021.

[AW10] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[BKS22] Jinwook Bok, Bohyoung Kim, and Jinwook Seo. Augmenting parallel coordinates plots with color-coded stacked histograms. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2563–2576, 2022.

[DH04] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 29, 2004.

[DK10] Aritra Dasgupta and Robert Kosara. Pargnostics: Screen-space metrics for parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1017–1026, 2010.

[GFVS12] Stephan Günnemann, Ines Färber, Kittipat Virochsiri, and Thomas Seidl. Subspace correlation clustering: finding locally correlated dimensions in subspace projections of the data. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 352–360, 2012.

[HAV13] Lynne Williams Hervé Abdi and Domininique Valentin. Multiple factor analysis: principal component analysis for multitable and multiblock data sets. *WIREs Computational Statistics*, 5(2):149–179, 2013.

[HK99] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering : Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25 th International Conference on Very Large Databases, 1999*, pages 506–517, 1999.

[HP18] Juhua Hu and Jian Pei. Subspace multi-clustering: a review. *Knowledge and information systems*, 56:257–284, 2018.

[JSEB19] Mohammad Jafarzadegan, Faramarz Safi-Esfahani, and Zahra Beheshti. Combining hierarchical clustering approaches using the pca method. *Expert Systems with Applications*, 137:1–10, 2019.

[Kas17] Alboukadel Kassambara. *Principal component analysis*, volume 2. 2017.

[KBH06] Robert Kosara, Fabian Bendix, and Helwig Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, jul 2006.

[KKZ09] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Acm transactions on knowledge discovery from data (tkdd)*, 3(1):1–58, 2009.

[KM+13] Trupti M Kodinariya, Prashant R Makwana, et al. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[LN03] Qing Li and C. North. Empirical comparison of dynamic query sliders and brushing histograms. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, pages 147–153, 2003.

[LYB+17] Yang Lei, Dai Yu, Zhang Bin, Yang Yang, et al. Interactive-means clustering method based on user behavior for different analysis target in medicine. *Computational and Mathematical Methods in Medicine*, 2017, 2017.

[OHT20] Yassine Ouali, Céline Hudelot, and Myriam Tami. Autoregressive unsupervised image segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 142–158, 2020.

[PHL04] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

[Rus69] Enrique Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, 1969.

[Suh05] Diana Suhr. Principal component analysis vs. exploratory factor analysis. *SUGI 30 proceedings*, 203(230):1–11, 2005.

[Swe88] John A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[SZS+17] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A. Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2017.

[TLJ+19] Yosuke Tanigawa, Jiehan Li, Johanne M Justesen, Heiko Horn, Matthew Aguirre, Christopher DeBoever, Chris Chang, Balasubramanian Narasimhan, Kasper Lage, Trevor Hastie, et al. Components of genetic associations across 2,138 phenotypes in the uk biobank highlight adipocyte biology. *Nature communications*, 10(1):4064, 2019.

[ZLH+16] Fangfang Zhou, Juncai Li, Wei Huang, Ying Zhao, Xiaoru Yuan, Xing Liang, and Yang Shi. Dimension reconstruction for visual exploration of subspace clusters in high-dimensional data. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 128–135, 2016.