

# DM\_PS4\_all

Fan Ye, Jinming Li, Xiangmeng Qin

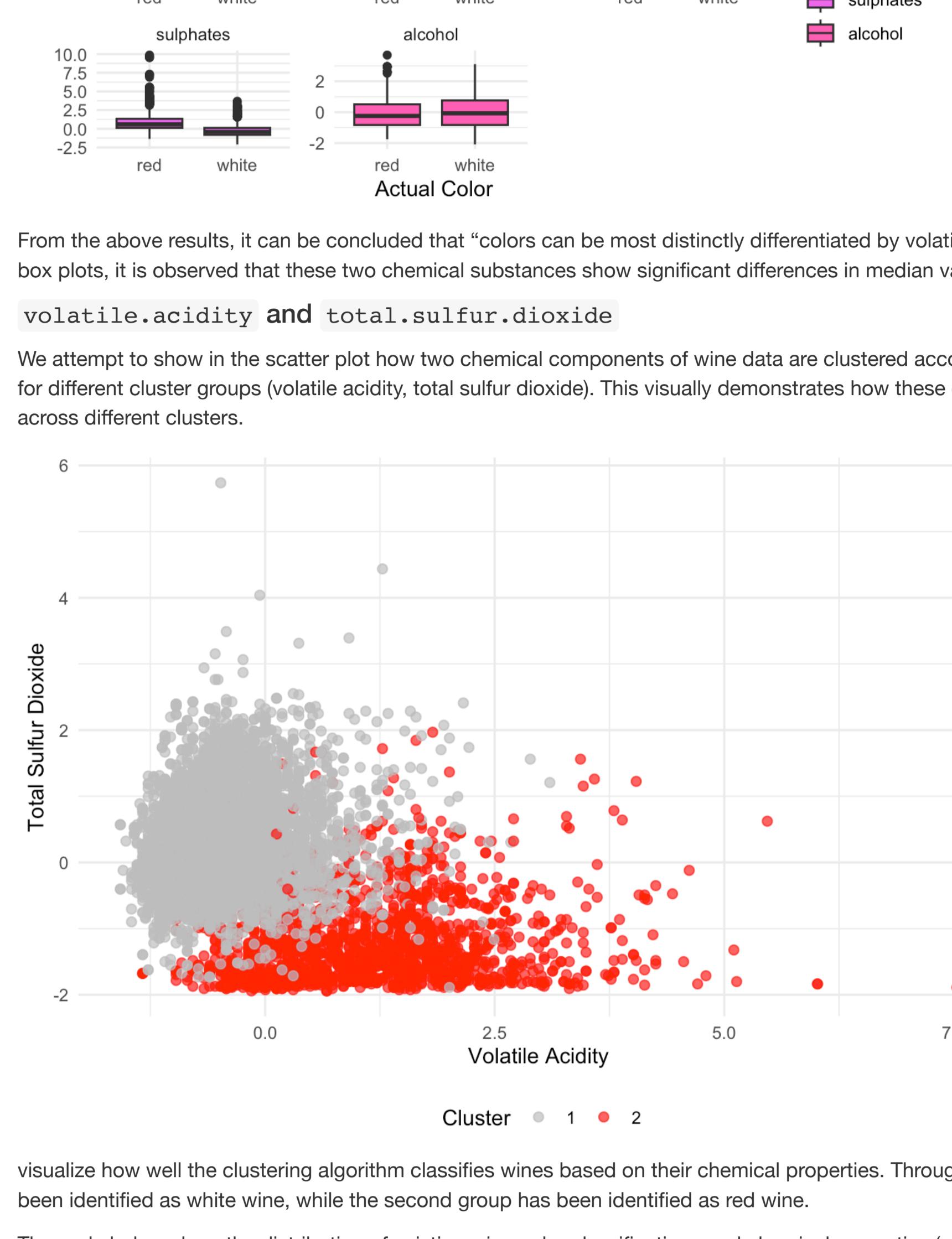
2024-04-22

## Question 1: Clustering and PCA

### Clustering

#### Color of wines

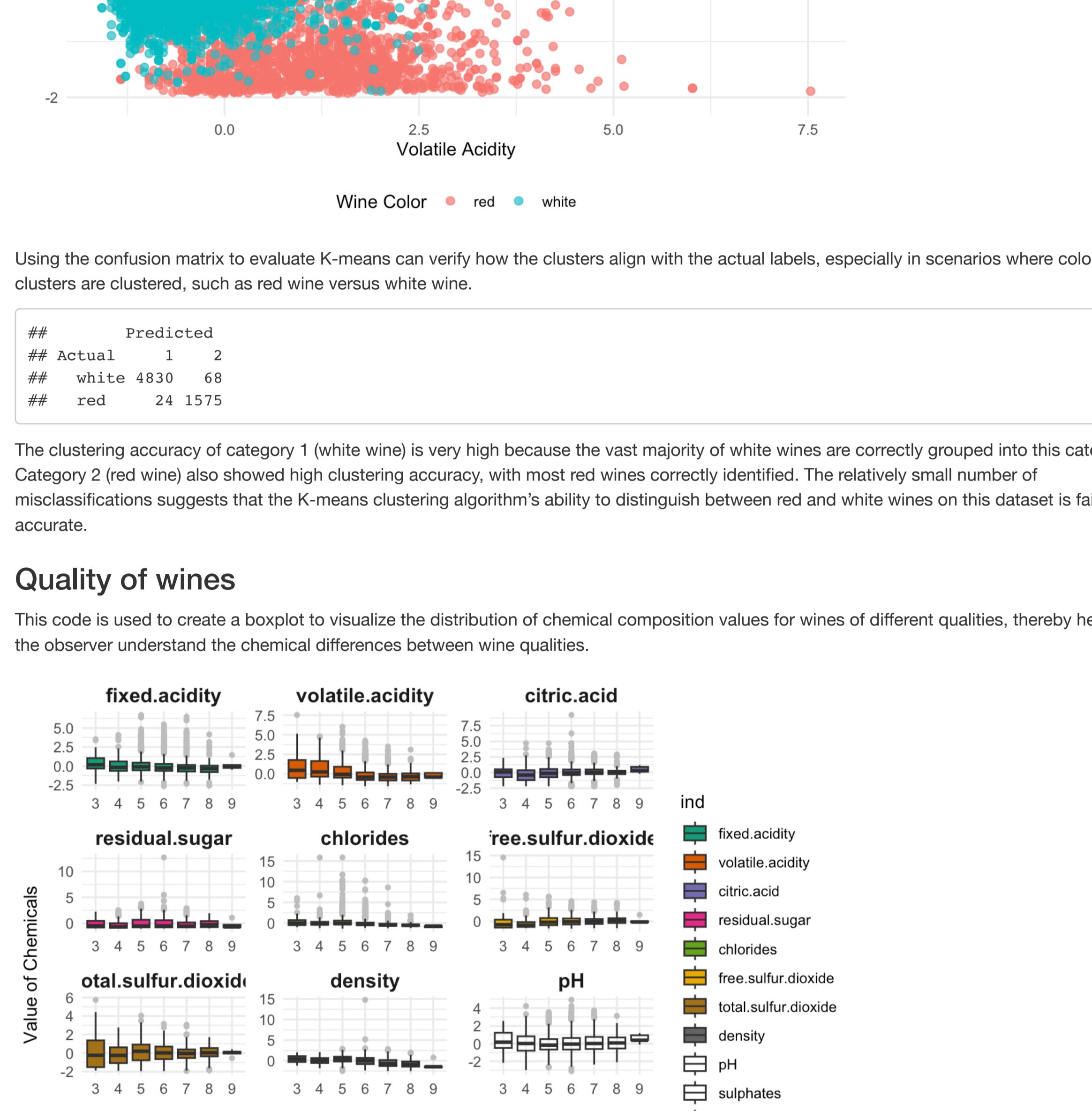
We standardizes the features of a wine dataset, excluding `quality` and `color`, and performs K-means clustering with two different numbers of centers (2 and 7). Then we visualizes the distribution of various chemical properties across actual wine colors using a box plot created with `ggplot2`.



From the above results, it can be concluded that "colors can be most distinctly differentiated by volatile acidity and total sulfur dioxide." In the box plots, it is observed that these two chemical substances show significant differences in median values among different colors of wine.

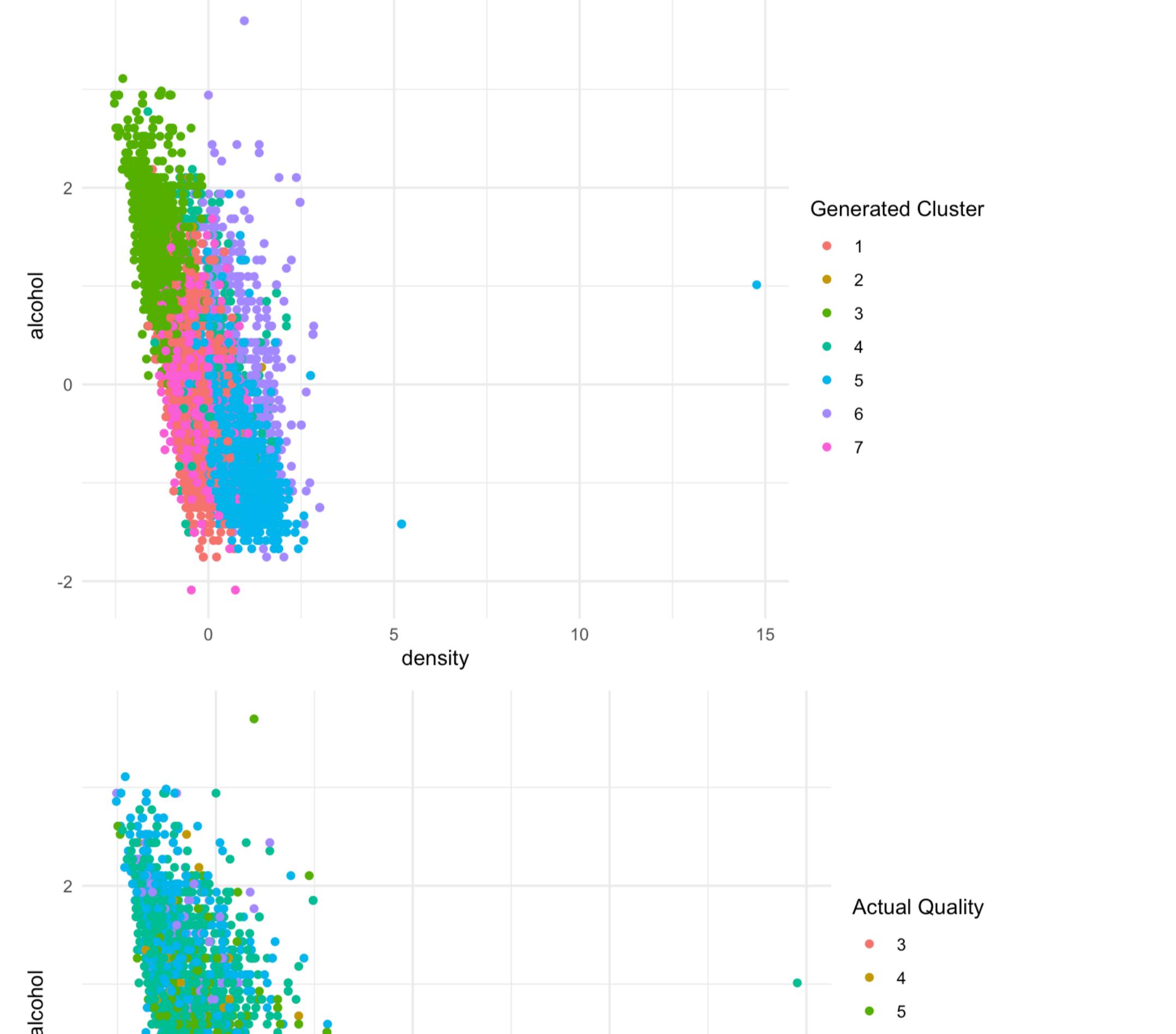
#### volatile.acidity and total.sulfur.dioxide

We attempt to show in the scatter plot how two chemical components of wine data are clustered according to color, marked by different colors for different cluster groups (volatile acidity, total sulfur dioxide). This visually demonstrates how these chemical components are distributed across different clusters.



visualize how well the clustering algorithm classifies wines based on their chemical properties. Through clustering analysis, the first group has been identified as white wine, while the second group has been identified as red wine.

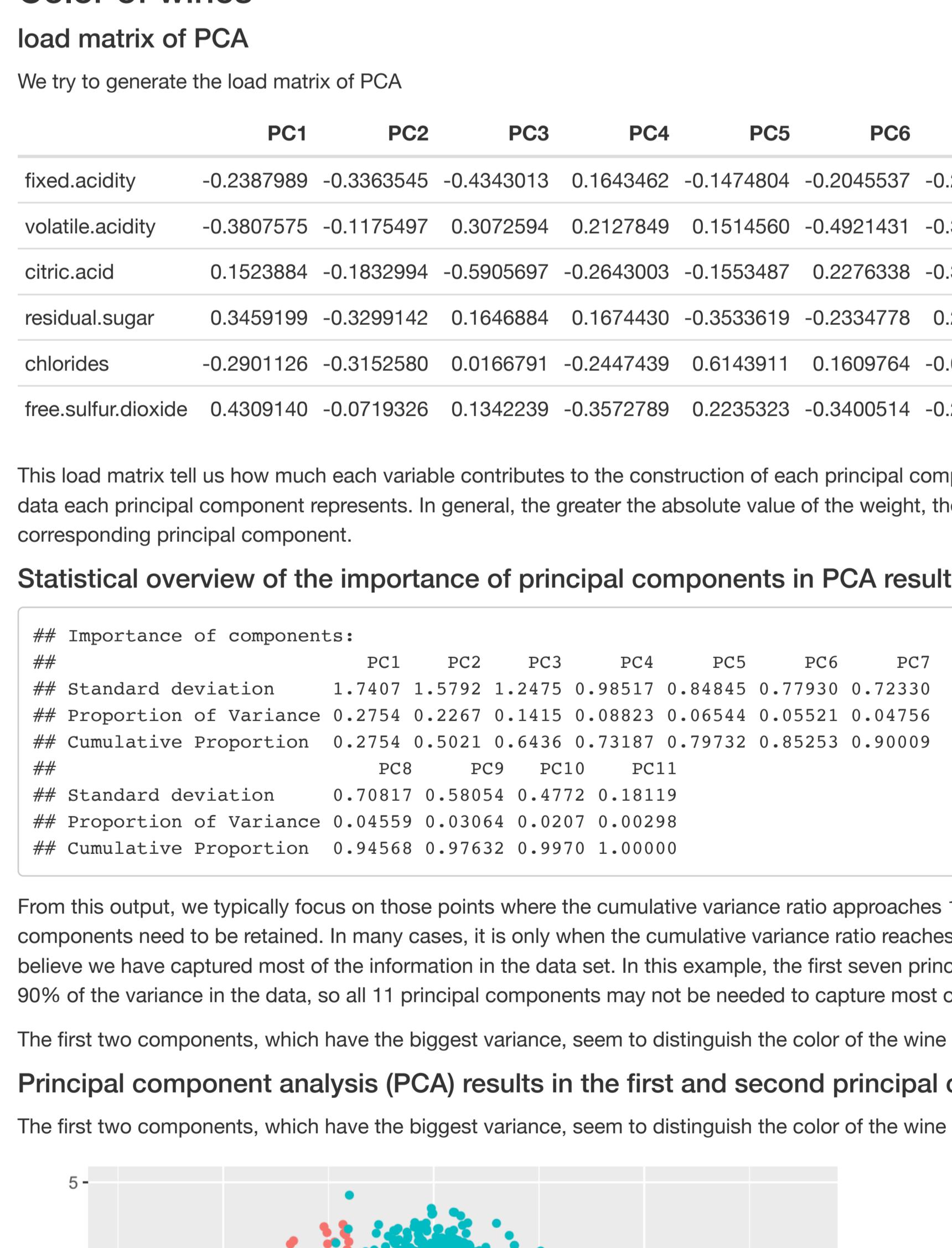
The code below shows the distribution of existing wine color classifications and chemical properties (volatile acidity and total sulfide) in the data set.



Using the confusion matrix to evaluate K-means can verify how the clusters align with the actual labels, especially in scenarios where color clusters are clustered, such as red wine versus white wine.

#### Quality of wines

This code is used to create a boxplot to visualize the distribution of chemical composition values for wines of different qualities, thereby helping the observer understand the chemical differences between wine qualities.



Based on the median values of these characteristics, we predict that at least density and alcohol can distinguish between high quality wines.

#### "density" and "alcohol"

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

#### Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
fixed.acidity	-0.2387989	-0.3363545	-0.4343013	0.1643462	-0.1474804	-0.2045537	-0.2830794	0.4012356	-0.3440567	0.2812677	0.3346793
volatile.acidity	-0.3807575	-0.1175497	0.3072594	0.2127849	0.1514560	-0.4921431	-0.3891598	-0.0874351	0.4969327	-0.1521767	0.0847718
citric.acid	0.1523884	-0.1832994	-0.5905697	-0.2643003	-0.1553487	0.2276338	-0.3812850	-0.2934123	0.4026887	-0.2344633	-0.0011090
residual.sugar	0.3459199	-0.3299142	0.1646884	0.1674430	-0.2447439	0.6143911	0.1609764	-0.0460682	-0.4715168	0.1966302	0.0434376
chlorides	-0.2901126	-0.3152580	0.0166791	-0.2447439	0.6143911	0.1609764	-0.0460682	-0.4715168	0.1966302	0.0434376	
free.sulfur.dioxide	0.4309140	-0.0719326	0.1342239	-0.3572789	0.2235323	-0.3400514	-0.2993632	0.2078076	-0.3666563	-0.4802433	-0.0002125

From this output, we typically focus on those points where the cumulative variance ratio approaches 1 to determine how many principal components need to be retained. In many cases, it is only when the cumulative variance ratio reaches a high value (such as 80% or 90%) that we believe we have captured most of the information in the data set. In this example, the first seven principal components already explain more than 90% of the variance in the data, so all 11 principal components may not be needed to capture most of the information in the data set.

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:

## Standard deviation PC1 PC2 PC3 PC4 PC5 PC6 PC7

## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756

## Cumulative Proportion 0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009

## Standard deviation 0.70817 0.58054 0.4772 0.18119

## Proportion of Variance 0.04559 0.03064 0.0207 0.00298

## Cumulative Proportion 0.94568 0.97632 0.9970 1.00000

From this output, we typically focus on those points where the cumulative variance ratio approaches 1 to determine how many principal components need to be retained. In many cases, it is only when the cumulative variance ratio reaches a high value (such as 80% or 90%) that we believe we have captured most of the information in the data set. In this example, the first seven principal components already explain more than 90% of the variance in the data, so all 11 principal components may not be needed to capture most of the information in the data set.

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:

## Standard deviation PC1 PC2 PC3 PC4 PC5 PC6 PC7

## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756

## Cumulative Proportion 0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009

## Standard deviation 0.70817 0.58054 0.4772 0.18119

## Proportion of Variance 0.04559 0.03064 0.0207 0.00298

## Cumulative Proportion 0.94568 0.97632 0.9970 1.00000

From this output, we typically focus on those points where the cumulative variance ratio approaches 1 to determine how many principal components need to be retained. In many cases, it is only when the cumulative variance ratio reaches a high value (such as 80% or 90%) that we believe we have captured most of the information in the data set. In this example, the first seven principal components already explain more than 90% of the variance in the data, so all 11 principal components may not be needed to capture most of the information in the data set.

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:

## Standard deviation PC1 PC2 PC3 PC4 PC5 PC6 PC7

## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756

## Cumulative Proportion 0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009

## Standard deviation 0.70817 0.58054 0.4772 0.18119

## Proportion of Variance 0.04559 0.03064 0.0207 0.00298

## Cumulative Proportion 0.94568 0.97632 0.9970 1.00000

From this output, we typically focus on those points where the cumulative variance ratio approaches 1 to determine how many principal components need to be retained. In many cases, it is only when the cumulative variance ratio reaches a high value (such as 80% or 90%) that we believe we have captured most of the information in the data set. In this example, the first seven principal components already explain more than 90% of the variance in the data, so all 11 principal components may not be needed to capture most of the information in the data set.

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:

## Standard deviation PC1 PC2 PC3 PC4 PC5 PC6 PC7

## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756

## Cumulative Proportion 0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009

## Standard deviation 0.70817 0.58054 0.4772 0.18119

## Proportion of Variance 0.04559 0.03064 0.0207 0.00298

## Cumulative Proportion 0.94568 0.97632 0.9970 1.00000

From this output, we typically focus on those points where the cumulative variance ratio approaches 1 to determine how many principal components need to be retained. In many cases, it is only when the cumulative variance ratio reaches a high value (such as 80% or 90%) that we believe we have captured most of the information in the data set. In this example, the first seven principal components already explain more than 90% of the variance in the data, so all 11 principal components may not be needed to capture most of the information in the data set.

The first two components, which have the biggest variance, seem to distinguish the color of the wine well;

Principal component analysis (PCA) results in the first and second principal components

The first two components, which have the biggest variance, seem to distinguish the color of the wine well.

This load matrix tell us how much each variable contributes to the construction of each principal component. It can explain what aspects of the data each principal component represents. In general, the greater the absolute value of the weight, the greater the influence of the variable on the corresponding principal component.

#### Statistical overview of the importance of principal components in PCA results

## Importance of components:

## Standard deviation PC1 PC2 PC3 PC4 PC5 PC6 PC7

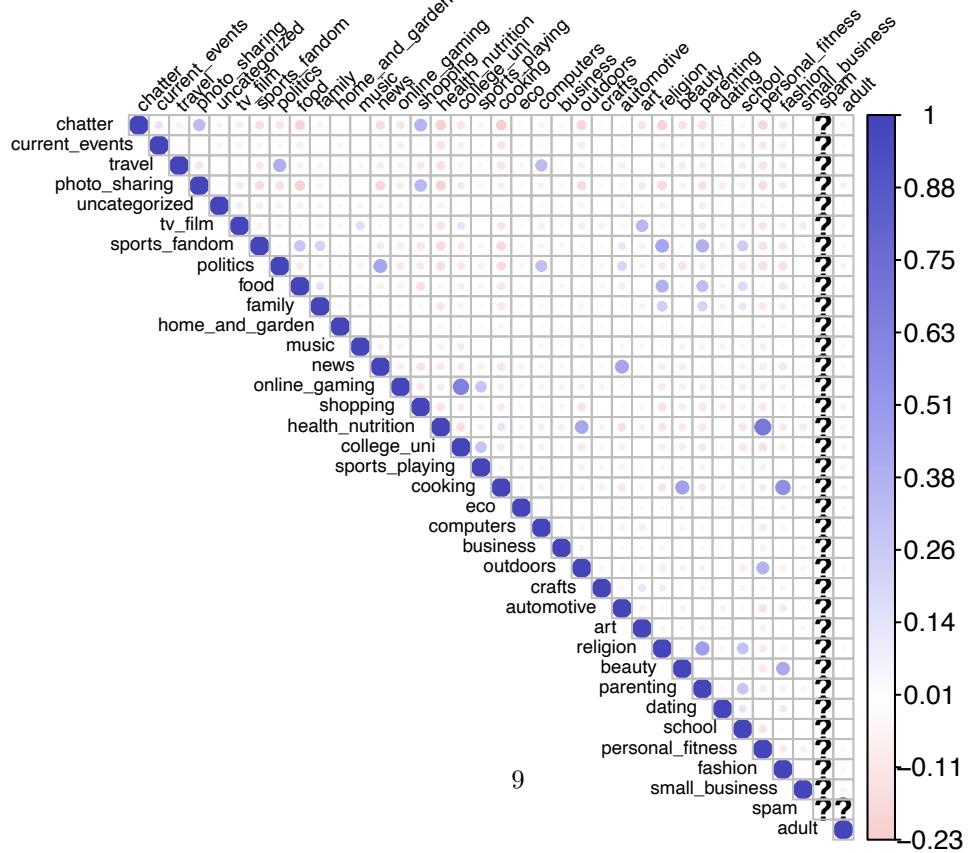
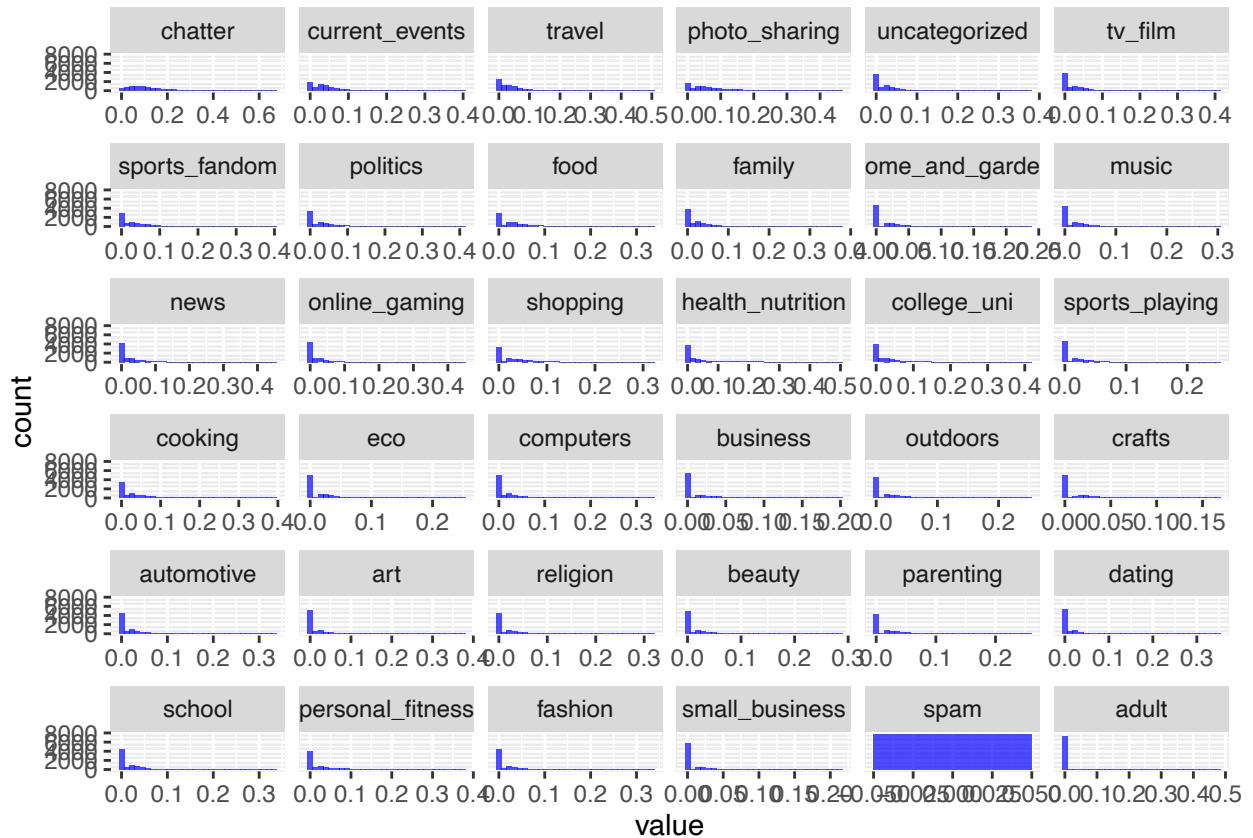
## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756

## Cumulative Proportion 0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009

## Standard deviation 0.70817 0.58054 0

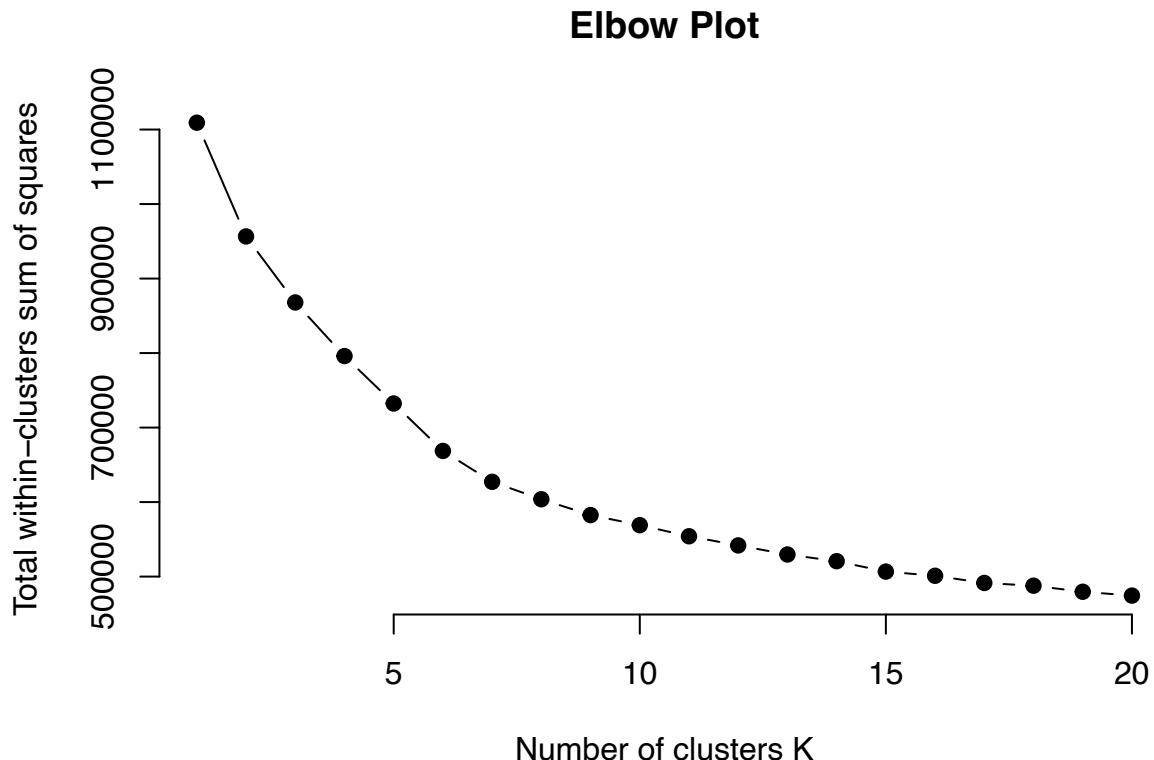
## Question 2: Market segmentation

### Step1: Data Preprocessing and Exploratory Data Analysis (EDA)



This graph shows how closely related different topics are based on social media data. Categories that are often talked about together, like ‘home\_and\_garden’ and ‘family’, show a strong positive connection with big, dark blue circles. Other topics like ‘sports\_playing’ and ‘health\_nutrition’ are somewhat related, with smaller, lighter blue circles. There are also topics that don’t seem to be talked about together much at all; they have very light blue or no circles connecting them.

## Step2: k-means and Elbow Method



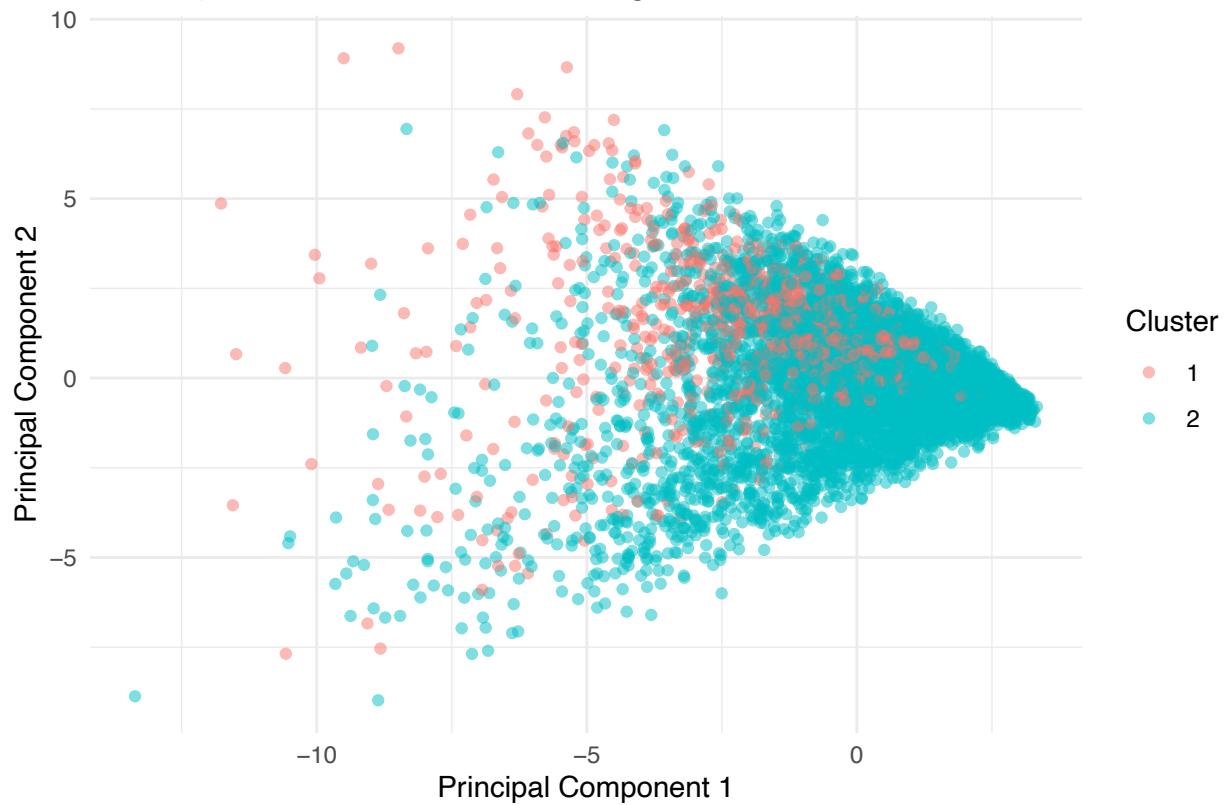
```
## [1] 2
```

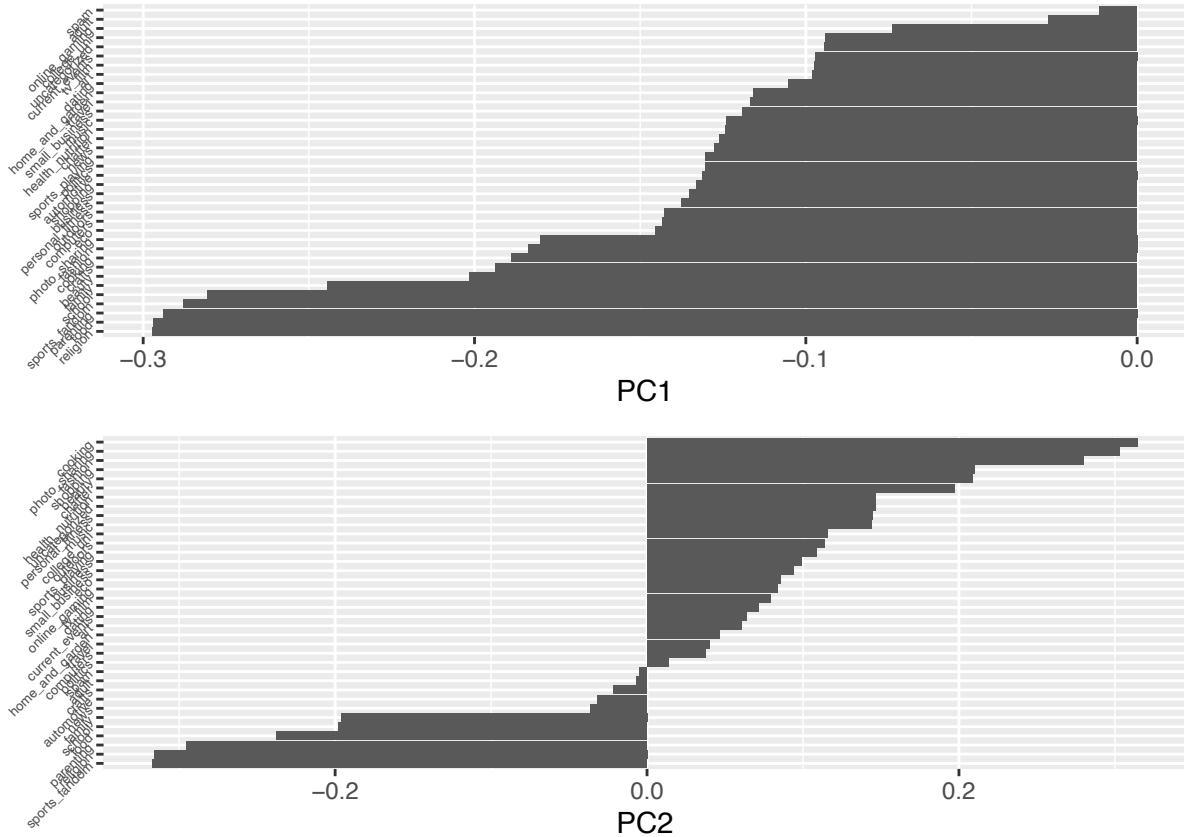
After calculating, we can find out the the optimal number of clusters is 2.

## Step3: PCA\_Clustering

First, we run a k-means clustering algorithm with a predetermined number of two clusters to categorize the social data into groups. We then perform Principal Component Analysis (PCA) on the same dataset to reduce its dimensions while retaining the essence of the original data.

PCA biplot with K-means Clustering





## Discussion and Conclusion

Principal Component 1 (PC1) suggests a significant negative relationship with variables such as 'sports\_fan', 'photo\_sharing', and 'personal\_care'. This implies a group within the audience that is less likely to be engaged with sports-related content, photo sharing, or personal care topics. This segment might represent consumers more interested in areas less correlated with an active lifestyle or personal image.

Principal Component 2 (PC2) shows a contrasting trend, where the same variables ('sports\_fan', 'photo\_sharing', 'personal\_care') present a positive relationship. This indicates a segment of the audience that is more aligned with an active, healthy lifestyle, which may also be interested in sharing their experiences and tips online, suggesting a higher engagement with social media.

### Key Segments Defined

**Discreet Consumers:** Fitted with PC1, this segment is less vocal on social media about personal activities and interests. They may value privacy and be interested in products without public endorsement. Marketing strategies here may focus on direct benefits instead of the need for social sharing.

**Active Lifestyle Consumers:** This segment fits with PC2, where interest in sports, health, and sharing content about personal care on social media is high. NutrientH2O can target this group with content and products related to fitness, health supplements, and active living.

## **Recommendations for NutrientH20**

1. Develop different campaigns for each segment. For Active Lifestyle Consumers, focus on social media engagement, sponsorships with sports influencers, and sharing success stories. For Discreet Consumers, focus on the quality of products and privacy-respecting marketing channels.
2. Tailor the positioning of products to align with each segment's values—highlighting the social aspect and community for Active Lifestyle Consumers and emphasizing product efficacy for Discreet Consumers.
3. Create content that appeals to the varied interests within each segment. Engage the Active Lifestyle Consumers with interactive and visually appealing posts, while providing informative and detailed content for Discreet Consumers.

## **Question 3: Association rules for grocery purchases**

We employed the arules to discover association rules that reveal the relationships between items purchased together to analyze the data from the grocery transactions. By setting the thresholds for support, confidence, and lift, we aimed to uncover meaningful patterns in shopping behavior.

### **Parameters and Methodology**

Support: A minimum support threshold of 0.001 was chosen to ensure we capture frequent enough patterns without focusing solely on the most common items. Confidence: We set a moderate confidence level of 0.25 to strike a balance between the reliability of the rule and the inclusion of less obvious associations. Lift: we focused on rules with the highest lift values after the generation of rules to spotlight the most significant associations.

### **Associations Found**

Alcoholic Beverages Combination: The rule involving {bottled beer, red/ blush wine} => {liquor} with a lift of 35.71 and confidence of 39.58% is indicative of a strong association between these beverages. It suggests that customers who purchase beer and wine are also likely to purchase liquor, pointing towards a trend in buying multiple types of alcoholic beverages together, possibly for events or gatherings.

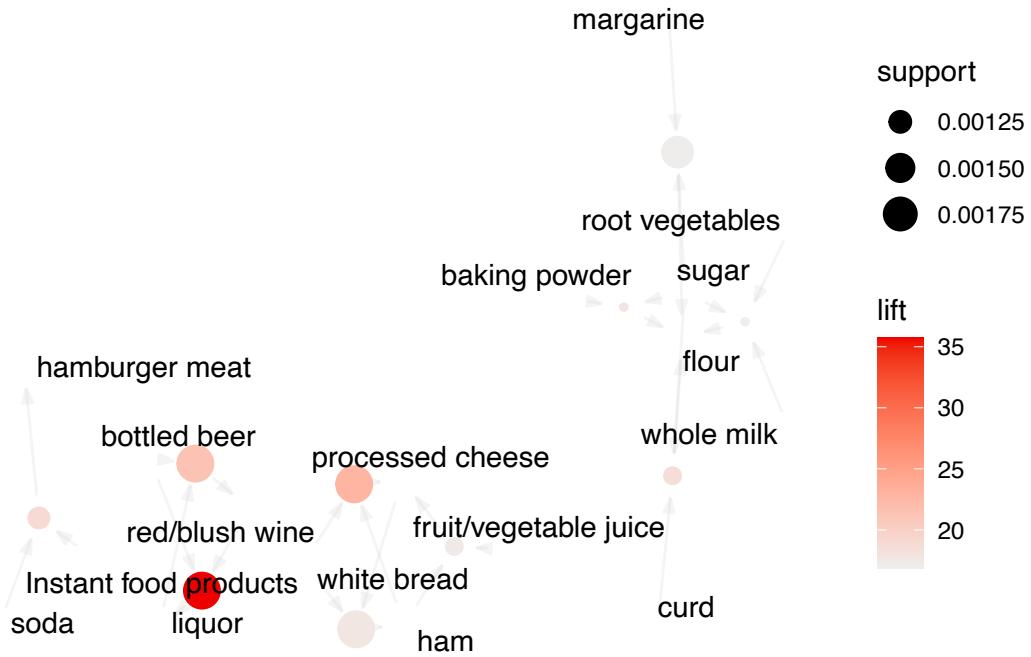
Baking Combination: Several rules indicate that customers purchasing certain baking ingredients like {curd, sugar} and {baking powder, sugar} are also likely to buy {flour}. The lifts of these rules range from 16.92 to 18.61, which could inform stores to place these items in close proximity to encourage baking-related purchases.

Meal Combination: The rule {Instant food products, soda} => {hamburger meat} with a lift of nearly 19 shows that instant food products and soda are often purchased along with hamburger meat. This could be useful for stores to bundle these items for promotions or place them near each other to increase sales.

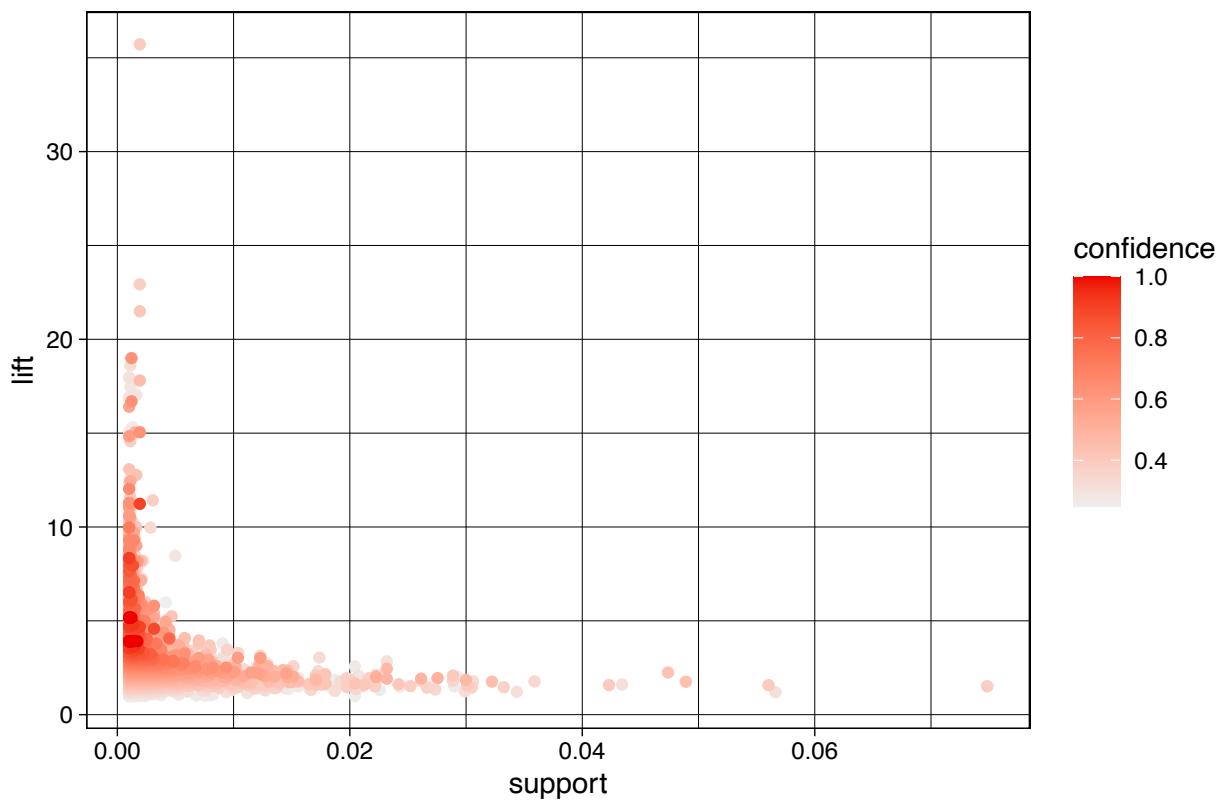
### **Visual Analysis**

The scatter plot visualizes the strength and reliability of these associations. The plot shows a clustering of rules with higher lift values at the lower levels of support, which is expected since high-lift rules are often less common.

The two-key plot shows that as the size of the itemset increases, the support generally decreases, which is again typical in market basket analysis.



Scatter plot for 17391 rules



## Scatter plot for 17391 rules

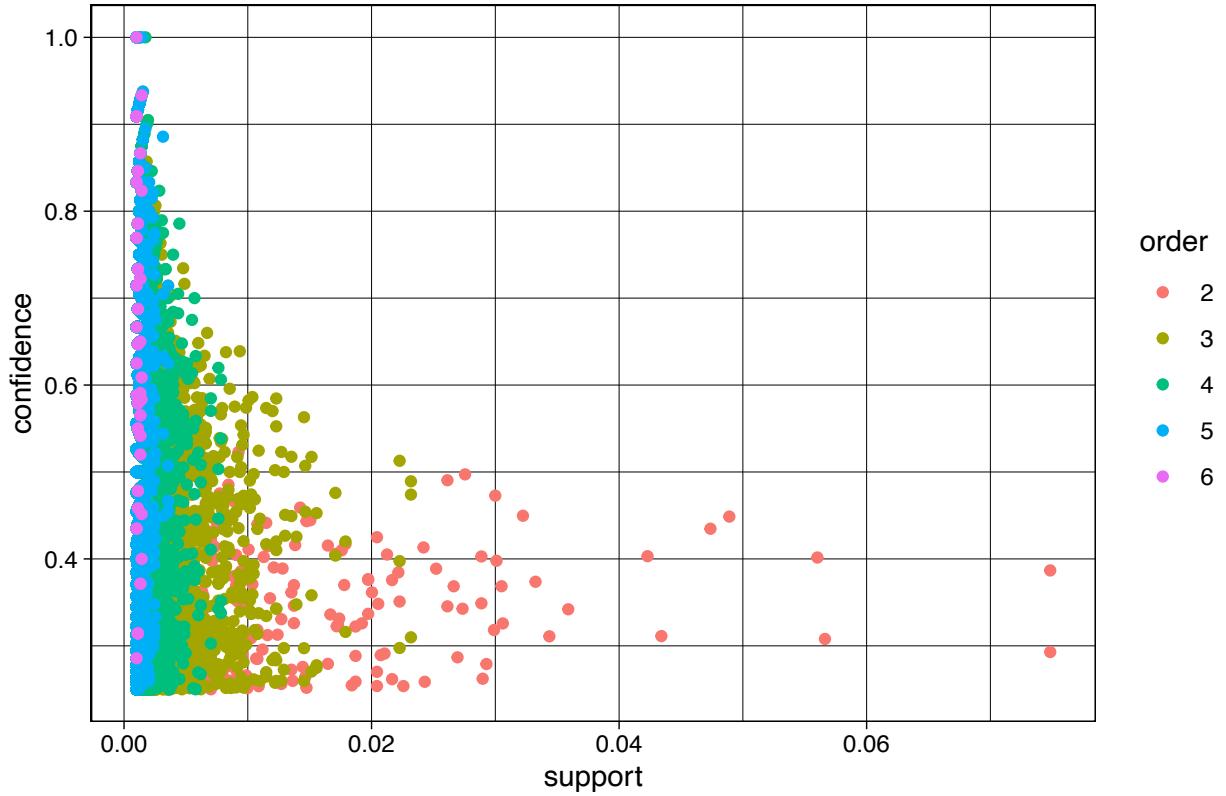


Table 2: Top 10 Rules by Lift

	rules	support	confidence	coverage	lift	count
380	{bottled beer,red/blend wine} => {liquor}	0.0019319	0.3958333	0.0048805	35.71579	19
1147	{ham,white bread} => {processed cheese}	0.0019319	0.3800000	0.0050839	22.92822	19
379	{bottled beer,liquor} => {red/blend wine}	0.0019319	0.4130435	0.0046772	21.49356	19
442	{Instant food products,soda} => {hamburger meat}	0.0012201	0.6315789	0.0019319	18.99565	12
1585	{curd,sugar} => {flour}	0.0011185	0.3235294	0.0034570	18.60767	11
1497	{baking powder,sugar} => {flour}	0.0010168	0.3125000	0.0032537	17.97332	10
1146	{processed cheese,white bread} => {ham}	0.0019319	0.4634146	0.0041688	17.80345	19
1150	{fruit/vegetable juice,ham} => {processed cheese}	0.0011185	0.2894737	0.0038638	17.46610	11
1588	{margarine,sugar} => {flour}	0.0016268	0.2962963	0.0054906	17.04137	16
7495	{root vegetables,sugar,whole milk} => {flour}	0.0010168	0.2941176	0.0034570	16.91606	10

## Conclusions

The identified rules make sense in the context of shopping behavior. Items that are frequently purchased together, such as combinations of alcoholic beverages or components for baking, can be targeted for cross-promotions or placed together in-store to increase basket size. The rules with high lift, especially those involving complementary items, validate the utility of the association rule mining in uncovering shopping basket patterns.

To find the neural network that can classify the images as accurately as possible, we use an 80/20 train test split through the use of PyTorch in a Jupyter notebook for this problem.

In [30]: `!pip install torch torchvision`

```
Requirement already satisfied: torch in d:\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: torchvision in d:\anaconda3\lib\site-packages (0.17.2
+cpu)
Requirement already satisfied: filelock in d:\anaconda3\lib\site-packages (from torc
h) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in d:\anaconda3\lib\site-pac
kages (from torch) (4.9.0)
Requirement already satisfied: sympy in d:\anaconda3\lib\site-packages (from torch)
(1.12)
Requirement already satisfied: networkx in d:\anaconda3\lib\site-packages (from torc
h) (3.1)
Requirement already satisfied: jinja2 in d:\anaconda3\lib\site-packages (from torch)
(3.1.3)
Requirement already satisfied: fsspec in d:\anaconda3\lib\site-packages (from torch)
(2023.10.0)
Requirement already satisfied: numpy in d:\anaconda3\lib\site-packages (from torchvi
sion) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in d:\anaconda3\lib\site-pacak
ges (from torchvision) (10.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\anaconda3\lib\site-packages (fr
om jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in d:\anaconda3\lib\site-packages (from
sympy->torch) (1.3.0)
```

Firstly, set necessary imports, and set the batch size for training and testing to 4, define a transformation to apply to the images of 32\*32

In [41]: `# Necessary Imports`

```
import torch
import torchvision
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import random_split, DataLoader
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In [45]: `# Set the directory where your data is stored`  
`data_dir = r'C:\Users\Siris\Desktop\UT Austin\DATA MINING\HW4\EuroSAT_RGB'`  
`# Set the batch size for training and testing`  
`batch_size = 4`

```
# Define a transformation to apply to the images
transform = transforms.Compose(
    [transforms.Resize((32, 32)), # Resize images to 32x32
     transforms.ToTensor(), # Convert image to PyTorch Tensor data type
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]) # Normalize the image
```

Set 2 layers of convolutions, in 10 epoches:

```
In [52]: # Load the dataset
dataset = ImageFolder(root=data_dir, transform=transform)

# Creating Train and Test Splits
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = random_split(dataset, [train_size, test_size])

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# Define the Neural Network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(8 * 8 * 64, 512) # Corrected the flattening size
        self.fc2 = nn.Linear(512, 128)
        self.fc3 = nn.Linear(128, len(dataset.classes))
        self.dropout = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.dropout(F.relu(self.fc2(x)))
        x = self.fc3(x)
        return x

net = Net()

# Train the Network
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

for epoch in range(10): # Loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

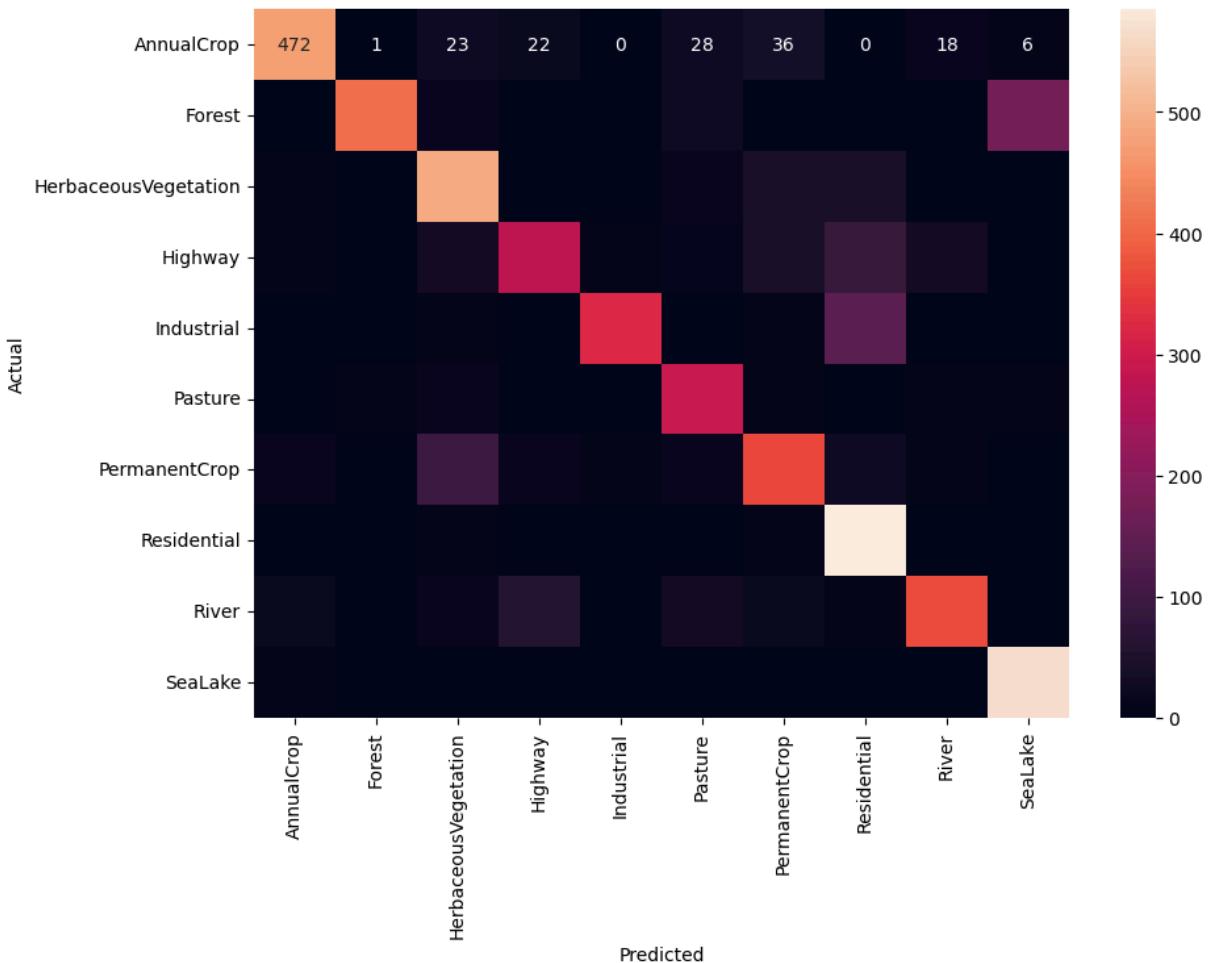
```
running_loss += loss.item()
print(f'Epoch {epoch + 1}, Loss: {running_loss / len(train_loader)}')

# Evaluate the Model
correct = 0
total = 0
predictions = []
with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        predictions.append((labels, predicted))

print(f'Accuracy of the network on the {test_size} test images: {100 * correct // t

# Generate a confusion matrix
y_true = [label.item() for batch in predictions for label in batch[0]]
y_pred = [pred.item() for batch in predictions for pred in batch[1]]
conf_mat = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=dataset.classes, yticklabels=dataset.classes)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Epoch 1, Loss: 1.6917541412981572  
Epoch 2, Loss: 1.1620475119103988  
Epoch 3, Loss: 0.9357443873409836  
Epoch 4, Loss: 0.8050874051512908  
Epoch 5, Loss: 0.6917304687366683  
Epoch 6, Loss: 0.6051972302721589  
Epoch 7, Loss: 0.5476186769722564  
Epoch 8, Loss: 0.4774237452176833  
Epoch 9, Loss: 0.41734543599355134  
Epoch 10, Loss: 0.363583017053849  
Accuracy of the network on the 5400 test images: 76%



The accuracy is 76%.

Set 3 layers of convolutions, in 10 epoches:

```
In [53]: # Define the Neural Network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(2048, 512)
        self.fc2 = nn.Linear(512, 128)
        self.fc3 = nn.Linear(128, len(dataset.classes))
        self.dropout = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
```

```

        return x
net = Net()

```

```

In [54]: # Train the Network
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

for epoch in range(10): # Loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print(f'Epoch {epoch + 1}, Loss: {running_loss / len(train_loader)}')

# Evaluate the Model
correct = 0
total = 0
predictions = []
with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        predictions.append((labels, predicted))

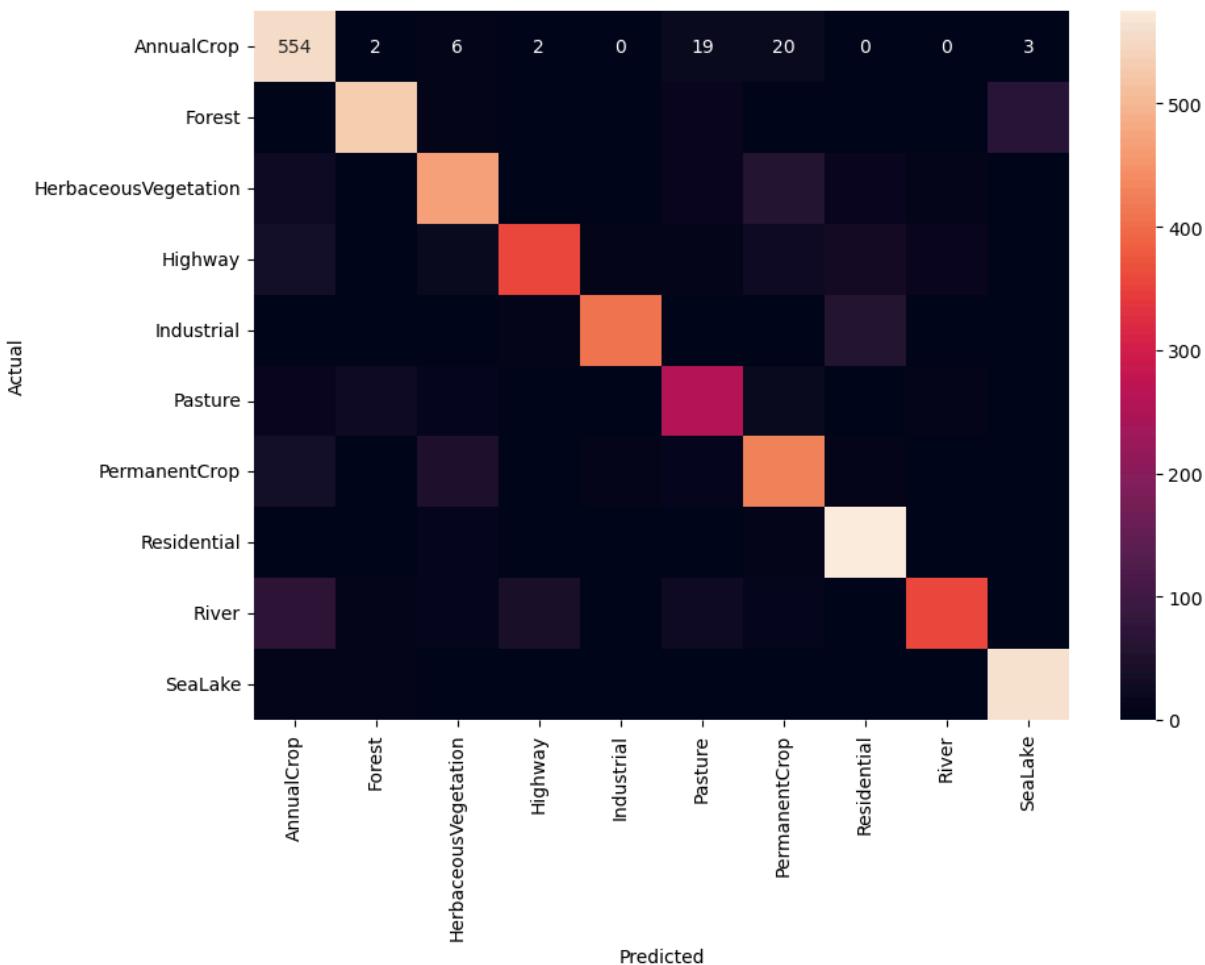
print(f'Accuracy of the network on the {test_size} test images: {100 * correct // t

# Generate a confusion matrix
y_true = [label.item() for batch in predictions for label in batch[0]]
y_pred = [pred.item() for batch in predictions for pred in batch[1]]
conf_mat = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=dataset.classes, yticklabels=
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

```

Epoch 1, Loss: 1.742134058525165  
 Epoch 2, Loss: 1.0924803275732256  
 Epoch 3, Loss: 0.8648180160774953  
 Epoch 4, Loss: 0.738349261645883  
 Epoch 5, Loss: 0.6402427593135723  
 Epoch 6, Loss: 0.5415999864476423  
 Epoch 7, Loss: 0.47394506097965255  
 Epoch 8, Loss: 0.4077014253906371  
 Epoch 9, Loss: 0.3563752009569025  
 Epoch 10, Loss: 0.3094131575263115

Accuracy of the network on the 5400 test images: 83%



The accuracy is now improved to 83%. The model's training process over 10 epochs shows a consistent decrease in loss, which is a good indicator of learning. Starting with a loss of approximately 1.74 in the first epoch, it ended with a loss of about 0.31 in the tenth epoch. This reduction in loss suggests that the network's weights and biases are being adjusted to minimize the error in predictions over time. Now run a sample test:

```
In [55]: net.eval()

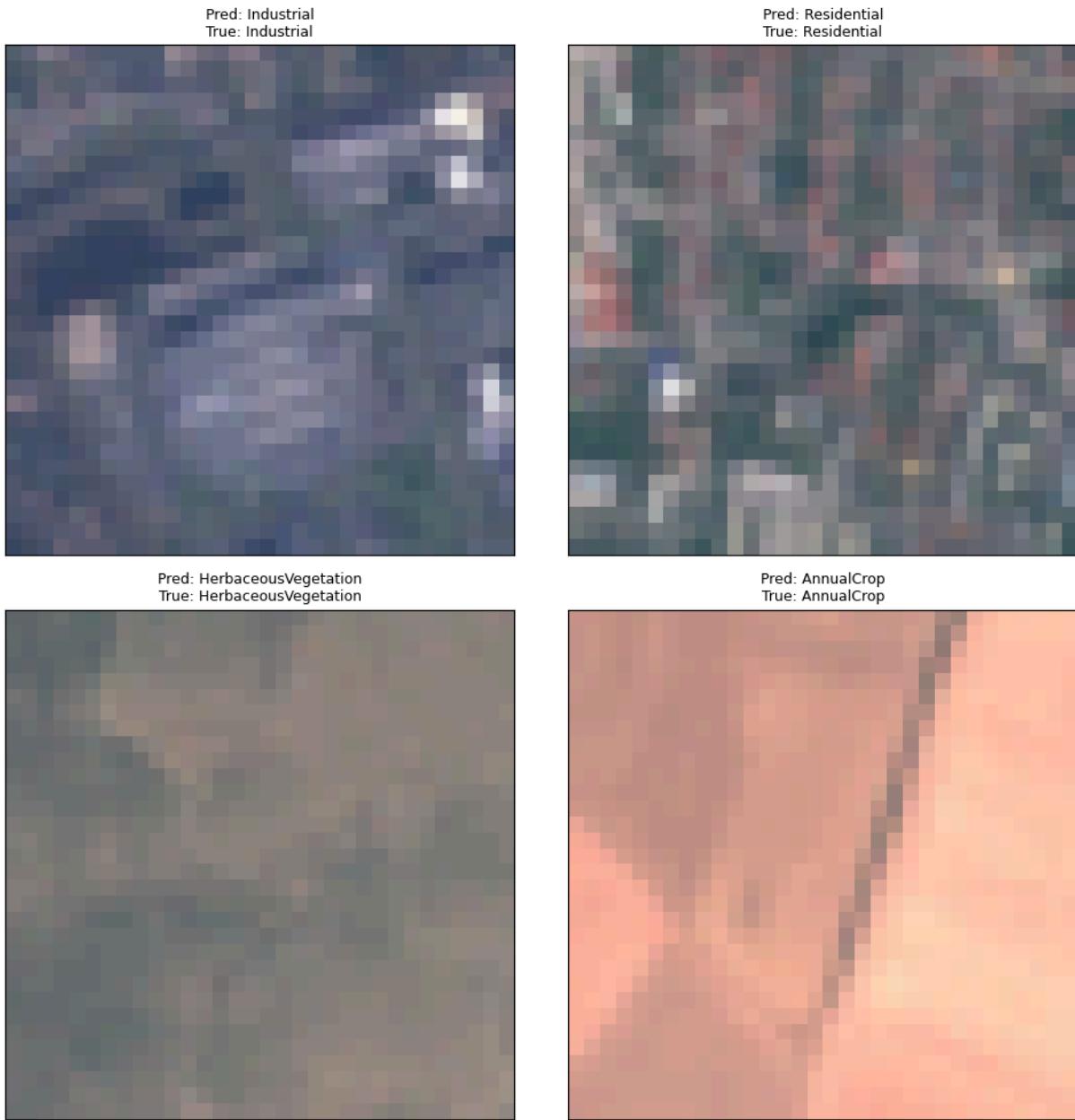
# No need to track gradients here
with torch.no_grad():
    # Get some random test images and Labels
    dataiter = iter(test_loader)

    # Corrected line with the next function
    images, true_labels = next(dataiter)

    # Run your model to get predictions
    outputs = net(images)

    # The predicted class is the one with the highest output score
    _, predicted_labels = torch.max(outputs, 1)
```

```
# Now call the function to plot the images with labels
plot_images(images, true_labels, predicted_labels, dataset.classes)
```



The network has achieved a test accuracy of 73%. While not perfect, this indicates that your model has learned significant features from the training set. The sample predictions shown reveal the model's output on individual test set images. Each sub-image is annotated with the predicted class and the actual class. In the given images, most predictions match the true labels, which is consistent with a model that has a decent level of learning to capture.