Izaak King

Professor Eleish

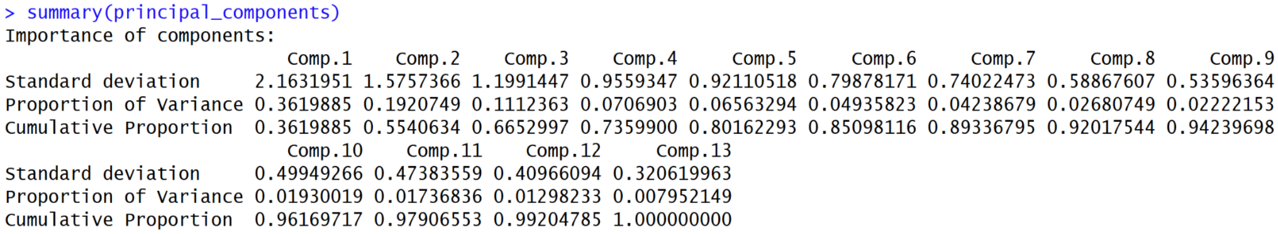
Data Science

October 24, 2025

"Lab 4"

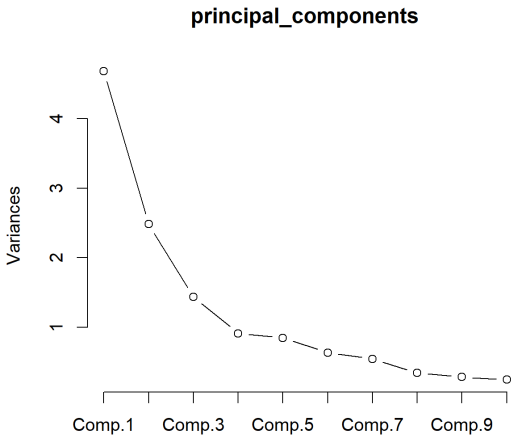
For this lab, I performed Principal Component Analysis on the provided wine dataset. I start with computing the Principal Components, plotting the dataset using the first and second PCs. Upon initial trials I noticed the Proline variable was much greater than all other attributes, so the data was scaled accordingly. Following this, I continued my analysis

I. Summary of Principal Components:

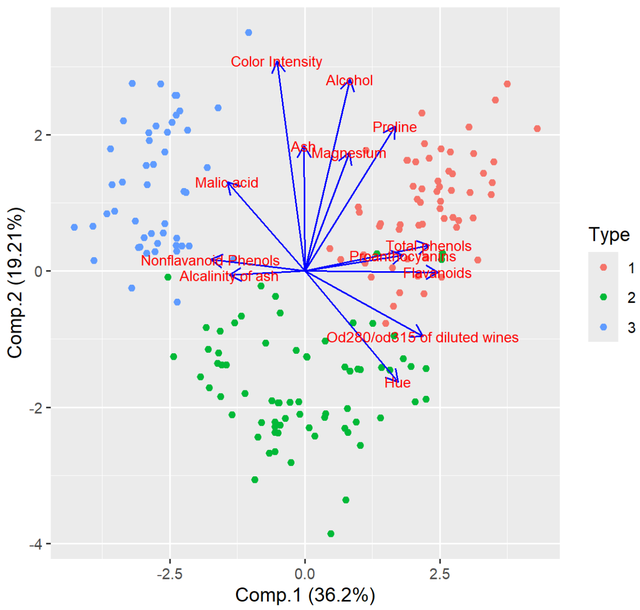


```
> summary(principal_components)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5     Comp.6     Comp.7     Comp.8     Comp.9
Standard deviation     2.1631951 1.5757366 1.1991447 0.9559347 0.92110518 0.79878171 0.74022473 0.58867607 0.53596364
Proportion of Variance 0.3619885 0.1920749 0.1112363 0.0706903 0.06563294 0.04935823 0.04238679 0.02680749 0.02222153
Cumulative Proportion  0.3619885 0.5540634 0.6652997 0.7359900 0.80162293 0.85098116 0.89336795 0.92017544 0.94239698
                          Comp.10    Comp.11    Comp.12     Comp.13
Standard deviation     0.49949266 0.47383559 0.40966094 0.320619963
Proportion of Variance 0.01930019 0.01736836 0.01298233 0.007952149
Cumulative Proportion  0.96169717 0.97906553 0.99204785 1.000000000
```

II. Line plot of variance among components:



III. Plot of the dataset along the first two components:



To identify what variables contribute most to the first and second Principal Components, we can look at the ‘loadings’ attribute for our Principal Component calculation. Particularly we will look at the first two components.

I. Principal Component loading values:

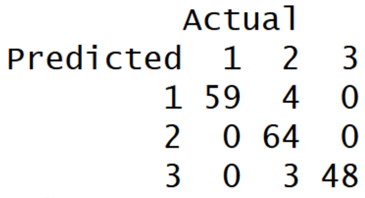
```
Loadings:
                             Comp.1 Comp.2
Alcohol                       0.144  0.484
Malic acid                   -0.245  0.225
Ash                                  0.316
Alcalinity of ash            -0.239
Magnesium                     0.142  0.300
Total phenols                 0.395
Flavanoids                    0.423
Nonflavanoid Phenols         -0.299
Proanthocyanins               0.313
Color Intensity                      0.530
Hue                           0.297 -0.279
Od280/od315 of diluted wines  0.376 -0.164
Proline                       0.287  0.365
```

Based on these loading values, we see the variables that contribute the most are as follows:

- Component 1: Flavanoids, Total phenols, od280/od315 of diluted wines
- Component 2: Color Intensity, Alcohol, Proline

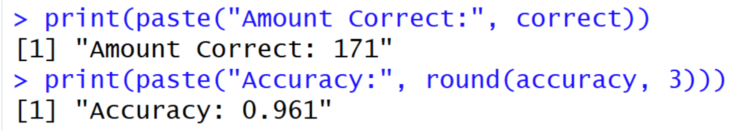
Now I will train a classifying k-NN model to predict wine type using all of the variables in the original dataset. We will use k = 3, for three clusters, each for the different type of wine. We will look at this models performance using the confusion matrix, calculating accuracy.

I. Confusion matrix for k-NN model based on all variables:



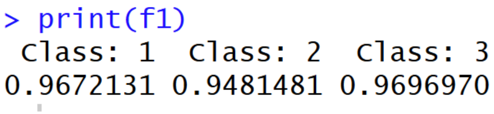
```
         Actual
Predicted  1  2  3
        1 59  4  0
        2  0 64  0
        3  0  3 48
```

II. Accuracy of k-NN model based on all variables:



```
> print(paste("Amount Correct:", correct))
[1] "Amount Correct: 171"
> print(paste("Accuracy:", round(accuracy, 3)))
[1] "Accuracy: 0.961"
```

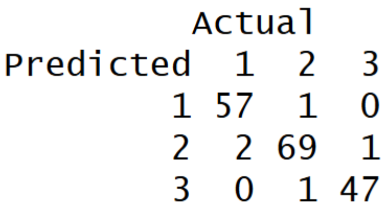
III. F1 Scores of k-NN model based on all variables:



```
> print(f1)
 Class: 1  Class: 2  Class: 3 
0.9672131 0.9481481 0.9696970 
```

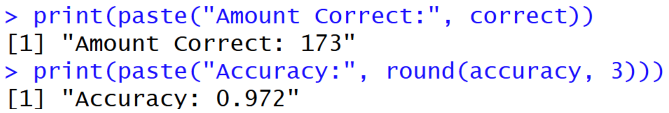
This model does a very good job of predicting wine type, with an accuracy of about 96.1%. However, this model relies on 13 variable attributes, which can have a potentially high computational cost. Thus, we will attempt to train a model with only 2 dimensions, using the first 2 PCs computed before. Specifically we will train these models using first two scores in the princomp function's return object.

I. Confusion matrix for k-NN model based on first 2 PCs:



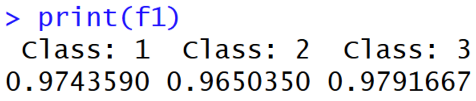
```
         Actual
Predicted  1  2  3
        1 57  1  0
        2  2 69  1
        3  0  1 47
```

II. Accuracy of k-NN model based on first 2 PCs:



```
> print(paste("Amount Correct:", correct))
[1] "Amount Correct: 173"
> print(paste("Accuracy:", round(accuracy, 3)))
[1] "Accuracy: 0.972"
```

III. F1 Scores of k-NN model based on first 2 PCs:



```
> print(f1)
 Class: 1  Class: 2  Class: 3 
0.9743590 0.9650350 0.9791667 
```

Thus, based on the confusion matrices and accuracy metrics for both models, we see the k-NN model based on the first 2 PCs actually performs better than the model using all 13 original variables. The PCA model has a higher F1 Score across all three classes than the original model. To build onto this, the PCA model has accuracy of about 97.2%

compared to the original models 96.1%. Though the accuracy is relatively similar, this is significant considering the reduction in dimensionality in these models. Overall, these k-NN models illustrate the power of PCA, and how it can be used to effectively reduce dimensionality while maintaining the nature and trends of the data.