## *Preliminary treatment:*

The previous data has been standardized to prevent bias and all non- numerical and missing values discarded. The filtering is 0.4 or 0.7 strength and 0 or 1 for italic. Further to increase accuracy and improve our model, we have chosen larger datasets listed below.

Before Filtering:

*BODONI – 3964 instances and 403 features*

*REFERENCE – 4652 instances and 403 features*

*ROMAN – 4776 instances and 403 features*

After Filtering:

*BODONI – 991 instances and 403 features*

*REFERENCE –1163 instances and 403 features*

*ROMAN –1194 instances and 403 features*

We then define the three classes (CL1, CL2. And CL3) to each dataset and unionize them into a full dataset. DATA then contains 3348 observations / instances and 403 features.

## *Computing the correlation matrix:*

We compute the correlation matrix of the whole dataset. Finding the correlation matrix will show us the correlation coefficients between the 400 features. A correlation matrix only calculates the correlation between numeric features, so we took out all non-numeric data, leaving us with a 400 x 400 array. First, we must look at the correlation matrix pixels to visualize if the features are correlated with each other. If they have high correlation it can indicate a redundancy in features, where they have potential to have dimension reduction. The cor() function is used below. Below is a small portion of the correlation matrix values, full values are in separate excel file on sheet CorrMatrix.

| | r0c0 | r0c1 | r0c2 | r0c3 | r0c4 | r0c5 | r0c6 | r0c7 | r0c8 |
|---|---|---|---|---|---|---|---|---|---|
| r0c0 | 1 | 0.89504562 | 0.75112749 | 0.60481181 | 0.4337054 | 0.31538981 | 0.20107913 | 0.11475737 | 0.0693473 |
| r0c1 | 0.89504562 | 1 | 0.89491245 | 0.72915431 | 0.52850123 | 0.37315448 | 0.23374816 | 0.13032083 | 0.07917103 |
| r0c2 | 0.75112749 | 0.89491245 | 1 | 0.88731809 | 0.67585315 | 0.48259681 | 0.30086064 | 0.16546258 | 0.08458307 |
| r0c3 | 0.60481181 | 0.72915431 | 0.88731809 | 1 | 0.85148588 | 0.63341493 | 0.39953748 | 0.21451667 | 0.08716931 |
| r0c4 | 0.4337054 | 0.52850123 | 0.67585315 | 0.85148588 | 1 | 0.8535846 | 0.61085039 | 0.39036786 | 0.19557298 |
| r0c5 | 0.31538981 | 0.37315448 | 0.48259681 | 0.63341493 | 0.8535846 | 1 | 0.84602812 | 0.60959895 | 0.37154847 |
| r0c6 | 0.20107913 | 0.23374816 | 0.30086064 | 0.39953748 | 0.61085039 | 0.84602812 | 1 | 0.85526655 | 0.60907632 |
| r0c7 | 0.11475737 | 0.13032083 | 0.16546258 | 0.21451667 | 0.39036786 | 0.60959895 | 0.85526655 | 1 | 0.8390825 |
| r0c8 | 0.0693473 | 0.07917103 | 0.08458307 | 0.08716931 | 0.19557298 | 0.37154847 | 0.60907632 | 0.8390825 | 1 |

## *Computing eigenvalues:*

Finding the eigenvalues are important as they are the variance of the principal components. Eigenvalues that are greater than 1 indicates that the principal components account for more variance than the original variables in the standardized data. The highest eigenvalues will comprise of the highest variance or more important features. To do this we will use the eigen() function in R, then $values to only present the eigenvalues. Below are the top 10 eigen values to see full eigen values go to attached excel file on sheet labeled “eigenvalues”. The full plot with all 400 values is also located on this excel sheet.

| r | Eigenvalues |
|---|---|
| 1 | 71.7589931 |
| 2 | 43.1150182 |
| 3 | 24.0136042 |
| 4 | 16.532113 |
| 5 | 14.553171 |
| 6 | 13.0683178 |
| 7 | 11.6020053 |
| 8 | 10.084509 |
| 9 | 8.2341475 |
| 10 | 7.69520475 |

*Computing the transpose of the matrix of eigenvectors:*

If we apply the PCA to data matrices whose number of variables (p) is greater than the number of objects (n), the calculation for the PCA will take a long time. This is because the PCA is based on the solution of the eigenvalue problem for the covariance matrix. The dimension of the covariance matrix is p x p, which will result in big matrices if there are a lot of variables. In order to make PCA calculation faster, we can transpose the data matrix before performing PCA. The covariance matrix of the transposed matrix will be a lot smaller. This makes the calculation process faster. First we must extract the eigenvectors, for which we will use the same eigen() function. Then only extract the vectors by using the $vectors function. Lastly, we will transpose the eigenvectors by using the t() function. Values are on excel document labeled eigenvector. Below is a small portion of the eigen vectors, to see the full values please open excel document and click sheet labeled “eigenvectors”.

| | V1 | V2 | V3 | V4 | V5 |
|---|---|---|---|---|---|
| 1 | 0.02202268 | 0.01708122 | 0.07738714 | 0.09203141 | -0.0654447 |
| 2 | 0.02828704 | 0.01122617 | 0.07064992 | 0.10433999 | -0.0748416 |
| 3 | 0.03216638 | 0.00862829 | 0.05275404 | 0.11327132 | -0.0780442 |
| 4 | 0.03117504 | 0.00922629 | 0.03887637 | 0.10964644 | -0.0805079 |
| 5 | 0.03452489 | 0.00632411 | 0.02831807 | 0.08778864 | -0.0831908 |

## *Plot Eigenvalues versus principal component:*

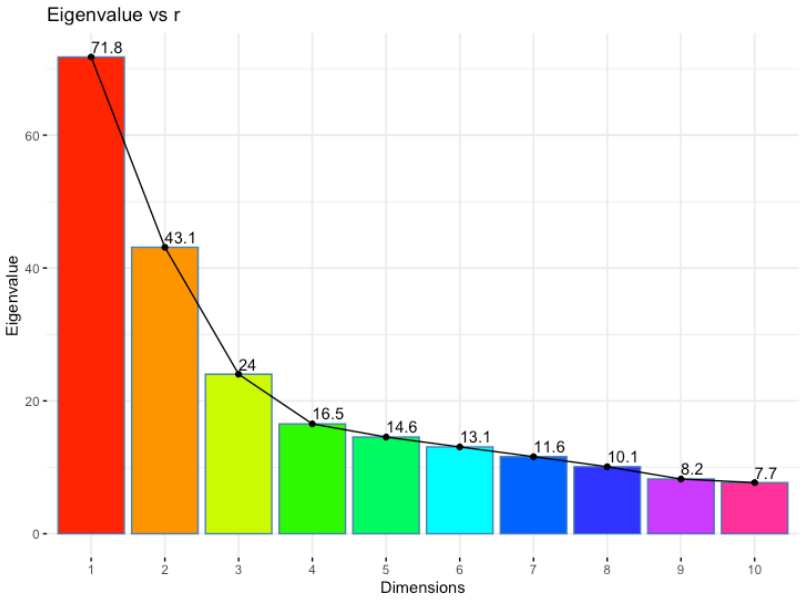



Figure 1 – A plot showing the eigen values with the dimensions. The x – axis is the number of dimensions and the y - axis is the eigen values. The plot shows the top 10 eigenvalues and their respects values at the top of the plot.

Eigen values with less than one should be discarded because the eigen values of 1 accounts for as much variance as one variable. Thus, only factors that explain at least the same amount of variance should be kept. Based on this plot we can see that a majority of the higher eigen values are in the first few principal components. We can see a downward trend as the principal components decreases. This is an effective way to conclude which components we should extract from. Since a cut off of 1 could result in more factors that required, it is better to compute and extract by using the total variance.

## *Plot Percentage of variance explained:*

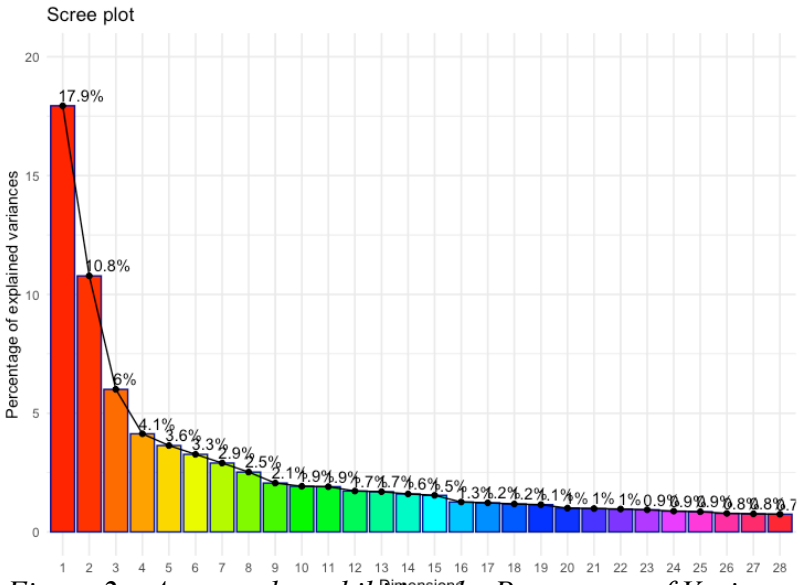



*Figure 2 – A scree plot exhibiting the Percentage of Variance Explained.*

The x – axis is the number of dimensions, and the y – axis is the percentage of variance. The scree plot shows the dimensions and its corresponding variance with the values at the top. Further, the scree plot only presents the top 28 dimensions, the plot continues with the decreasing trend to 400 dimensions.

The scree plot will show us the total amount of variability of the target explained by each factor. This will show us the meaningful dimensions we want to extract. We can see that majority of the variance is explained in the first few dimensions, which is expected. We also can see that there is a downward trend of variance as the dimensions increase. The plot will not show which dimensions we should stop at exactly to achieve 95% variance, so we must calculate the PVE.

*Compute the principal components for 95% variance:*

To compute the proportion of variance explained by each principal component, we simply divide the variance explained by each principal component by the total variance explained by all 95% principal components. We use the prcomp function to do this. The get_eigen function will provide a table exhibiting the eigenvalues, the percent variance of the corresponding dimensions and the cumulative variance of the dimensions.

| PC | Eigenvalue | Variance % | Cumulative Variance % |
|---|---|---|---|
| Dim.1 | 71.758993 | 17.939748 | 17.93975 |
| Dim.2 | 43.115018 | 10.778755 | 28.7185 |
| Dim.3 | 24.013604 | 6.0034011 | 34.7219 |
| Dim.4 | 16.532113 | 4.1330283 | 38.85493 |
| Dim.5 | 14.553171 | 3.6382927 | 42.49322 |
| Dim.106 | 0.2996598 | 0.0749149 | 94.82021 |
| Dim.107 | 0.2934484 | 0.0733621 | 94.89357 |
| Dim.108 | 0.2922575 | 0.0730644 | 94.96664 |

The first principal components are 17.93%, the second principal component is 10.77% and so forth. According to the scree plot and the table above, the first 108 principal components accounts for about 95% of the explained variance.

*Compute the principal components Y1(n) …Yr(n) for each case:*

We first obtain the eigenvectors of the correlation matrix and transpose it. We call this 400x400 matrix of the transposed eigenvector values *NewFeatures*. We multiply *NewFeatures* by the transpose of the original standardized features and call it *pcad*. The rows of the resulting matrix *pcad* represent the 400 principal components. Next, we subset the first 108 rows of the matrix pcad and then transpose it, calling it *pcadataset.* We do this because the first 108 rows represent the first 108 principal components, and as we see from the table above, the first 108 principal components explain 95% of the variance. Each row is a linear combination of the 400 original features. We also add the label vector to *pcadataset,* making it a 3348x109 matrix. Below is a portion of the computed Y(n) for each case. To see the full values, refer to Excel sheet labeled PC.

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.5511269 | 1.19884697 | -14.784759 | 3.74594232 | -2.1773112 | 2.80351752 | -2.4532746 | -0.957237 | 1.01339822 | 2.76648836 |
| 2 | 11.6098172 | 0.98836738 | -14.613826 | 2.94614591 | -2.728333 | 1.60569534 | -0.223909 | -1.1922183 | 3.23377641 | -2.0224166 |
| 3 | -2.2381129 | -2.4304828 | -4.475051 | -2.5693029 | 7.63692726 | 1.34794184 | 2.56584797 | 0.39978221 | -0.3927364 | 1.53776227 |
| 4 | -3.2377431 | -1.9588758 | 1.23354818 | -0.1603586 | -4.5287113 | -1.2779065 | 0.08068129 | 1.1349389 | -0.8566504 | 1.96112126 |
| 5 | -3.3189016 | -2.0414673 | 0.87001302 | 0.31746832 | -4.2141286 | -1.8080736 | 0.47806271 | 0.73408534 | 0.00150047 | 0.27600714 |
| 6 | -2.5641969 | -4.2940421 | -1.9458968 | 0.278473 | 2.05242607 | 1.68775072 | 3.16383514 | 1.98792412 | -1.2655175 | 3.94188565 |
| 7 | -0.7740936 | -1.8315662 | 3.04801517 | 0.64161734 | -0.2108567 | 1.25995063 | 0.5123143 | 0.81061342 | -0.0390435 | 2.18337922 |
| 8 | -2.8803829 | -12.848484 | 7.94840094 | -1.9272018 | -0.7666012 | -1.1129624 | 0.92508157 | 0.65647868 | -2.9782401 | -1.3458053 |
| 9 | -2.3949051 | -3.8582231 | 5.26312905 | -0.6136375 | 0.2572437 | -2.687019 | -1.6966461 | 0.19859713 | -0.3385678 | 1.15065842 |
| 10 | -7.5969539 | -3.3469573 | -1.6764639 | 0.88462766 | 3.11142927 | 5.44983292 | 0.63335667 | 0.92221562 | 3.15576851 | -2.7659325 |

*Applying KNN on new features made by PCA.*

Principal Component Analysis uses the sum of percentage of variation in the data set to see how strongly correlated they are with one another. It uses eigenvalues and eigenvectors to create the number of dimensions in the dataset. Using a PCA requires all numerical features, as it is using its correlation values, eigenvalues, and eigenvectors. Using PCA reduction will increase accuracy on KNN by decreasing the number of features (high dimensionality to risk over fitting). PCA uses an algorithm that linearly transforms dimensional input where r < p to discard/ minimize the features with less variance. After the features are discarded, we will run a KNN algorithm on the 108 new features using the k = 5. Below is the confusion matrix accuracy showing the train and test set accuracy of before PCA dimension reduction and after dimension reduction.

*Train total = 0.8827045*

| | | TRAIN PREDICTION (normal) | | |
|---|---|---|---|---|
| | **Test K = 5** | **CL1** | **CL2** | **CL3** |
| TRUE | **CL1** | 84% | 10% | 7% |
| | **CL2** | 4% | 94% | 1% |
| | **CL3** | 5% | 8% | 87% |

*Test total= 0.8196721*

| | | TEST PREDICTION (normal) | | |
|---|---|---|---|---|
| | **Test K = 5** | **CL1** | **CL2** | **CL3** |
| TRUE | **CL1** | 65% | 12% | 22% |
| | **CL2** | 7% | 91% | 2% |
| | **CL3** | 9% | 10% | 81% |

*Train total = 0.8827065*

| | | TRAIN PREDICTION (PCA) | | |
|---|---|---|---|---|
| | **Test K = 5** | **CL1** | **CL2** | **CL3** |
| TRUE | **CL1** | 83% | 11% | 6% |
| | **CL2** | 4% | 95% | 1% |
| | **CL3** | 5% | 8% | 87% |

*Test total= 0.8360656*

| | | TEST PREDICTION (PCA) | | |
|---|---|---|---|---|
| | **Test K = 5** | **CL1** | **CL2** | **CL3** |
| TRUE | **CL1** | 79% | 13% | 7% |
| | **CL2** | 6% | 94% | 1% |
| | **CL3** | 9% | 11% | 80% |

TRAIN SET

The overall accuracy did not influence the train set. The percentages are about the same.

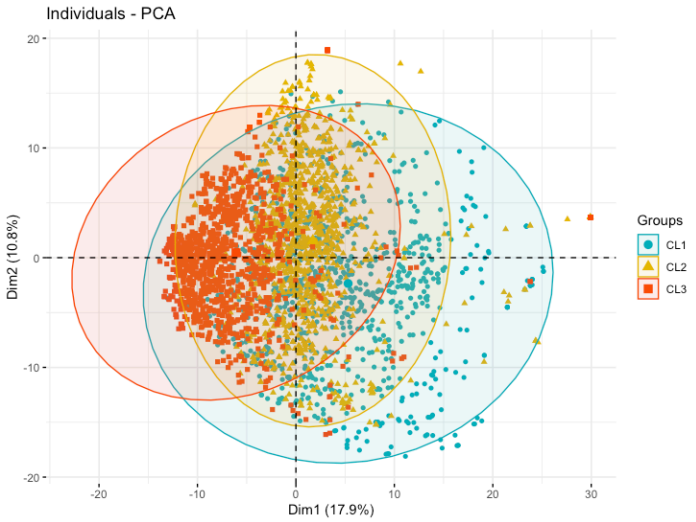
TEST SET

Here we can see that the overall accuracy increased in the test set by about 2%. From the test set confusion matrix before dimension reduction, we can see that about 34% of CL1 is classified as false negative. Comparatively to the PCA reduction it is only 20%. Further, we can see the false positive also decreased by 1% in CL1. In CL2 we can see that the accuracy of CL2 increased as well. But the accuracy for CL3 decreased by 1%.

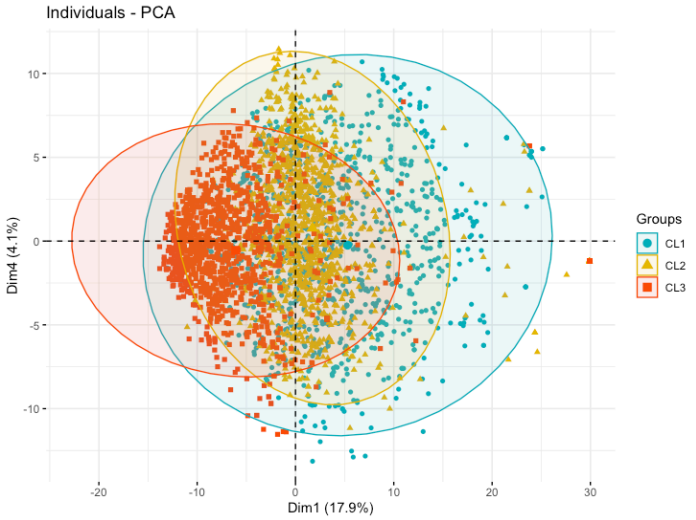
Overall, we can see that the dimension reduction increases accuracy for the test set. It is possible running an LDA will also improve the overall accuracy of the KNN prediction model.

*Compute 6 color graphic scatterplot displays of your 3 classes in the 6 planes and its interpretations*

The function fviz_pca_ind() is used to produce the graph of individuals based on the results of PCA (font.pca). The scatter plots/score plots show the projection of the data onto the span of the principal components. We create 6 score plots displaying 3 classes (CL1, CL2, CL3) for each pair combination of principal components. These plots can be used to detect clusters, outliers, and trends. Groupings of data on the plot may indicate two or more separate distributions in the data.







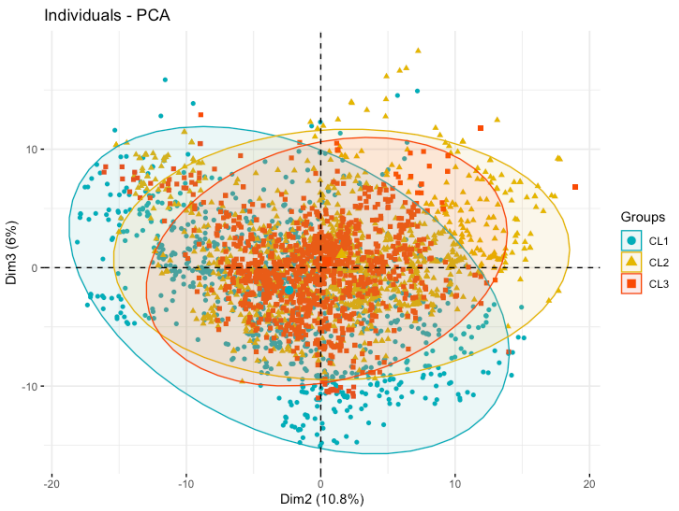




*Figure 3 - Scatter plot of the for each pair of PCs (Y1, Y2) (Y1, Y3) (Y1, Y4) (Y2, Y3) (Y2, Y4) (Y3, Y4)*
The X- axis exhibits the second dimension, and the Y- axis exhibits the first dimension we wish to compare. There are three classes in each plot (CL1, CL2, and CL3) which correspond to different colors. There is a figure legend located on the right side of each graph. Each dot represents the instances.

In figure 3 the score plots show that the 3 classes show the most differentiation along PC1 while there no significant differentiation along PC2, PC3, and PC4. PC1 separates data into 3 clusters. The left cluster contains CL3, the middle cluster contains CL2, and the right cluster contains CL1. The score plots of (Y1, Y2) (Y1, Y3) (Y1, Y4) pairs show differentiation between classes CL1, CL2, and CL3. The score plots (Y2, Y3) (Y2, Y4) (Y3, Y4) pairs show no significant variation to differentiate between classes CL1, CL2, and CL3. Since most of the information are in PC1, the plot with PC1 will show the most distinction between classes.