HW-4 K-Means - Fonts Dataset

Jose Rodriguez, Johnny Nino Ladino, Dayana Sosa; University of Houston

Background

This report will cover the questions presented by Dr. Azencott over the fonts dataset from the University of California Irvine Machine Learning Repository. The fonts dataset is made up of 153 files that are in a csv format. Each file represents a font and within each file there are 412 columns with varying number of rows depending on the file chosen. Each row represents a 'case' and each case describes numerically a digitized image of some specific character (letter or digit) type in a certain font. Each image has a 20 by 20 size and therefore 400 pixels which are represented by 400 columns. The three font files named "Courier", "Calibri" and "Times" will be used for this report. The first 12/412 columns describe the font and the other 400 columns represent the gray levels of the 400 pixels of a specific image. For this dataset, font type will be our target variable and the 400 columns describing the pixel gray levels will be our features. The aim of this report will be to implement Kmeans algorithm to cluster the font type. This report will aim to answer all questions presented in Dr. Azencott's homework instructions in a systematic way while analyzing each question and result to have an in depth understanding of this dataset. The underlying code sustaining this report will be created using python and the IDE of Jupyter notebook.

Methodology

The following methodology was used for developing the code and answering the questions in the document

1. Code will be written in python and python libraries such as pandas, numpy, scikit learn, etc will be imported
2. Code and report will abide to the question numbers presented in Dr. Azencott's instructions
3. The report/code will use the Kmeans clustering algorithm to partition the cases into disjoint clusters
4. The report/code will aim obtain the best value of k and analyze the cluster information

Preliminary Treatment of Data

Each of the three chosen font files has the same column structure. To begin the analysis, data cleaning and preparation needs to be done on the dataset. Out of the 12 non-related pixel columns, we choose to keep only three of them. The three columns that will be kept are font, strength, and italic. Table 1 below explains what each of these columns represent. Any row that contained any missing numerical data was discarded from each of the three files.

FEATURE	DESCRIPTION
FONT	Same as file name. Describes what font the dataset belongs to.
STRENGTH	Column lists values either equal to 0.4 or to 0.7: in each row, strength = 0.4 for normal character; strength = 0.7 for bold character
ITALIC	Column lists values either equal to 0 or to 1; in each row, italic = 0 for normal character; italic = 1 for italic character;

Table 1: 3 Column Descriptions

The three files (referred to as classes) were then filtered down to only include rows containing a strength value of 0.4 and italic value of 0. This filtered the dataset to only images of "normal" characters. The "Calibri", "Courier" and "Times" classes will be referred to as CL1, CL2 and CL3 respectively. Their respective sizes are shown in Table 2 below.

CLASS	ROWS	COLUMNS
CL1 (CALIBRI)	4,768	403
CL2 (COURIER)	4,262	403
CL3 (TIMES)	4,805	403
TOTAL	**13,835**	

Table 2: Class Size

A full data set denoted as DATA was created through the union of the three class CL1, CL2 and CL3 and has a size of 13,835 rows. DATA will contain only the 400 features related to the gray level of the pixels to create a feature matrix. The basic machine learning tool covered in Dr. Azencott's class known as Kmeans will be used to attempt an automatic clustering of the 3 classes CL1, CL2 and CL3.

Part 0

The means and standard deviations of DATA are computed to be used to standardize the dataset DATA. We standardize the dataset so values in certain features are not over weighed or under weighed when we are applying our classifier. We compute the mean and standard deviation of each column leaving us with $\hat{\mu}(mean) = \mu_{X1}, \ldots, \mu_{X400}$ and $\hat{\sigma}(standard\ deviation) = \sigma_{X1}, \ldots, \sigma_{X400}$. We then perform the following to rescale DATA:

$$\boldsymbol{SDATA(X_i, X_j) = \frac{DATA(X_i, X_j) - \mu_j}{\sigma_j}}$$

Equation 1: Standardized Matrix Equation

We store the standardized dataset DATA into a separate data frame named SDATA. From SDATA, we compute the correlation matrix, COR, of all 400 features. Table 3 below highlights the first 10 by 10 portion of the COR matrix.

	R0C0	R0C1	R0C2	R0C3	R0C4	R0C5	R0C6	R0C7	R0C8	R0C9
R0C0	1	0.92	0.79	0.63	0.45	0.28	0.17	0.08	0.02	-0.02
R0C1	0.92	1	0.89	0.70	0.51	0.31	0.18	0.08	0.01	-0.03
R0C2	0.79	0.89	1	0.87	0.64	0.42	0.26	0.15	0.06	-0.01
R0C3	0.63	0.70	0.87	1	0.84	0.60	0.40	0.25	0.12	0.03
R0C4	0.45	0.51	0.64	0.84	1	0.83	0.60	0.38	0.20	0.08
R0C5	0.28	0.31	0.42	0.60	0.83	1	0.83	0.58	0.35	0.19
R0C6	0.17	0.18	0.26	0.40	0.60	0.83	1	0.84	0.58	0.39
R0C7	0.08	0.08	0.15	0.25	0.38	0.58	0.84	1	0.84	0.61
R0C8	0.02	0.01	0.06	0.12	0.20	0.35	0.58	0.84	1	0.86
R0C9	-0.02	-0.03	-0.01	0.03	0.08	0.19	0.39	0.61	0.86	1

Table 3: 10 by 10 subset of Correlation Matrix

Part 1

We will use the 400 standardized features and apply the k-means algorithm for k =1, …, 10, using 50 runs with different initial centroids to partition our cases into k disjoint clusters. We first constructed a plot of the within cluster sum of squares, or SSE, for each number of k clusters in Figure 1. We can see that as the number of clusters increases, SSE decreases. We wish to find the "elbow", where the rate at which SSE decreases appears to slow down, to help determine the best number of clusters for our data. Here, we obtain a kbest value at k=4.

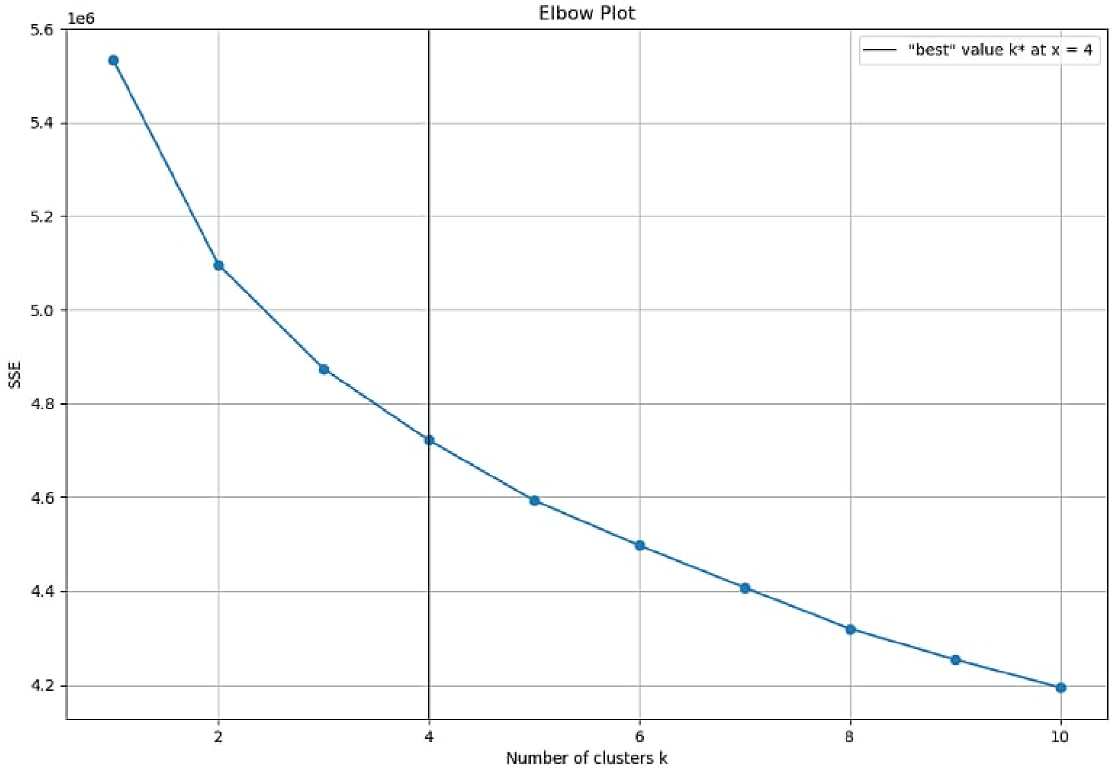

Figure 1: Plot of SSE for Various Number of Clusters (Elbow Plot)

For each k, we compute the reduction of variance, which we denote as Perf(k), using Equation 2. The results for the Perf(k) for each k are displayed in Table 4 below.

$$\boldsymbol{Perf(k) = 1 - \frac{SSE(k)}{SSE(1)}}$$

Equation 2: Calculating Perf(k)

K	PERF(K)
1	0.00
2	0.08
3	0.12
4	0.15
5	0.17
6	0.19
7	0.20
8	0.22
9	0.23
10	0.24

Table 4: Perf(k) for Each k

In Figure 2, we plot Perf(k) vs k and refer to this plot as the "knee plot" due to its shape. As expected, Perf(k) increases as k increases. From Figure 2, we wish to obtain the “knee” where we see the that the rate at which Perf(k) increases appears to slow down. This occurs when k=4, which coincides with our elbow plot, thus we shall denote our kbest value at k=4.

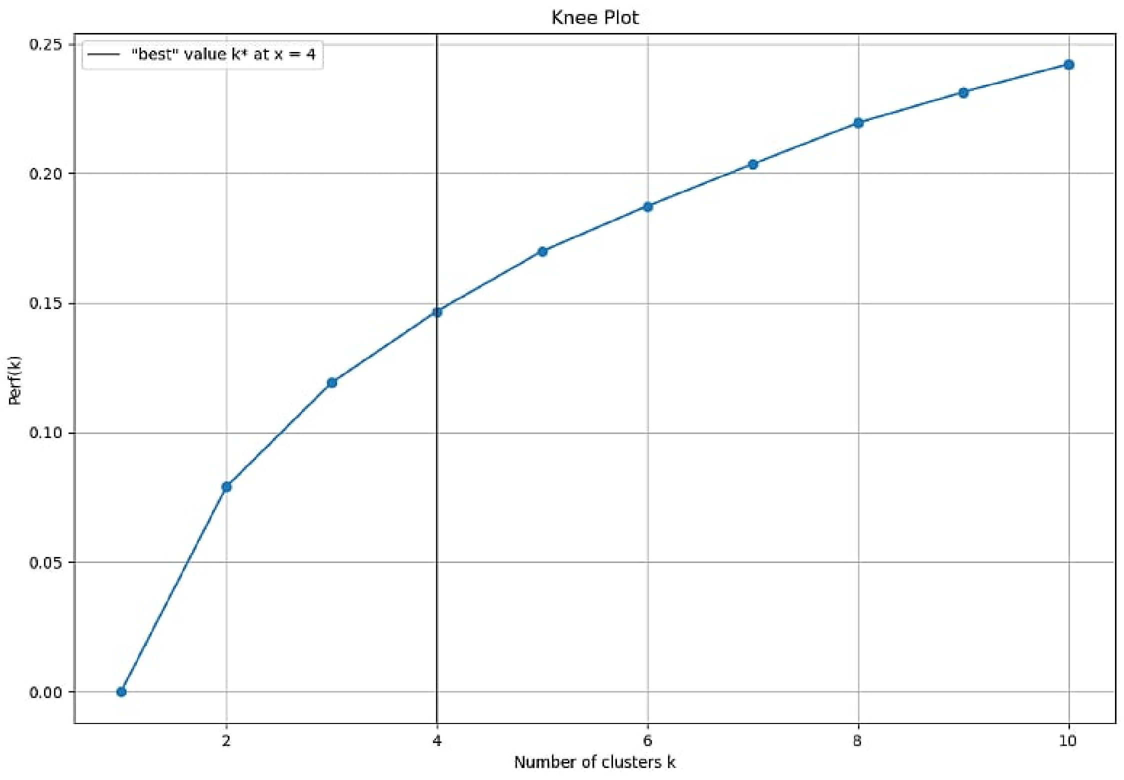

Figure 2: Plot of Perf(k) for Various Number of Clusters (Knee Plot)

Part 2

We will use kbest=4 to compute the centroids for each cluster. Table 5 shows the first 10 columns of the 4x400 matrix, where each row corresponds to the 400 coordinates for the center of each cluster. We will utilize the 400x400 correlation matrix of SDATA to obtain the first 3 eigenvectors, denoted w1, w2, w3. Then, we will compute <w1, CENTk>, <w2, CENTk>, and <w3, CENTk> to obtain the 3 coordinates for the 3-dimensional vector of each cluster center k. This will allow us to obtain a 4x3 matrix, shown below in Table 6, where each row is the coordinates corresponding to each cluster center. We display the cluster centers on a 3D plot below in Figure 3.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**
1	0.083	0.085	0.140	0.163	0.142	0.113	0.094	0.113	0.124	0.107
2	0.066	0.037	-0.007	-0.069	-0.123	-0.125	-0.034	0.046	0.057	0.042
3	-0.066	-0.026	-0.035	-0.043	-0.028	-0.050	-0.125	-0.204	-0.243	-0.235
4	-0.013	-0.074	-0.050	0.028	0.086	0.174	0.261	0.333	0.405	0.420

Table 5: k by N Matrix (1st 10 Columns)

	<w1, CENTk>,	**<w1, CENTk>,**	**<w1, CENTk>,**
1	1.544	-8.122	-1.384
2	-0.572	6.286	-0.517
3	-4.726	-0.540	0.930
4	11.059	1.494	-0.173

Table 6: k by 3 Matrix of Cluster Centers

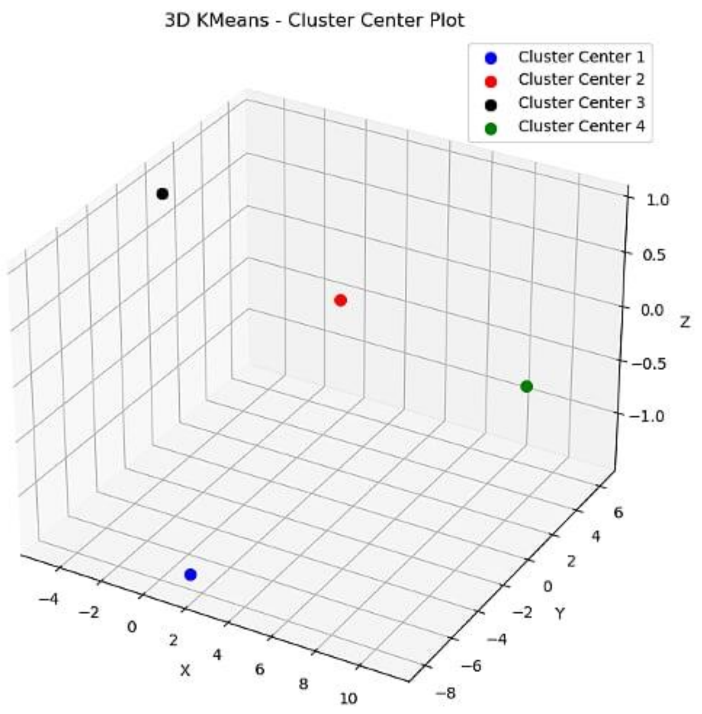

Figure 3: 3D KMeans – Cluster Center Plot

We obtain the size of each cluster by evaluating how many cases, or points, are in each cluster. The size of each cluster is displayed below in Table 7. We see from the table that cluster 3 had the biggest size of 5877 points, thus we denote cluster 3 as BigCLU. We will implement PCA on the 5877x400 BigCLU matrix to obtain the first 3 principal components that will be used to plot each case in BigCLU. Figure 4 below shows the 3D plot of all the data points in BigCLU, where each point is assigned a specific color based on the class (font) it belongs to. As we can see, our cluster is composed of all 3 classes. Upon further inspection, we noted that the class that makes up majority of BigClu is Courier followed by Times and Calibri, in that order.

The PCA algorithm stands for Principal Component Analysis and is a dimension reduction technique that constructs new features as linear combinations of original features. PCA helps eliminate redundant information and attempts to capture as much information as possible by transforming large sets of features into a smaller set while preserving as much information as possible. We used PCA to plot the coordinates in 3D of each point in BigCLU for visualization purposes.

In an ideal situation, we would like to see one class making up the entire cluster, however in Figure 4, we see all 3 classes within BigCLU, which may indicate poor clustering.

Cluster	Size
1	**2478**
2	**3152**
3	**5877**
4	**2328**

Table 7: Cluster Size

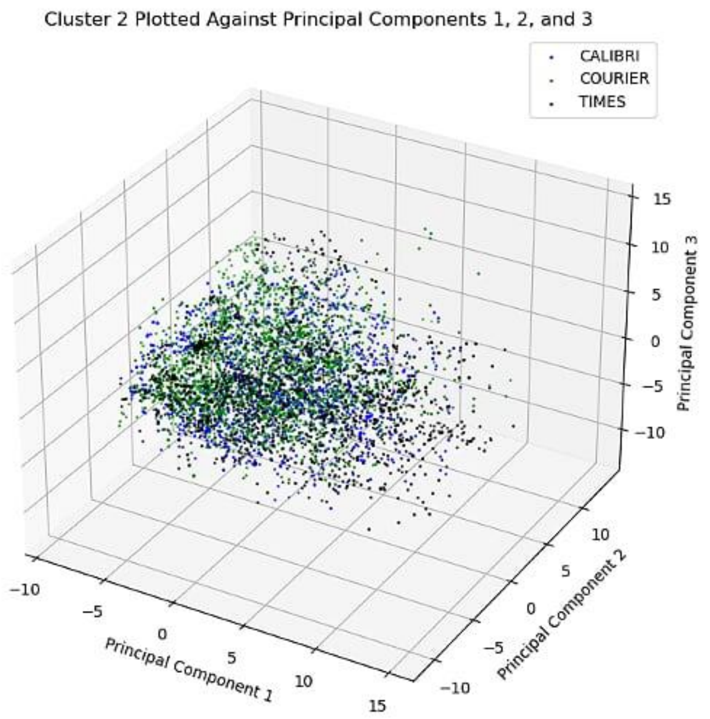

Figure 4: 3D - KMeans - BigCLU PCA Plot

Part 3

The k value of 4 (kbest) was chosen to be used in the Kmeans algorithm and subsequentially used to analyze the 4 different clusters created. Table 8 displays the class size within each cluster. The CALIBRI, COURIER and TIMES columns from Table 8 display how many cases of that class are displayed in each cluster. The class size within each cluster and the totals of each and all the clusters will be used to further analyze the cluster information. For reference, m refers to the classes Calibri, Courier and Times and j refers to the cluster 1, 2, 3 and 4.

Cluster (CLUj)	Size (Sj)	CALIBRI	COURIER	TIMES
1	2478	832	520	1126
2	3152	1905	604	643
3	5877	1360	2508	2009
4	2328	671	630	1027
Total	**13835**	**4768**	**4262**	**4805**

Table 8: Class Size Within Each Cluster and Totals

The class frequency within each cluster was calculated using equation 3 below. This helps analyze the percentage each class within each cluster to easily identify which class was predominately present in each cluster.

$$\boldsymbol{Fm(j) = \frac{A(m,j)}{Sizej}}$$

Equation 3: Class Frequency Within Each Cluster

The Gini index was calculated for each cluster using equation 4 below. The Gini index represents the impurity of each cluster CLUj. The maximum Gini index for a set made of up 3 classes is 0.67. The higher the value and closer it is to 0.67, the more impure the cluster is. High purity for a cluster would be close to 0. This is one measure that is used to be able to drilldown on the accuracy of the clustering done by the Kmeans algorithm.

$$\boldsymbol{Gini(CLUj) = F_{1,1} * (1 - F_{1,1}) + \ldots + F_{m,j} * (1 - F_{m,j})}$$

Equation 4: Gini(CLUj)

The impurity of the best Kmeans clustering into 4 clusters was calculated using equation 5 below. The kbest value is also represented as k*. This impurity was calculated by summing all the Gini indexes for each of the 4 clusters and gives a more holistic view of to the impurity of the clustering. Since we established that the maximum Gini index for each cluster is 0.67, our maximum IMP(k*) is 2.680.

$$\boldsymbol{IMP(k*) = gini(CLU1) + \ldots + gini(CLUj)}$$

Equation 5: Impurity IMP(k*)

Table 9 below compiles all the information and calculations stated above into one table for easy viewing and analysis. The highest Gini index for a cluster came from cluster 4 and the lowest came from cluster 2. This indicates that cluster 4 was the most impure cluster while cluster 2 was the purest of the set. All 4 cluster's Gini index range from 0.556 to 0.649 which is a relatively narrow range which indicates all clusters have similar Gini indexes. All 4 clusters had relatively high Gini indexes since they were all above 0.5 and were closest to the maximum impurity of 0.67, which indicate the clustering did a poor job. The impurity of best Kmeans clustering into 4 clusters is 2.490 which is also a relatively high value and close to the maximum possible IMP(k*) and further indicates the clustering did not do a good job.

Cluster	Size	CALIBRI	COURIER	TIMES	F_CALIBRI_J	F_COURIER_J	F_TIMES_J	Gini(CLUj)	IMP(k*)
1	2478	832	520	1126	0.336	0.210	0.454	0.637	2.490
2	3152	1905	604	643	0.604	0.192	0.204	0.556	2.490
3	5877	1360	2508	2009	0.231	0.427	0.342	0.647	2.490
4	2328	671	630	1027	0.288	0.271	0.441	0.649	2.490

Table 9: Compiled Cluster Data

Fm(j)	CLU1	CLU2	CLU3	CLU4
F_CALIBRI_J	0.336	0.604	0.231	0.288
F_COURIER_J	0.210	0.192	0.427	0.271
F_TIMES_J	0.454	0.204	0.342	0.441

Table 10: FREQ Table

A(m, j)	CLU1	CLU2	CLU3	CLU4
CALIBRI	832	1905	1360	671
COURIER	520	604	2508	630
TIMES	1126	643	2009	1027
A(TOP(j), j)	TIMES	CALIBRI	COURIER	TIMES

Table 11: Class Index Top(j) Table

Table 10 above displays the 3 x 4 matrix of class frequencies within each cluster that were calculated using Equation 3. Table 11 above displays the class size in each cluster which coincide with the data displayed in Table 10. The last row of Table 11 displays the top class of each cluster to help interpret what the predominate class is. This class index will be used as the predicted class for each cluster. This will be used in Part 4 to compare to the true class of each case and the predicted class of each case.

Part 4

Table 12 displays the class assignments to each cluster based on A(Top(j),j) from Table 11. We can see that our clustering will classify the class font as Times for 2 of our clusters. The aim of assigning a prediction class for each of the clusters that we created is to be able to create a predictor to compare our 13,835 actual case labels.

Cluster	PRED
1	TIMES
2	CALIBRI
3	COURIER
4	TIMES

Table 12: Prediction for Each Cluster

We applied our predictor based on the top frequency of classes in each cluster to predict the class for each case dependent on which cluster it belongs to. Table 13 shows the first 10 case's actual class label displayed alongside with its prediction label given by our predictor. There are some inconsistencies from the first 10 cases, indicating that our clustering may be a poor predictor.

	Actual Label	PRED Label
0	CALIBRI	CALIBRI
1	CALIBRI	TIMES
2	CALIBRI	COURIER
3	CALIBRI	TIMES
4	CALIBRI	CALIBRI
5	CALIBRI	CALIBRI
6	CALIBRI	COURIER
7	CALIBRI	COURIER
8	CALIBRI	COURIER
9	CALIBRI	CALIBRI

Table 13: Actual vs Predicted (First 10 Cases)

In Figure 5, our confusion matrix shows how well our predictor performed when we chose kbest=4 clusters. We can see that the predictor performs best for the Courier class, correctly predicting 58.85% of the Courier cases correctly. Our predictor performs worst on our Calibri class and suggests that we should seek alternatives to improve our Kmeans clustering. The bigget misclassification occurs on cases that are Times but are predicted to be Courier. All of our classes had poor individual percentages and all indicate that we had a poor Kmeans clustering model.

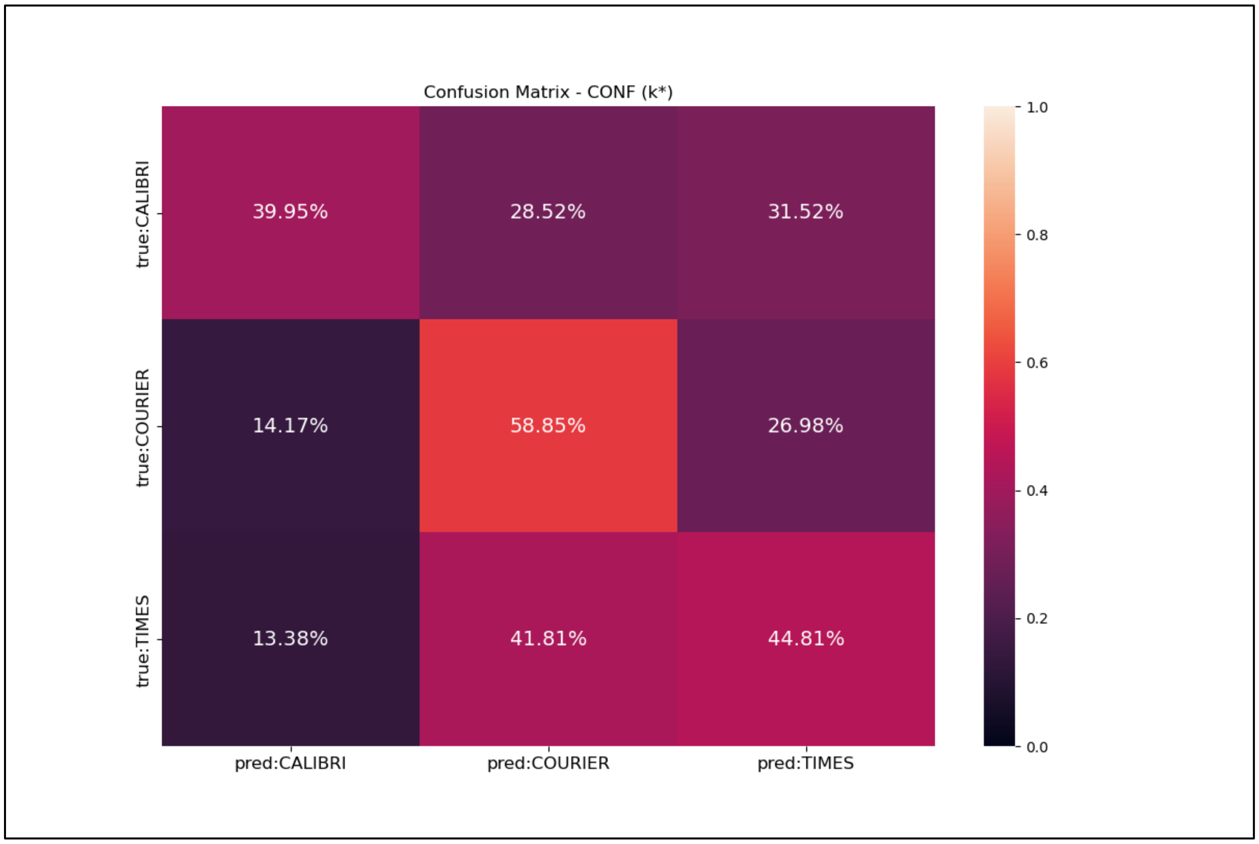

Figure 5: Confusion Matrix of KMeans Clustering kbest=4

FREQ

	TIMES Cluster 1	CALIBRI Cluster 2	COURIER Cluster 3	TIMES Cluster 4
CALIBRI Class	0.336	0.604	0.231	0.288
COURIER Class	0.210	0.192	0.427	0.271
TIMES Class	0.454	0.204	0.342	0.441

CONF

	CALIBRI	COURIER	TIMES
CALIBRI	0.400	0.285	0.315
COURIER	0.142	0.589	0.270
TIMES	0.134	0.418	0.448

Figure 6: FREQ vs CONF Matrices

Based on Figure 6, we observe that for the Courier class, our frequencies are minimum in clusters 1, 2, and 4 whereas, in cluster 3, our Courier frequency is the highest in comparison to the other class frequencies. This might explain why Courier is our highest correctly predicted class in our confusion matrix. Looking at our confusion matrix in Figure 6, we see that our prediction for our true Times and Courier cases are roughly split between each other. This is mainly because, our biggest cluster is cluster 3 which assigns every case in that cluster as a Courier class. We have 2,009 cases in that cluster that are Times classes. When we look at Clusters 1 and 4, which assign every case in that cluster as Times class, we see that in cluster 1 there are 1,126 cases of Times classes and in cluster 2 there are 1,027 cases of Times classes. Despite the frequencies in each cluster being high, the size of the cluster also plays a role in predicting the correct class.

The predictor extracted from Kmeans clustering was relatively poor at predicting the true class of each case. The true k is 3 since there are only 3 classes in the dataset. Using a k* of 3 could potentially improve the prediction and purity of the clusters. Applying Kmeans after applying PCA to the standardized dataset could also potentially result in higher purity of the clusters and better prediction since PCA eliminates a lot of redundancy in the data.

Summary

We performed pre-processing and filtering of the data to properly setup the data. Next, we standardized the features to be used in the Kmeans clustering algorithm. After performing Kmeans with various number of k clusters, we determined that the best number of clusters was kbest=4. We utilized a 3D plot to show the centers for each of our 4 clusters and displayed all the cases within the largest cluster, differentiating each case to its respective class. We computed the Gini index of each cluster and then computed the impurity for the entire set of clusters. We created a predictor for each cluster based on the top frequency class in each cluster and implemented this predictor for all the cases. The predictor extracted from Kmeans was relatively poor due to the high impurity of the clusters. Overall, our Kmeans clustering model was poor and could potentially be improved. Using a k* of 3 or applying PCA beforehand could potentially improve the predictor and purity of the clusters.

Acknowledgements

The authors gratefully acknowledge the guidance of Professor Robert Azencott.

Fonts Dataset Source

This dataset was taken from the University of California Irvine repository of machine learning datasets. The dataset can be found at following link: https://archive.ics.uci.edu/ml/machine-learning-databases/00417/

References

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) *An Introduction to Statistical Learning with applications in R*, www.StatLearning.com, Springer-Verlag, New York

Work Distribution

Jose Rodriguez: 33%
Dayana Sosa: 33%
Johnny Nino Ladino: 33%

Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import math
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.pyplot import cm
from collections import Counter, defaultdict

fsz=(12,8)

# Import text files of three specific fonts
Calibri = pd.read_csv(r"C:\Users\Rodri\Desktop\Fall 2020\MATH 6350 - Statistical Learning & Data Mining\2. Homework\
HW-2\fonts\CALIBRI.csv")
Courier = pd.read_csv(r"C:\Users\Rodri\Desktop\Fall 2020\MATH 6350 - Statistical Learning & Data Mining\2. Homework\
HW-2\fonts\COURIER.csv")
Times = pd.read_csv(r"C:\Users\Rodri\Desktop\Fall 2020\MATH 6350 - Statistical Learning & Data Mining\2. Homework\
HW-2\fonts\TIMES.csv")

################################################
#  ** Preliminary Treatment of the dataset **  #
################################################

# Discard the 9 columns below
# fontVariant, m_label, orientation, m_top, m_left, originalH, originalW, h, w
Calibri.drop(columns = ['fontVariant', 'm_label', 'orientation', 'm_top', 'm_left', 'originalH', 'originalW', 'h', 'w'], inplace=True)
Courier.drop(columns = ['fontVariant', 'm_label', 'orientation', 'm_top', 'm_left', 'originalH', 'originalW', 'h', 'w'], inplace=True
)
Times.drop(columns = ['fontVariant', 'm_label', 'orientation', 'm_top', 'm_left', 'originalH', 'originalW', 'h', 'w'], inplace=True)

# any row containing missing numerical data should be be discarded
Calibri.dropna(how='any', inplace = True)
Courier.dropna(how='any', inplace = True)
Times.dropna(how='any', inplace = True)

# define then three CLASSES of images of "normal" characters as follows
# strngth =0.4 and italic = 0
CL1 =(Calibri[(Calibri['strength']==0.4) & (Calibri['italic']==0)])
CL2 =(Courier[(Courier['strength']==0.4) & (Courier['italic']==0)])
CL3 =(Times[(Times['strength']==0.4) & (Times['italic']==0)])

# union of the three classes CL1 , CL2, CL3 and hence regroups N= n1 +n2 +n3 cases
DATA = pd.concat([CL1, CL2, CL3])
DATA.reset_index(inplace=True, drop=True)

######################
#  **** PART 0 ****  #
######################

# mean and standard deviation
m = DATA.iloc[:,3:].mean()
sd = DATA.iloc[:,3:].std()
# standardizing the features - rescaled data matrix
SDATA = (DATA.iloc[:,3:] - m)/sd
```

```
# Adding the font column to the standardized matrix
SDATA_ID = pd.concat([DATA.iloc[:,0], SDATA], axis=1)
SDATA.reset_index(inplace=True,drop=True)

#ID
ID = DATA.iloc[:,0]

######################
#  ** Question 1 **  #
######################

# kmeans clustering to partition the N cases into k disjoint clusters

from sklearn.cluster import KMeans
# calculate distortion for a range of number of cluster
sse = {}
for i in range(1, 11):
    km = KMeans(
        n_clusters=i,
        n_init=50,
        random_state=0
    )
    #km.fit(X_Train)
    km.fit(SDATA)
    # for each k compute the reduction of variance Perf(k)
    sse[i] = km.inertia_ #Sum of squared distances of samples to their closest cluster center.

sse_df = {'k': list(sse.keys()), 'SSE': list(sse.values())}
sse_df = pd.DataFrame(sse_df)

# Plot the curve SEE versus k
# Apply the "elbow rule" to select the "best" value k* for number of clusters k

k = 4 #elbow rule

plt.figure(figsize=fsz)
plt.grid()
plt.plot(list(sse.keys()), list(sse.values()), marker='o')
plt.xlabel('Number of clusters k')
plt.ylabel('SSE')
plt.title('Elbow Plot')
plt.axvline(x=k, label='"best" value k* at x = ' + str(k), c='k')
plt.legend()
plt.savefig('Elbow Plot.png')

# Plot the curve Perf(k) versus k
perf_k = 1 - (np.asarray(list(sse.values()))/list(sse.values())[0])

plt.figure(figsize=fsz)
plt.grid()
plt.plot(list(sse.keys()), perf_k, marker='o')
plt.xlabel('Number of clusters k')
plt.ylabel('Perf(k)')
plt.title('Knee Plot')
plt.axvline(x=k, label='"best" value k* at x = ' + str(k), c='k')
plt.legend()
plt.savefig('Knee Plot.png')

perf_k_df = {'k': list(sse.keys()), 'Perf(k)': perf_k}
perf_k_df = pd.DataFrame(perf_k_df)

#####################
#  ** Question 2 **  #
#####################

# Using k* and SDATA
km = KMeans(n_clusters=k, n_init=50, random_state=0).fit(SDATA)
c_center = km.cluster_centers_
c_center_df = pd.DataFrame(c_center, columns=['C%i' % i for i in range(1,401,1)])

# Use PCA to compute the 3-dimensional vectors c1,...,c4 gathering the first 3 pc of CENT1,...,CENT4
CORR = SDATA.corr()

# Compute the eigenvalues and eigenvectors of COR
eig, v = np.linalg.eig(CORR)

# W1 W2 W3 = the first three eigenvector of the 400x400 correlation matrix
w1 = pd.DataFrame(v)[0]
w2 = pd.DataFrame(v)[1]
w3 = pd.DataFrame(v)[2]

W = pd.DataFrame([w1,w2,w3])

# 3 dimensional vectors c1,...,c4
c1 = []
for i in [0,1,2]:
    c1.append(np.inner(W.loc[i], c_center_df.iloc[0,:]))
c2 = []
for i in [0,1,2]:
    c2.append(np.inner(W.loc[i], c_center_df.iloc[1,:]))
c3 = []
for i in [0,1,2]:
    c3.append(np.inner(W.loc[i], c_center_df.iloc[2,:]))
c4 = []
for i in [0,1,2]:
    c4.append(np.inner(W.loc[i], c_center_df.iloc[3,:]))

c_center_PCA_df  = pd.DataFrame([c1, c2, c3, c4], columns=['PCA%i' % i for i in range(1,4,1)])

# 3D Plot
#colors = ['b','g','k','r','c','m','y']
rgb_values = ['b','r','k','g']

# Plotting Values to 3D Plot
fig = plt.figure(figsize=fsz)
ax = fig.add_subplot(1,1,1, projection='3d')

#Plotting each individual cluser
ax.scatter(xs=c_center_PCA_df.iloc[0,0], ys=c_center_PCA_df.iloc[0,1], zs=c_center_PCA_df.iloc[0,2], c= rgb_values[0], s
=50)
ax.scatter(xs=c_center_PCA_df.iloc[1,0], ys=c_center_PCA_df.iloc[1,1], zs=c_center_PCA_df.iloc[1,2], c= rgb_values[1], s
=50)
ax.scatter(xs=c_center_PCA_df.iloc[2,0], ys=c_center_PCA_df.iloc[2,1], zs=c_center_PCA_df.iloc[2,2], c= rgb_values[2], s
=50)
ax.scatter(xs=c_center_PCA_df.iloc[3,0], ys=c_center_PCA_df.iloc[3,1], zs=c_center_PCA_df.iloc[3,2], c= rgb_values[3], s
=50)

ax.set_title('3D KMeans - Cluster Center Plot')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.legend(labels=['Cluster Center 1','Cluster Center 2','Cluster Center 3','Cluster Center 4'])
#plt.show()
plt.savefig('3D KMeans - Cluster Center Plot.png')

# Getting Cluster Size
C_Size = pd.DataFrame(list(dict(Counter(km.labels_)).items()),columns = ['Cluster','Size'])

# Largest Cluser
C_Size.sort_values(by=['Size'], ascending= False, ignore_index=True, inplace=True)
bigCLU = int(C_Size.iloc[0,:][0])
#bigCLU

# Narrowing to just rows that fell under bigCLU (Cluster 3)
bigCLU_DATA = SDATA[km.labels_ == bigCLU]

####################
#  ** PART PCA **  #
####################

from sklearn.decomposition import PCA

pca_model = PCA(n_components=3)
pca_model.fit(bigCLU_DATA)
bigCLU_DATA_PCA = pca_model.transform(bigCLU_DATA)

bigCLU_DATA_PCA_df = pd.DataFrame(bigCLU_DATA_PCA, columns=['PCA%i' % i for i in range(1,4,1)])

# Mapping Colors
# Unique category labels
bigCLU_ID = ID[km.labels_ == bigCLU]
bigCLU_ID.reset_index(inplace=True, drop=True)
font_labels = ID[km.labels_ == bigCLU].unique()

# List of RGB triplets
rgb_values = ['b','g','k']

# Appending bigCLU_DATA_PCA_df to be able to filter data
bigCLU_DATA_PCA_df = pd.concat([bigCLU_DATA_PCA_df,bigCLU_ID], axis=1)

# individual df to specific font
bigCLU_DATA_PCA_df1 = bigCLU_DATA_PCA_df[bigCLU_DATA_PCA_df['font'] == font_labels[0]]
bigCLU_DATA_PCA_df2 = bigCLU_DATA_PCA_df[bigCLU_DATA_PCA_df['font'] == font_labels[1]]
bigCLU_DATA_PCA_df3 = bigCLU_DATA_PCA_df[bigCLU_DATA_PCA_df['font'] == font_labels[2]]

# 3D Plot
fig = plt.figure(figsize=fsz)
ax = fig.add_subplot(1,1,1, projection='3d')

# Plotting each individual case from each class
ax.scatter(xs=bigCLU_DATA_PCA_df1.iloc[:,0], ys=bigCLU_DATA_PCA_df1.iloc[:,1], zs=bigCLU_DATA_PCA_df1.ilo
c[:,2], c = rgb_values[0] , s=1)
ax.scatter(xs=bigCLU_DATA_PCA_df2.iloc[:,0], ys=bigCLU_DATA_PCA_df2.iloc[:,1], zs=bigCLU_DATA_PCA_df2.ilo
c[:,2], c = rgb_values[1] , s=1)
ax.scatter(xs=bigCLU_DATA_PCA_df3.iloc[:,0], ys=bigCLU_DATA_PCA_df3.iloc[:,1], zs=bigCLU_DATA_PCA_df3.ilo
c[:,2], c = rgb_values[2] , s=1)

ax.set_title('Cluster ' + str(bigCLU) + ' Plotted Against Principal Components 1, 2, and 3')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
plt.legend(labels=list(font_labels))
#plt.show()
plt.savefig('3D KMeans - BigCLU PCA Plot.png')

######################
#  ** Question 3 **  #
######################

# m is classes
# j is clusters

C_Size.sort_values(by=['Cluster'], ascending= True, ignore_index=True, inplace=True)

# All clusters in a list
Cluster = list(C_Size.iloc[:,0])

Cluster_Class = {}
for i in range(len(Cluster)):
    Cluster_Class[i] = dict(Counter(ID[km.labels_ == i]))

Cluster_Class_df = pd.DataFrame(Cluster_Class).T
Cluster_Class_df['A(Top(j),j)'] = Cluster_Class_df.idxmax(axis=1)

#A(m,j)
#Cluster_Class_df

# Merge both dataframes
Cluster_Data = pd.concat([C_Size, Cluster_Class_df], axis=1)
#Cluster_Data

#Compute the Fj for each cluster j = 1 … k
#Cluster_Data['F_CALIBRI'] =  Cluster_Data['CALIBRI']/Cluster_Data['Size'].sum()
#Cluster_Data['F_COURIER'] =  Cluster_Data['COURIER']/Cluster_Data['Size'].sum()
#Cluster_Data['F_TIMES'] =  Cluster_Data['TIMES']/Cluster_Data['Size'].sum()

#Compute the Fmj for each cluster j = 1 … k
Cluster_Data['F_CALIBRI_J'] =  Cluster_Data['CALIBRI']/Cluster_Data['Size']
Cluster_Data['F_COURIER_J'] =  Cluster_Data['COURIER']/Cluster_Data['Size']
Cluster_Data['F_TIMES_J'] =  Cluster_Data['TIMES']/Cluster_Data['Size']

#Compute the gini indexes gin(CLUj) for each cluster j = 1 … k
Cluster_Data['Gini(CLUj)'] =  Cluster_Data['F_CALIBRI_J']*(1-
Cluster_Data['F_CALIBRI_J']) + Cluster_Data['F_COURIER_J']*(1-
Cluster_Data['F_COURIER_J']) + Cluster_Data['F_TIMES_J']*(1-Cluster_Data['F_TIMES_J'])
#Cluster_Data

#Compute the Impurity IMP(k*) of the clustering CLU1 ,…, CLUk
Cluster_Data['Impurity IMP(k*)'] = Cluster_Data['Gini(CLUj)'].sum()
#Cluster_Data

#Display the 3 x k matrix of frequencies FREQ = Fm(j)
FREQ = Cluster_Data.iloc[:,6:9].T
#FREQ

#Compute for each j  =1 .. K the class index  TOP(j)   such that
#A(TOP(j), j)= max (A(1,j), A(2,j), A(3,j))
A_Top = Cluster_Data.iloc[:,2:6].T
#A_Top
```

```
######################
#  ** Question 4 **  #
######################

#Use the clustering to define predictor  Pred(n)   for class of case(n) by
#Pred(n) = TOP(j(n)) not necessarily = true class of casen
PRED = pd.concat([Cluster_Data.iloc[:,0],Cluster_Data.iloc[:,5]], axis=1)

ID_s = pd.Series(ID)
Label_s = pd.Series(km.labels_)

ID_Label_dict = {'ID':ID_s, 'Label': Label_s}
ID_Label = pd.DataFrame(ID_Label_dict)

PRED_map = dict(zip(Cluster_Data.iloc[:,0], Cluster_Data.iloc[:,5]))
PRED_Label = Label_s.map(PRED_map)

ID_PRED_Label_dict = {'Actual Label':ID_s, 'PRED Label': PRED_Label}
ID_PRED_Label = pd.DataFrame(ID_PRED_Label_dict)
#ID_PRED_Label

#Compute the 3x3 confusion matrix CONF of the predictor Pred(n)

# Making the Confusion Matrix
CONF = confusion_matrix(ID_PRED_Label.iloc[:,0], ID_PRED_Label.iloc[:,1], labels = ['CALIBRI', 'COURIER', 'TIMES']
, normalize = 'true')

# Visuals for Report
cmtx_CONF = pd.DataFrame(CONF,
    index=['true:CALIBRI', 'true:COURIER', 'true:TIMES'],
    columns=['pred:CALIBRI', 'pred:COURIER', 'pred:TIMES'])

# Confusion Matrix Plots
plt.figure(figsize=fsz)
a = sns.heatmap(cmtx_CONF, annot=True, fmt = '.02%', annot_kws={"size": 14}, vmin = 0, vmax = 1)
a.set_xticklabels(a.get_xticklabels(), rotation = 0, fontsize = 12, horizontalalignment = "center")
a.set_yticklabels(a.get_yticklabels(), fontsize = 12, verticalalignment = 'center')
a.set_title("Confusion Matrix - CONF (k*)")
plt.savefig('Confusion Matrix - CONF.png')

#######################
#  ** DF To Excel **  #
#######################

with pd.ExcelWriter('DataFrame Outputs.xlsx') as writer:
    Cluster_Data.to_excel(writer, sheet_name='Cluster Master Table')
    sse_df.to_excel(writer, sheet_name='SSE Table')
    perf_k_df.to_excel(writer, sheet_name='Perf(k) Table')
    c_center_PCA_df.to_excel(writer, sheet_name='Cluster Center (PCA) Table')
    c_center_df.to_excel(writer, sheet_name='Cluster Center (Non-PCA) Table')
    C_Size.to_excel(writer, sheet_name='Cluster Size Table')
    Cluster_Class_df.to_excel(writer, sheet_name='Cluster Class Table')
    FREQ.to_excel(writer, sheet_name='FREQ Table')
    A_Top.to_excel(writer, sheet_name='A Top Table')
    PRED.to_excel(writer, sheet_name='PRED Table')
    bigCLU_DATA_PCA_df.to_excel(writer, sheet_name='bigCLU (PCA) Data')
    ID_PRED_Label.to_excel(writer, sheet_name='Actual & Pred Data')
```