DATASET:

The font classes selected for the group were “CAMBRIA”, “NIRMALA”, and “CONSOLAS”. Preliminary treatment was conducted on the dataset. This involved the following:

i) The following 9 columns were discarded: fontVariant, label, orientation, m_top, m_left, originalH, originalW, h, w

ii) The following 3 columns were kept, in addition to the 400 columns named r0c0, r0c1,r0c2, ... , r19c18, r19c19: font, strength and italic

iii) 3 CLASSES CL1, CL2 and CL3 were defined such that:

a. CL1 = All rows of “CAMBRIA”, for which strength = 0.4 and italic=0 (size n1 = 3208)

b. CL2 = All rows of “NIRMALA” for which strength = 0.4 and italic=0 (size n2 = 2348)

c. CL3 = All rows of “CONSOLAS” for which strength = 0.4 and italic=0 (size n3 = 2285)

The full data set used for the rest of the report was the union of all 3 classes CL1, CL2, and CL3. Parts 0 to 1.8 of the report covers how basic machine learning tools covered in class so far were used to attempt a rough automatic classification of the full data set, into 3 classes CL1, CL2 and CL3

Part 0:

The dataset was standardized by centering and rescaling each feature X_j into a new feature Y_j such that $Y_j = (X_j – m_j)/ s_j$, where m_j is the mean for each case and s_j is the standard deviation. This new dataset was called SDATA. The 400X400 correlation matrix of SDATA was computed and the 10 pairs Xi,Xj of features which have the 20 highest absolute values were extracted, as shown below in Table 1.

It can be observed from the table that the pixels at the top and bottom corners display the highest correlation among the top 10 particularly in the 19^{TH} and 0^{th} row.

TABLE 1. The 10 pairs Xi,Xj of features which have the 20 highest correlation values: |CORR(i,j)|

ROW	COLUMN	|CORR|
r19c1	r19c0	0.9321
r19c19	r19c18	0.9287
r19c10	r19c9	0.9259
r10c19	r9c19	0.9236
r0c1	r0c0	0.9229
r0c2	r0c1	0.9218

r10c18	r9c18	0.9143
r10c0	r9c0	0.9141
r19c2	r19c1	0.9132
r0c19	r0c18	0.9130

Part 1.0:

The classes CL1, CL2 and CL3 in SDATA, were each split by a ratio of 20% to 80%. The full training set TRAINSET was defined as the union of 80% of CL1, CL2 and CL3. The full test set TESTSET was defined as the union of the remaining 20% of CL1, CL2 and CL3

Part 1.1:

The K nearest neighbor (KNN) algorithm was used on SDATA for the automatic classification of arbitrary cases into one of the three classes CL1 CL2 CL3, with K fixed as 12. The two percentages of correct classifications on the TRAINSET AND TESTSET were computed, as shown below in TABLE 2.

TABLE 2. Accuracy Percentage for the TRAINSET and TESTSET

SET	**ACCURACY**
TRAIN	80.69%
TEST	75.65%

This means that using the KNN algorithm for K = 12, the automatic classification placed each case in the training dataset into one of the three classes CL1, CL2, and CL3 correctly 80.69% of the time while the classification placed each case in the test dataset into one of the three classes correctly 75.69% of the time. Also, the ratio Accuracy(Test)/Accuracy(Train) = 0.9376, which is reasonably close to 1. This tells us that there is satisfactory performance so far, but it would still have to be validated on other new cases. Since Accuracy(Train) is not much larger than Accuracy(Test), we know that there is not an overfit to the training data.

Part 1.2:

The preceding operation done in Part 1.1 above, was repeated for the following values of K: = 5, 10 , 15, 20, 30, 40, 50 ,100. The associated accuracy percentages of correct classifications on TESTSET were

computed, as shown in TABLE 3 below. The curve of the accuracy of K versus K was plotted. This was done to identify the best range of values of K.

TABLE 3. The associated percentages of correct classifications on TESTSET

K-VALUE	TESTSET ACCURACY
5	78.01%
10	76.35%
15	74.95%
20	74.12%
30	72.66%
40	71.96%
50	70.87%
100	66.86%
200	61.44%

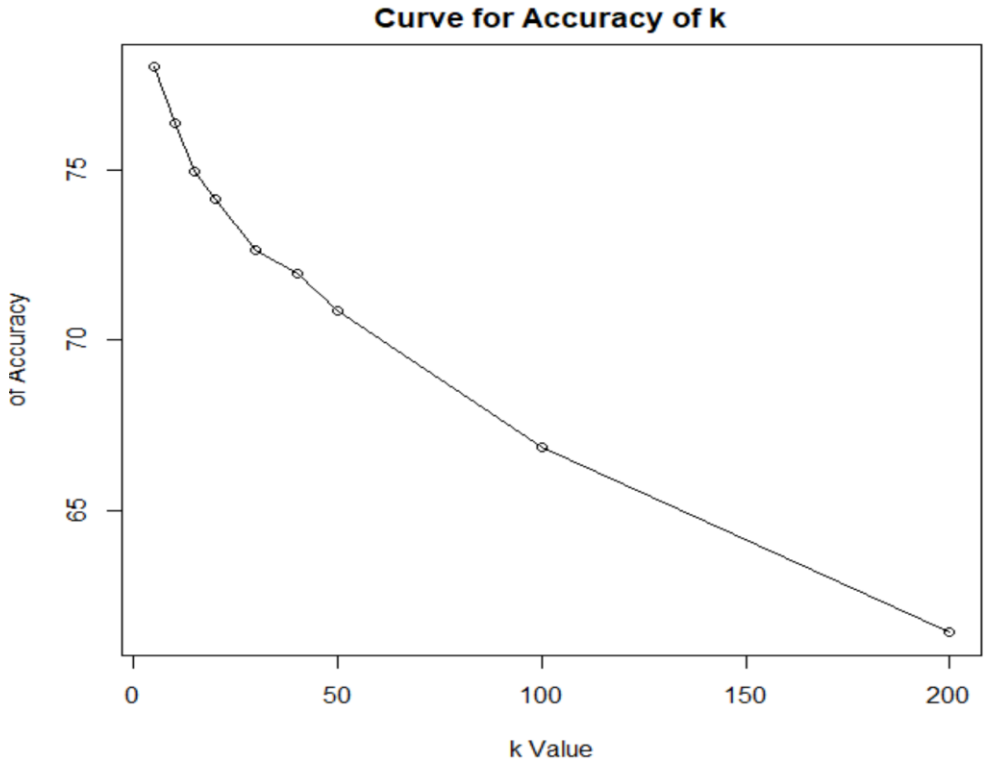


FIGURE 1. TESTSET Accuracy versus K

From FIGURE 1 above, there seems to be a somewhat inverse linearity between the TESTSET accuracy percentage and the K-value. [5,10] was the range of values chosen by the group for further exploration, to choose the best value of K.

Part 1.3:

The preceding exploration was repeated for a few more values of K within the range [5,10]. The associated accuracy percentages of correct classifications on TESTSET were computed, as shown in TABLE 4 below. The curve of the accuracy of K versus K was plotted. This was done to identify the best value of K.

TABLE 4. The associated percentages of correct classifications on TESTSET

K-VALUE	TESTSET ACCURACY
5	78.01%
6	78.08%
7	78.71%
8	78.14%
9	76.61%
10	76.04%

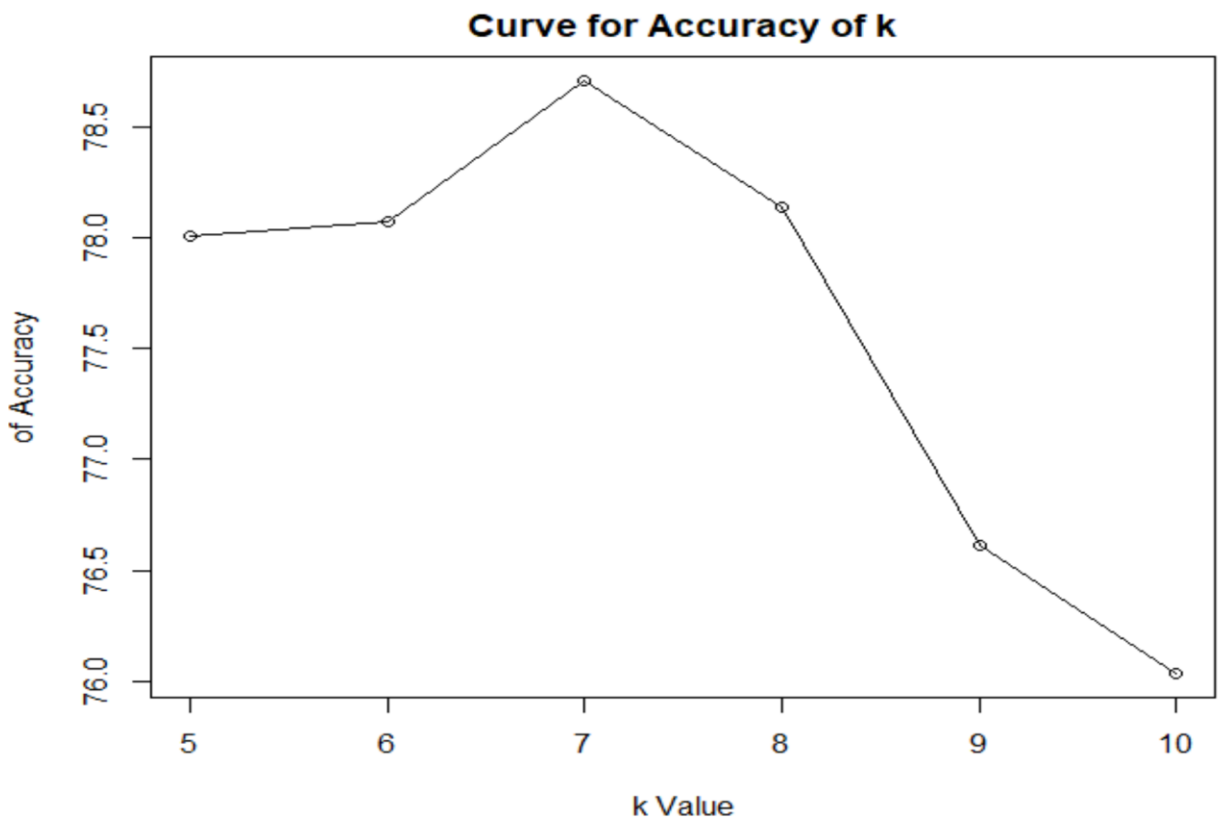


FIGURE 2. TESTSET Accuracy versus K

As shown above in FIGURE 2, the best value of K in the range [5,10] is 7, having the highest accuracy of 78.71%

Part 1.4:

From Part 1.3, the best value of K was identified to be 7. Using K as 7, the 3X3 confusion matrix for the KNN classifications was computed for the TRAINSET and TESTSET, as shown below in TABLES 5 and 6.

TABLE 5. TESTSET Confusion Matrix

	PredictedClass CL1	PredictedClass CL2	PredictedClass CL3
TrueClass CL1	0.810	0.120	0.070
TrueClass CL2	0.166	0.753	0.081
TrueClass CL3	0.092	0.040	0.868

TABLE 6. TRAINSET Confusion Matrix

	PredictedClass CL1	PredictedClass CL2	PredictedClass CL3
TrueClass CL1	0.845	0.101	0.054
TrueClass CL2	0.142	0.778	0.080
TrueClass CL3	0.086	0.030	0.884

From Table 5, it is shown that 81% of the cases that were truly in CL1 were predicted correctly, while 12% of them were predicted as CL2 and 7% predicted as CL 3. 75.3% of the cases that were truly in CL2 were predicted correctly, while 16.6% of them were predicted as CL1 and 8.1% predicted as CL3. Lastly, 86.8% of the cases that were truly in CL3 were predicted correctly, while 4% of them were predicted as CL2 and 9.2% predicted as CL1. Overall, for the TESTSET, 19% of the CL1 cases were predicted incorrectly, 24.7% of the CL2 cases were predicted incorrectly and 13.2% of the CL 3 cases were predicted incorrectly. From Table 6, it is shown that 84.5% of the cases that were truly in CL1 were predicted correctly, while 10.1% of them were predicted as CL2 and 5.4% predicted as CL 3. 77.8% of the cases that were truly in CL2 were predicted correctly, while 14.2% of them were predicted as CL1 and 8% predicted as CL3. Lastly, 88.4% of the cases that were truly in CL3 were predicted correctly, while 3% of them were predicted as CL2 and 8.6% predicted as CL1. Overall, for the TRAINSET, 15.5% of the CL1 cases were predicted incorrectly, 22.2% of the CL2 cases were predicted incorrectly and 11.6% of the CL 3 cases were predicted incorrectly.
As expected, the accuracy percentage of the TRAINSET is higher than that of the TESTSET. However, the algorithm is not able to predict CL2 as well as CL1 or CL3, with the percentage of incorrect predictions in both the TRAINSET AND TESTSET, being above 20%.

Part 1.5:

As shown below, the confidence intervals (Testconf) for the 3 diagonal terms of the matrix in table 5 were computed.

Testconf(TrueClass1, PredictedClass31) = (0.7781, 0.8425)

Testconf(TrueClass 2, PredictedClass32) = (0.7172, 0.7880)

Testconf(TrueClass 3, PredictedClass3) = (0.8396, 0.8953)

Regarding the TESTSET, this implies that for CL1, we are 95% confident that the percentage of correctly predicted cases should lie between 77.81% and 84.25%. For CL2, we are 95% confident that the

percentage of correctly predicted cases should lie between 71.72% and 78.80%. Lastly, for CL3, we are 95% confident that the percentage of correctly predicted cases should lie between 83.96% and 89.53%.

Part 1.6:

The package PACK1 of 100 attributes with names rLcM where L = 0, 1, 2, ,9 and M= 0, 1, 2, ,9 was considered. PACK1= { Z1, Z2, Z3, ... ,Z100 }, where Z1, Z2,Z3, …,Z100 are 100 features that correspond to the 100-pixel intensities displayed in a specific 10X10 window of the 20X20 pixel image. The KNN classification with the best value of K set at 7, was applied on PACK 1. The percentage of correct classifications on the TESTSET was computed to be 76.04%.

Part 1.7:

The operation from Part 1.6 above was repeated for 3 other packages PACK2, PACK3, PACK4 of 100 features each , such that:

PACK2 contains 100 attributes with names rLcM where L = 0, 1, 2, …,9 and M= 10, 11, 12, …,19.

PACK3 contains 100 attributes with names rLcM where L = 10, 11, 12, …,19 and M= 10, 11, 12, …,19.

PACK4 contains 100 attributes with names rLcM where L = 10, 11, 12, …,19 and M= 0, 1, 2, …,9

The percentage accuracy gotten from using each PACK was computed and is displayed below in TABLE 7

TABLE 7. The associated percentages of correct classifications on TESTSET gotten by using each PACK

PACKAGE	TESTSET ACCURACY
PACK2 (w2)	77.57%
PACK3 (w3)	79.60%
PACK4 (w4)	78.71%

PART 1.8:

Weights were assigned to each feature in PACK1, PACK2, PACK3 and PACK4. All 400 weights were normalized so that their total sum became 1. Using best value of K; 7, the K nearest neighbor (KNN) algorithm was used on all 400 features but with the weighted distance defined by these 400 weights. As

shown below in TABLE 8, the 3X3 confusion matrix for the KNN classifications was computed for the TESTSET.

TABLE 8. TESTSET Confusion Matrix

	PredictedClass CL1	**PredictedClass CL2**	**PredictedClass CL3**
TrueClass CL1	0.902628	0.0819	0.015456
TrueClass CL2	0.112936	0.8008	0.086242
TrueClass CL3	0.006897	0.0322	0.96092

The overall global performance of this method is that there were 88.71893% of correct classifications the best K-value (7). In comparison to the ordinary distance version of KNN, the weighted version provides more accurate predictions. Overall prediction accuracy was improved from 78.71% as stated in Part 1.3, to 88.72%. With this weighted version, CL3 prediction was nearly perfect, with about 96% accuracy, and CL2 with about 90% accuracy. However, the most notable improvement is in the case of CL2, which now has 19.91% of incorrectly predicted cases, as opposed to 24.7% in Part 1.4. With this weighted version, CL3 prediction was nearly perfect, with about 96% accuracy, and CL2 with about 90% accuracy.

CODE

```
library(readr)

library(dplyr)
library(tidyr)
library(class)


#CL1
CAMBRIA =read_csv("fonts/CAMBRIA.csv") %>%
      select(font,strength,italic,r0c0:r19c19) %>%
      filter(strength == 0.4, italic == 0) %>%
      mutate(font = "CAMBRIA")

#CL2
NIRMALA =read_csv("fonts/NIRMALA.csv") %>%
       select(font,strength,italic,r0c0:r19c19) %>%
       filter(strength == 0.4, italic == 0) %>%
       mutate(font = "NIRMALA")

#CL3
CONSOLAS =read_csv("fonts/CONSOLAS.csv") %>%
       select(font,strength,italic,r0c0:r19c19) %>%
       filter(strength == 0.4, italic == 0) %>%
       mutate(font = "CONSOLAS")

# sizes of each font class
n1 = dim(CAMBRIA)[1]
n2 = dim(NIRMALA)[1]
n3 = dim(CONSOLAS)[1]


# dataset for the union of CL1, CL2, CL3
df = rbind(CAMBRIA,NIRMALA,CONSOLAS) %>% select(-c(strength,italic)) %>% drop_na() %>
% mutate(font = as.numeric(as.factor(font)))
df$font = as.factor(df$font)
head(df)

## # A tibble: 6 x 401
##   font   r0c0  r0c1  r0c2  r0c3  r0c4  r0c5  r0c6  r0c7  r0c8  r0c9 r0c10 r0c11
##   <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1       255   255   115    84   255   255   240     1   214   255   255   110
## 2 1         1     1     1    58   167   255   255   255    38     1    10   151
## 3 1         1     1     1    58   167   255   255   255    38     1    10   151
## 4 1         1     1     1     1     6    38   140   226   255   255   255   237
## 5 1         1     1     1     1    12    51   147   249   255   255   255   196
## 6 1         1     1     1     1    53   147   235   255   255   255   122     1
## # ... with 388 more variables: r0c12 <dbl>, r0c13 <dbl>, r0c14 <dbl>,
## #   r0c15 <dbl>, r0c16 <dbl>, r0c17 <dbl>, r0c18 <dbl>, r0c19 <dbl>,
## #   r1c0 <dbl>, r1c1 <dbl>, r1c2 <dbl>, r1c3 <dbl>, r1c4 <dbl>, r1c5 <dbl>,
```

```
## #   r1c6 <dbl>, r1c7 <dbl>, r1c8 <dbl>, r1c9 <dbl>, r1c10 <dbl>, r1c11 <dbl>,
## #   r1c12 <dbl>, r1c13 <dbl>, r1c14 <dbl>, r1c15 <dbl>, r1c16 <dbl>,
## #   r1c17 <dbl>, r1c18 <dbl>, r1c19 <dbl>, r2c0 <dbl>, r2c1 <dbl>, r2c2 <dbl>,
## #   r2c3 <dbl>, r2c4 <dbl>, r2c5 <dbl>, r2c6 <dbl>, r2c7 <dbl>, r2c8 <dbl>,
## #   r2c9 <dbl>, r2c10 <dbl>, r2c11 <dbl>, r2c12 <dbl>, r2c13 <dbl>,
## #   r2c14 <dbl>, r2c15 <dbl>, r2c16 <dbl>, r2c17 <dbl>, r2c18 <dbl>,
## #   r2c19 <dbl>, r3c0 <dbl>, r3c1 <dbl>, r3c2 <dbl>, r3c3 <dbl>, r3c4 <dbl>,
## #   r3c5 <dbl>, r3c6 <dbl>, r3c7 <dbl>, r3c8 <dbl>, r3c9 <dbl>, r3c10 <dbl>,
## #   r3c11 <dbl>, r3c12 <dbl>, r3c13 <dbl>, r3c14 <dbl>, r3c15 <dbl>,
## #   r3c16 <dbl>, r3c17 <dbl>, r3c18 <dbl>, r3c19 <dbl>, r4c0 <dbl>, r4c1 <dbl>,
## #   r4c2 <dbl>, r4c3 <dbl>, r4c4 <dbl>, r4c5 <dbl>, r4c6 <dbl>, r4c7 <dbl>,
## #   r4c8 <dbl>, r4c9 <dbl>, r4c10 <dbl>, r4c11 <dbl>, r4c12 <dbl>, r4c13 <dbl>,
## #   r4c14 <dbl>, r4c15 <dbl>, r4c16 <dbl>, r4c17 <dbl>, r4c18 <dbl>,
## #   r4c19 <dbl>, r5c0 <dbl>, r5c1 <dbl>, r5c2 <dbl>, r5c3 <dbl>, r5c4 <dbl>,
## #   r5c5 <dbl>, r5c6 <dbl>, r5c7 <dbl>, r5c8 <dbl>, r5c9 <dbl>, r5c10 <dbl>,
## #   r5c11 <dbl>, ...
```

```
print(paste("There are",n1,"cases for CAMBRIA font" ))
```

```
## [1] "There are 3208 cases for CAMBRIA font"
```

```
print(paste("There are",n2,"cases for NIRMALA font" ))
```

```
## [1] "There are 2348 cases for NIRMALA font"
```

```
print(paste("There are",n3,"cases for CONSOLAS font" ))
```

```
## [1] "There are 2285 cases for CONSOLAS font"
```

```
print(paste("There are",dim(df)[1],"cases for the final dataset" ))
```

```
## [1] "There are 7841 cases for the final dataset"
```

Compute the 400x400 correlation matrix CORR of the 400 features X1 …X400 Extract the 10 pairs Xi,Xj of features which have the 20 highest absolute values |CORR(i,j)|

```
SDATA = df %>% mutate_at(.vars = colnames(df)[2:401], .funs = scale)
head(SDATA)
```

```
## # A tibble: 6 x 401
##   font  r0c0[,1] r0c1[,1] r0c2[,1] r0c3[,1] r0c4[,1] r0c5[,1] r0c6[,1] r0c7[,1]
##   <fct>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 1        1.75     1.55     0.244  -0.0907    1.38     1.30     1.13    -1.04
## 2 1       -0.640   -0.714   -0.767  -0.322     0.594    1.30     1.26     1.22
## 3 1       -0.640   -0.714   -0.767  -0.322     0.594    1.30     1.26     1.22
## 4 1       -0.640   -0.714   -0.767  -0.828    -0.839   -0.618    0.235    0.961
## 5 1       -0.640   -0.714   -0.767  -0.828    -0.785   -0.503    0.298    1.17
## 6 1       -0.640   -0.714   -0.767  -0.828    -0.420    0.346    1.08     1.22
## # ... with 392 more variables: r0c8[,1] <dbl>, r0c9[,1] <dbl>, r0c10[,1] <dbl>,
## #   r0c11[,1] <dbl>, r0c12[,1] <dbl>, r0c13[,1] <dbl>, r0c14[,1] <dbl>,
## #   r0c15[,1] <dbl>, r0c16[,1] <dbl>, r0c17[,1] <dbl>, r0c18[,1] <dbl>,
```

```
## #   r0c19[,1] <dbl>, r1c0[,1] <dbl>, r1c1[,1] <dbl>, r1c2[,1] <dbl>,
## #   r1c3[,1] <dbl>, r1c4[,1] <dbl>, r1c5[,1] <dbl>, r1c6[,1] <dbl>,
## #   r1c7[,1] <dbl>, r1c8[,1] <dbl>, r1c9[,1] <dbl>, r1c10[,1] <dbl>,
## #   r1c11[,1] <dbl>, r1c12[,1] <dbl>, r1c13[,1] <dbl>, r1c14[,1] <dbl>,
## #   r1c15[,1] <dbl>, r1c16[,1] <dbl>, r1c17[,1] <dbl>, r1c18[,1] <dbl>,
## #   r1c19[,1] <dbl>, r2c0[,1] <dbl>, r2c1[,1] <dbl>, r2c2[,1] <dbl>,
## #   r2c3[,1] <dbl>, r2c4[,1] <dbl>, r2c5[,1] <dbl>, r2c6[,1] <dbl>,
## #   r2c7[,1] <dbl>, r2c8[,1] <dbl>, r2c9[,1] <dbl>, r2c10[,1] <dbl>,
## #   r2c11[,1] <dbl>, r2c12[,1] <dbl>, r2c13[,1] <dbl>, r2c14[,1] <dbl>,
## #   r2c15[,1] <dbl>, r2c16[,1] <dbl>, r2c17[,1] <dbl>, r2c18[,1] <dbl>,
## #   r2c19[,1] <dbl>, r3c0[,1] <dbl>, r3c1[,1] <dbl>, r3c2[,1] <dbl>,
## #   r3c3[,1] <dbl>, r3c4[,1] <dbl>, r3c5[,1] <dbl>, r3c6[,1] <dbl>,
## #   r3c7[,1] <dbl>, r3c8[,1] <dbl>, r3c9[,1] <dbl>, r3c10[,1] <dbl>,
## #   r3c11[,1] <dbl>, r3c12[,1] <dbl>, r3c13[,1] <dbl>, r3c14[,1] <dbl>,
## #   r3c15[,1] <dbl>, r3c16[,1] <dbl>, r3c17[,1] <dbl>, r3c18[,1] <dbl>,
## #   r3c19[,1] <dbl>, r4c0[,1] <dbl>, r4c1[,1] <dbl>, r4c2[,1] <dbl>,
## #   r4c3[,1] <dbl>, r4c4[,1] <dbl>, r4c5[,1] <dbl>, r4c6[,1] <dbl>,
## #   r4c7[,1] <dbl>, r4c8[,1] <dbl>, r4c9[,1] <dbl>, r4c10[,1] <dbl>,
## #   r4c11[,1] <dbl>, r4c12[,1] <dbl>, r4c13[,1] <dbl>, r4c14[,1] <dbl>,
## #   r4c15[,1] <dbl>, r4c16[,1] <dbl>, r4c17[,1] <dbl>, r4c18[,1] <dbl>,
## #   r4c19[,1] <dbl>, r5c0[,1] <dbl>, r5c1[,1] <dbl>, r5c2[,1] <dbl>,
## #   r5c3[,1] <dbl>, r5c4[,1] <dbl>, r5c5[,1] <dbl>, r5c6[,1] <dbl>,
## #   r5c7[,1] <dbl>, ...
```

Compute the 400x400 correlation matrix CORR of the 400 features X1 …X400 Extract the 10 pairs Xi,Xj of features which have the 20 highest absolute values |CORR(i,j)|

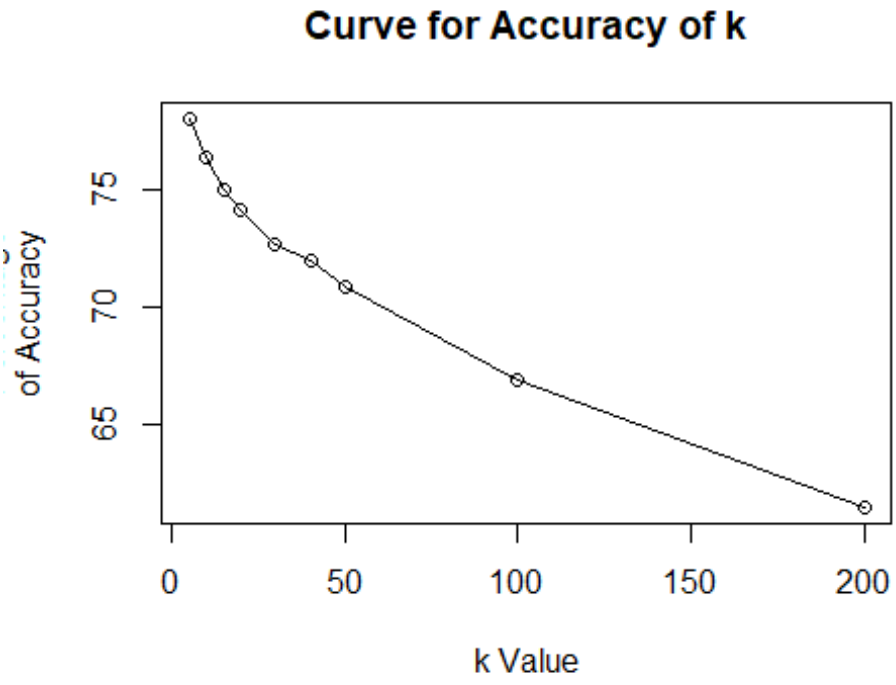
```
mat = abs(cor (SDATA[,-1]))
# sorted_mat = sort(mat,decreasing =TRUE)
# sorted_mat = sorted_mat[sorted_mat !=1]
cor_df = as_data_frame(mat,rownames = "Row")  %>%
        gather(Column,corr,r0c0:r19c19) %>%
        filter(corr!= 1,Row!= Column) %>% arrange(desc(corr)) %>% filter(!duplicated(corr))

cor_df[1:10,]
```

```
## # A tibble: 10 x 3
##    Row    Column  corr
##    <chr>  <chr>  <dbl>
##  1 r19c1  r19c0  0.932
##  2 r19c19 r19c18 0.929
##  3 r19c10 r19c9  0.926
##  4 r10c19 r9c19  0.924
##  5 r0c1   r0c0   0.923
##  6 r0c2   r0c1   0.922
##  7 r10c18 r9c18  0.914
##  8 r10c0  r9c0   0.914
##  9 r19c2  r19c1  0.913
## 10 r0c19  r0c18  0.913
```

```r
library(caTools)

set.seed(777)
sample = sample.split(SDATA$font,SplitRatio = 0.8)

print("SDATA proprtions")

## [1] "SDATA proprtions"

rbind(table(SDATA$font),
    table(SDATA$font)/dim(SDATA)[1])

##              1          2          3
## [1,] 3208.0000000 2285.0000000 2348.0000000
## [2,]    0.4091315    0.2914169    0.2994516

training_set = subset(SDATA,sample==TRUE)
test_set = subset(SDATA,sample==FALSE)

cat("\n \n training_set proprtions\n")

##
##
##  training_set proprtions

rbind(table(training_set$font),
    table(training_set$font)/dim(training_set)[1])

##              1          2          3
## [1,] 2566.0000000 1828.0000000 1878.000000
## [2,]    0.4091199    0.2914541    0.299426

cat("\n \n test_set proprtions\n")

##
##
##  test_set proprtions

rbind(table(test_set$font),
    table(test_set$font)/dim(test_set)[1])

##             1         2         3
## [1,] 642.0000000 457.0000000 470.0000000
## [2,]   0.4091778   0.2912683   0.2995539

accuracy <- function(x){
  x = as.matrix(x)
  sum(diag(x)/(sum(rowSums(x)))) * 100}


y_pred_train = knn(train = training_set[,-1],
            test = training_set[,-1],
            cl = training_set$font,
```

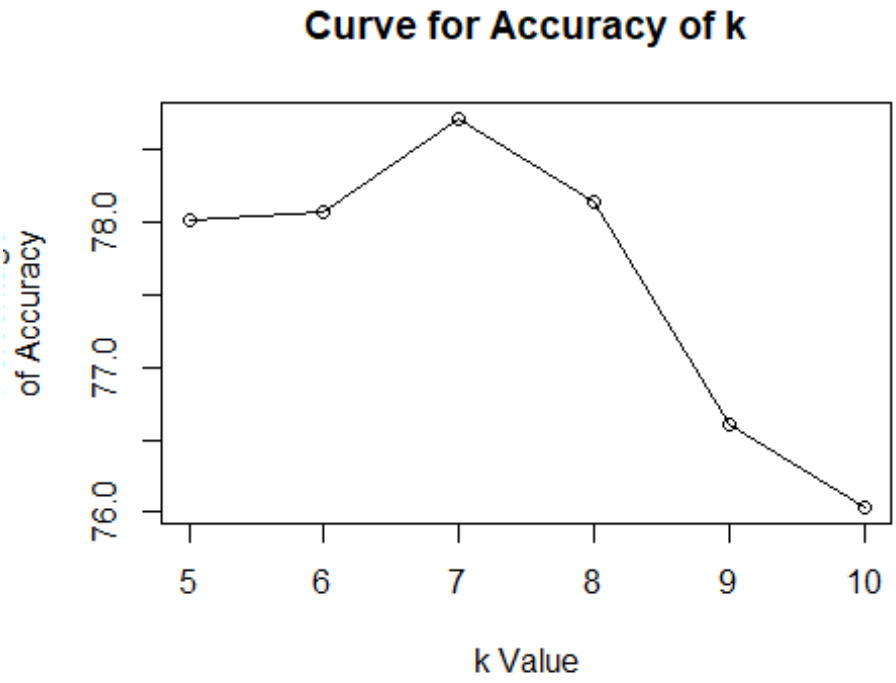
```
            k=12)

y_pred_test = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set$font,
            k=12)

# tab = table(y_pred,test_set$font)

cat("\n Training performance \n\n")

##
##  Training performance

table(y_pred_train,training_set$font) %>% accuracy()

## [1] 80.69196

cat("\n Test performance \n")

##
##  Test performance

table(y_pred_test,test_set$font) %>% accuracy()

## [1] 75.65328

y_pred_list = c()
k_list = c(5,10,15,20,30,40,50,100,200)
set.seed(1)

for (i in 1:length(k_list)){
  y_pred = knn(train = training_set[,-1],
               test = test_set[,-1],
               cl = training_set$font,
               k = k_list[i])
  tab = table(y_pred,test_set$font)
   y_pred_list[i] = accuracy(tab)
   print(i)
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
```

```
plot(k_list, y_pred_list, xlab = 'k Value', ylab = 'Percentage
of Accuracy',
main = 'Curve for Accuracy of k', type = 'o')
```




```
y_pred_list_2 = list()
k_list = seq(5,10)
set.seed(1)

for (i in k_list){
  y_pred = knn(train = training_set[,-1],
               test = test_set[,-1],
               cl = training_set$font,
               k =i)

  tab = table(y_pred,test_set$font)
   # y_pred_list_2[i] = accuracy(tab)
  y_pred_list_2 = append(y_pred_list_2,accuracy(tab))
   print(i)
}

## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```
plot(k_list, y_pred_list_2, xlab = 'k Value', ylab = 'Percentage
of Accuracy',
main = 'Curve for Accuracy of k', type = 'o')
```




```
set.seed(10)
y_pred_best = knn(train = training_set,
                  test = test_set,
                  cl = training_set$font,
                  k=7)
y_pred_trnbest = knn(train = training_set[,-1],
                  test = training_set[,-1],
                  cl = training_set$font,
                  k=7)
mat2 = table(y_pred_best,test_set$font)
tab2 = mat2/rowSums(mat2)
mat3 = table(y_pred_trnbest,training_set$font)
tab3 = mat3/rowSums(mat3)

# 1.5 90% confidence intervals
tab2[1,1] + c(-1,1)*qnorm(0.95)*sqrt(tab2[1,1]*(1-tab2[1,1])/length(test_set))

## [1] 0.7780579 0.8424708

tab2[2,2] + c(-1,1)*qnorm(0.95)*sqrt(tab2[2,2]*(1-tab2[2,2])/length(test_set))

## [1] 0.7171552 0.7880423

tab2[3,3] + c(-1,1)*qnorm(0.95)*sqrt(tab2[3,3]*(1-tab2[3,3])/length(test_set))
```

```
## [1] 0.8395600 0.8952715

subset_df = function(data = df,L=seq(0,9),M=seq(0,9)){
        a = strsplit(gsub("([a-z]*)([0-9]*)([a-z]*)", "\\1 \\2 \\3", colnames(data)), " ")
        index = list()
        sequence1 = seq(length(colnames(data)))
        for(i in sequence1){
           if( (a[[i]][2] %in% L) & (a[[i]][4] %in% M) ){

             index = c(index,paste(a[[i]],collapse = ''))
  }
 }

        data[,c("font",unlist(index))]
}
#PACK1
training_set_P1 = subset_df(data = training_set,L=seq(0,9),M=seq(0,9))
test_set_P1 = subset_df(data = test_set,L=seq(0,9),M=seq(0,9))
y_pred_P1 = knn(train = training_set_P1,
          test = test_set_P1,
          cl = training_set_P1$font,
          k=7)
w1 = table(y_pred,test_set_P1$font) %>% accuracy()
w1

## [1] 76.03569

#PACK2
training_set_P2 = subset_df(data = training_set,L=seq(0,19),M=seq(10,19))
test_set_P2 = subset_df(data = test_set,L=seq(0,19),M=seq(10,19))
y_pred_P2 = knn(train = training_set_P2,
          test = test_set_P2,
          cl = training_set_P2$font,
          k=7)
w2 = table(y_pred_P2,test_set_P2$font) %>% accuracy()
w2

## [1] 77.56533

#PACK3
training_set_P3 = subset_df(data = training_set,L=seq(10,19),M=seq(10,19))
test_set_P3 = subset_df(data = test_set,L=seq(10,19),M=seq(10,19))
y_pred_P3 = knn(train = training_set_P3,
          test = test_set_P3,
          cl = training_set_P3$font,
          k=7)
w3 = table(y_pred_P3,test_set_P3$font) %>% accuracy()
w3

## [1] 79.60484
```

```r
#PACK4
training_set_P4 = subset_df(data = training_set,L=seq(10,19),M=seq(0,9))
test_set_P4 = subset_df(data = test_set,L=seq(10,19),M=seq(0,9))
y_pred_P4 = knn(train = training_set_P4,
          test = test_set_P4,
          cl = training_set_P4$font,
          k=7)
w4 = table(y_pred_P4,test_set_P4$font) %>% accuracy()
w4
```

```
## [1] 78.71256
```

```r
W1 = w1/sum(w1+w2+w3+w4)
W2 = w2/sum(w1+w2+w3+w4)
W3 = w3/sum(w1+w2+w3+w4)
W4 = w4/sum(w1+w2+w3+w4)

cat("W1 :",W1," W2:", W2," W3:", W3," W4:", W4)
```

```
## W1 : 0.2437679  W2: 0.2486718  W3: 0.2552105  W4: 0.2523498
```

```r
subset_df(data = training_set,L=seq(0,19),M=seq(0,19))%>%  head()
```

```
## # A tibble: 6 x 401
##   font  r0c0[,1] r0c1[,1] r0c2[,1] r0c3[,1] r0c4[,1] r0c5[,1] r0c6[,1] r0c7[,1]
##   <fct>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 1        1.75     1.55     0.244  -0.0907    1.38     1.30     1.13     -1.04
## 2 1       -0.640   -0.714   -0.767  -0.322     0.594    1.30     1.26      1.22
## 3 1       -0.640   -0.714   -0.767  -0.322     0.594    1.30     1.26      1.22
## 4 1       -0.640   -0.714   -0.767  -0.828    -0.785   -0.503    0.298     1.17
## 5 1       -0.640   -0.714   -0.767  -0.828    -0.420    0.346    1.08      1.22
## 6 1        1.75     1.55     0.740  -0.828    -0.883   -0.945   -1.01     -1.04
## # ... with 392 more variables: r0c8[,1] <dbl>, r0c9[,1] <dbl>, r0c10[,1] <dbl>,
## #   r0c11[,1] <dbl>, r0c12[,1] <dbl>, r0c13[,1] <dbl>, r0c14[,1] <dbl>,
## #   r0c15[,1] <dbl>, r0c16[,1] <dbl>, r0c17[,1] <dbl>, r0c18[,1] <dbl>,
## #   r0c19[,1] <dbl>, r1c0[,1] <dbl>, r1c1[,1] <dbl>, r1c2[,1] <dbl>,
## #   r1c3[,1] <dbl>, r1c4[,1] <dbl>, r1c5[,1] <dbl>, r1c6[,1] <dbl>,
## #   r1c7[,1] <dbl>, r1c8[,1] <dbl>, r1c9[,1] <dbl>, r1c10[,1] <dbl>,
## #   r1c11[,1] <dbl>, r1c12[,1] <dbl>, r1c13[,1] <dbl>, r1c14[,1] <dbl>,
## #   r1c15[,1] <dbl>, r1c16[,1] <dbl>, r1c17[,1] <dbl>, r1c18[,1] <dbl>,
## #   r1c19[,1] <dbl>, r2c0[,1] <dbl>, r2c1[,1] <dbl>, r2c2[,1] <dbl>,
## #   r2c3[,1] <dbl>, r2c4[,1] <dbl>, r2c5[,1] <dbl>, r2c6[,1] <dbl>,
## #   r2c7[,1] <dbl>, r2c8[,1] <dbl>, r2c9[,1] <dbl>, r2c10[,1] <dbl>,
## #   r2c11[,1] <dbl>, r2c12[,1] <dbl>, r2c13[,1] <dbl>, r2c14[,1] <dbl>,
## #   r2c15[,1] <dbl>, r2c16[,1] <dbl>, r2c17[,1] <dbl>, r2c18[,1] <dbl>,
## #   r2c19[,1] <dbl>, r3c0[,1] <dbl>, r3c1[,1] <dbl>, r3c2[,1] <dbl>,
## #   r3c3[,1] <dbl>, r3c4[,1] <dbl>, r3c5[,1] <dbl>, r3c6[,1] <dbl>,
## #   r3c7[,1] <dbl>, r3c8[,1] <dbl>, r3c9[,1] <dbl>, r3c10[,1] <dbl>,
## #   r3c11[,1] <dbl>, r3c12[,1] <dbl>, r3c13[,1] <dbl>, r3c14[,1] <dbl>,
## #   r3c15[,1] <dbl>, r3c16[,1] <dbl>, r3c17[,1] <dbl>, r3c18[,1] <dbl>,
## #   r3c19[,1] <dbl>, r4c0[,1] <dbl>, r4c1[,1] <dbl>, r4c2[,1] <dbl>,
## #   r4c3[,1] <dbl>, r4c4[,1] <dbl>, r4c5[,1] <dbl>, r4c6[,1] <dbl>,
```

```
## #   r4c7[,1] <dbl>, r4c8[,1] <dbl>, r4c9[,1] <dbl>, r4c10[,1] <dbl>,
## #   r4c11[,1] <dbl>, r4c12[,1] <dbl>, r4c13[,1] <dbl>, r4c14[,1] <dbl>,
## #   r4c15[,1] <dbl>, r4c16[,1] <dbl>, r4c17[,1] <dbl>, r4c18[,1] <dbl>,
## #   r4c19[,1] <dbl>, r5c0[,1] <dbl>, r5c1[,1] <dbl>, r5c2[,1] <dbl>,
## #   r5c3[,1] <dbl>, r5c4[,1] <dbl>, r5c5[,1] <dbl>, r5c6[,1] <dbl>,
## #   r5c7[,1] <dbl>, ...

training_set_final = subset_df(data = training_set,L=seq(0,19),M=seq(0,19)) %>%
                  mutate_at(vars(r0c0:r9c9),.funs = funs(.*W1)) %>%
                  mutate_at(vars(r0c10:r9c19),.funs = funs(.*W2)) %>%
                  mutate_at(vars(r10c19:r10c19),.funs = funs(.*W3)) %>%
                  mutate_at(vars(r10c0:r19c9),.funs = funs(.*W4))

test_set_final = subset_df(data = test_set,L=seq(0,19),M=seq(0,19)) %>%
                  mutate_at(vars(r0c0:r9c9),.funs = funs(.*W1)) %>%
                  mutate_at(vars(r0c10:r9c19),.funs = funs(.*W2)) %>%
                  mutate_at(vars(r10c19:r10c19),.funs = funs(.*W3)) %>%
                  mutate_at(vars(r10c0:r19c9),.funs = funs(.*W4))
y_pred_final = knn(train = training_set_final,
             test = test_set_final,
             cl = training_set_final$font,
             k=7)


# confusion matrix for test_set
mat_fin = table(y_pred_final,test_set_final$font)
tab_fin = mat_fin/rowSums(mat_fin)

table(y_pred_final,test_set_final$font) %>% accuracy()

## [1] 88.71893
```