Report

Name: Ying Xu
E-mail: xuying1991@hotmail.com
Last four digits of student ID: 9859

Question1:

a. The purpose of KLExpand function is to have eigen values, eigen vectors and feature vectors as outputs. Input for this function should be a data matrix with measurements (e.g. blood test parameters, gene expression levels) as columns and observations (e.g. patients) as rows. The first step in this function is to get row numbers which is the number of observations. Then I calculated covariance matrix by using crossprod function which is basically: (data matrix) ^{T} *(data matrix) / (number of observations). After that, I used this covariance matrix as input for eigen function in R, which calculates eigen values and eigen vectors. Then the calculation of feature vectors is done by multiplying original normalized data matrix with eigen vectors. Since data matrix has patients in rows and eigen vector matrix has engen vectors in column, so by doing this I am finding the correlation between patients and eigen vectors, which is the definition of feature value. Finally, since it is not possible to return three values, making a list of these three values is a good way and easy to retrieve eigen values, eigen vectors and feature vectors when calling the function afterwards.

b. For the input data, I took out last two columns: Fhbf and Class, therefore my input data into KLExpand function is a 72x23 matrix. As output, I returned eigen vectors for the dimensionality of the feature space, which is a 23x23 matrix.

c. In order to do variable selection, I first calculated mu values, which is each eigen values / sum of all eigen values. Since eigen values are in descending order, so are mu values. Accumulative mu values are convenient for choosing threshold, and they are stored in variable: EV_contribution_percent (shown below). I set threshold at 98% so that 98% of the data will be included, and 16 variables will be selected. Since the number of parameters cannot be extremely bigger or smaller from number of patients, 16 variables are appropriate for the dimension of 72 patients.

```
> EV_contribution_percent
 [1] 0.2029606 0.3132193 0.4111505 0.5016609 0.5729575 0.6375370 0.6913171 0.7392804
 [9] 0.7819651 0.8227672 0.8603984 0.8924983 0.9182727 0.9402984 0.9589476 0.9743805
[17] 0.9870528 0.9939836 0.9979237 0.9993927 0.9997221 1.0000000 1.0000000
```

Then I used correlation function to get correlation between feature values and data set. The output communalities show how important each variable is, then I take the square of it to get rid of negative numbers and finally sort it in decreasing order as mu values. From mu values, I see that 98% of data needs top 16 parameters, so I took first 16 variables for building pattern recognition model. The variables I select and their communality values are shown as below.

```
$`variables I select`
       RBC        PCV         Hb        HbS        MCH        MCV        Age
0.89937940 0.82302398 0.74623676 0.71747814 0.29659999 0.22413020 0.16177367
      Nsex        BEN      Polys        WBC      Retic        BAN        SEN
0.15525089 0.12241855 0.10607918 0.09063568 0.07882567 0.07113847 0.04988064
       CAM        HbF
0.04711436 0.03974894
```

d. For the input into KLExpand function, I combined training and testing data set and took out the last class column, so the input data has dimension of 72x2000. I returned eigen vectors for the dimensionality of the feature space, which is a 2000x2000 matrix, but it's too big to get printed out, so I also put the dimension of feature space as an output.

e. Same calculation method is used as question2c, the accumulative sum of mu values is shown below (stored in variable: accu_mu). I set threshold at 90%, which will include first 41 variables. The reason for selecting 90% threshold is that in training and testing data set, there are 38 and 34 patients respectively, therefore a higher threshold including more parameters will not be an appropriate corresponding dimension as to patients. Hence, I choose a smaller subset of parameters. Adjusting parameters for a model is important, if accuracy is low for such subset, I will set a higher threshold to include more predictors and test again.

```
> accu_mu
  [1] 0.1857609 0.3103581 0.3755955 0.4265224 0.4723885 0.5024729 0.5321021 0.5579985
  [9] 0.5814187 0.6028979 0.6235417 0.6419712 0.6580168 0.6729276 0.6869228 0.6999066
 [17] 0.7121062 0.7235799 0.7349378 0.7455267 0.7557841 0.7652887 0.7744937 0.7834211
 [25] 0.7919614 0.8001208 0.8081775 0.8156432 0.8230012 0.8300579 0.8369592 0.8436708
 [33] 0.8502036 0.8565330 0.8627112 0.8686500 0.8745448 0.8801807 0.8857153 0.8912069
 [41] 0.8964280 0.9014549 0.9064251 0.9113097 0.9160288 0.9206239 0.9250599 0.9294150
 [49] 0.9335937 0.9377015 0.9416799 0.9455770 0.9494139 0.9531257 0.9566683 0.9600959
 [57] 0.9634197 0.9666111 0.9696507 0.9725948 0.9754798 0.9783086 0.9810859 0.9837793
 [65] 0.9864060 0.9889908 0.9913936 0.9936731 0.9958845 0.9980700 1.0000000 1.0000000
 [73] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [81] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [89] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [97] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[105] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[113] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[121] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[129] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[137] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[145] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

The variables I select and their communality values are shown as below.

```
$`variables I select`
         X98743_at          U54778_at          Z29505_at        M20867_s_at
         0.7216825          0.7004982          0.6879932          0.6871754
HG1322.HT5143_s_at        X05855_s_at          Z23064_at          X81198_at
         0.6829055          0.6707877          0.6704797          0.6612895
         X80199_at          D28476_at          X78627_at          M30938_at
         0.6584736          0.6563379          0.6456626          0.6454638
         X94232_at          M13450_at          M94630_at          X64330_at
         0.6374193          0.6340987          0.6315067          0.6304551
         M16342_at          L40410_at     M60858_rna1_at HG3076.HT3238_s_at
         0.6295752          0.6246873          0.6229878          0.6125890
         D87684_at   X75755_rna1_s_at          M91432_at          D80005_at
         0.6098540          0.6092505          0.6078634          0.6068044
         Z48042_at          D87078_at          X81003_at          X66397_at
         0.6060664          0.6013707          0.5997770          0.5985769
         M93651_at          U07857_at        M19311_s_at          D50911_at
         0.5952078          0.5952028          0.5951154          0.5949282
         L43631_at          M37033_at          M81601_at          D63878_at
         0.5921677          0.5914014          0.5911019          0.5905440
       AB004884_at          D14812_at          L20298_at          U81556_at
         0.5903049          0.5896005          0.5885849          0.5868232
         X60036_at
         0.5867744
```

Question2:

a. I used knn.cv function in R to do classification, because this uses leave-one-out cross validation. The input is each parameter (each column, 23 in total) as training set, true classification based on 15% final Hbf criteria as cl, k is equal to 7, I also set 'use.all=TRUE' to control handling of ties so that all distances equal to the 7^{th} largest are included. The output from knn.cv function is used as input for a function I wrote to calculate accuracy, false negative and false positive. The result for each parameter is stored in variable: model_result_hemo. Below is the average of 23 parameters:
   Accuracy average: 53.02% (stored in variable: mean_accu_hemo)
   False negative average: 28.16% (stored in variable: mean_fn_hemo)
   False positive average: 73.33% (stored in variable: mean_fp_hemo)

b. For leukemia dataset, since it has been split into training and testing already, I used knn function to do classification. This function requires four inputs: training data, which will be data frame of golubtrain.csv; testing data, which will be data frame of golubtest.csv; true classification of training set which is the last column of golubtrain.csv; k equals to 7. After knn function output, I calculated accuracy, false negative, false positive. For this dataset, I defined false negative as the percentage of patients classified as class 2 but are actually class 1, false positive as the percentage of patients classified as class 1 but are actually class 2. The accuracy, false negative and false positive are shown below:
   Accuracy: 91.18%
   False negative: 0%
   False positive: 92.86%