Part 1 Regression

1. How many predictors are there, i.e. what is p?

The number of predictors, i.e. *p* is 65. They are the original 11 features and 55 new pairwise interacted features.

```
> # load the data
> Credit = read.csv("D:/comp2020/424/Credit.csv",header=TRUE,row.names=1)
> X =model.matrix(Balance~.*., Credit)[,-1]
> y =Credit$Balance
> dim(X)
[1] 400  65
```

2. How did you generate your training and test sets?

In order to split the samples into a training set and a test set, we begin to set a random seed 12345 (RNG: random number generator), so that the results obtained will be reproducible. Then use the sample() function to split the set of observations sample() into two parts.

Take 1/2 of the data as training set and the rest data as test set (also could choose 2/3 or 3/4 as training data).

By selecting a random subset of 200 observations out of the original 400 observations, we refer to these observations as the training set. The rest ones are the test set.

```
> # load the data
> Credit = read.csv("D:/comp2020/424/Credit.csv",header=TRUE,row.names=1)
> dim(Credit)
[1] 400  11
> X =model.matrix(Balance~.*., Credit)[,-1]
> y =Credit$Balance
> # generate the training and test sets
> set.seed(12345)
> train = sample(1:nrow(X),nrow(X)/2)
> test = -train
> dim(Credit[train,])
[1] 200  11
> dim(Credit[test,])
[1] 200  11
```

Note that:

The model.matrix() function is used for building an "X" matrix from data.

The -train index below selects only the observations that are not in the training set.

3. Select the tuning parameter for the ridge regression model using cross-validation, and show the process.

The glmnet() function has an alpha argument that determines what type of model is fit. alpha=0 means to do ridge regression. Note that by default, the glmnet() function standardizes the variables so that they are on the same scale.

Set a random seed 12345, and k=10.

```
> library("glmnet")
> grid = 10^seq(3, -1, length=100)
> # cross-validation for ridge regression
> set.seed(12345)
> cv.out = cv.glmnet(X[train,],y[train],alpha=0,lambda=grid,nfolds=10,thresh = 1
e-10)
> cv.out$lambda.min
[1] 1.023531
```

The tuning parameter λ_{min} is 1.023531.

4. Select the tuning parameter for the lasso regression model using cross-validation, and show the process. How many features have been selected by the lasso?
In order to fit a lasso model, we once again use the glmnet() function; however, this time we use the argument alpha=1, which means to do lasso regression.
Set a random seed 12345, and k=10.

```
> # cross-validation for lasso
> set.seed(12345)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=10,thre
sh = 1e-10)
> cv.out$lambda.min
[1] 1.232847
```

The tuning parameter λ_{min} is 1.232847.

Features selected by the lasso are 26:

```
> bestlam=cv.out$lambda.min
> out=glmnet (X,y,alpha =1, lambda =grid)
> lasso.coef=predict (out,type ="coefficients",s=bestlam)[1:66,]
> lasso.coef[lasso.coef!=0]
                   (Intercept)                        Income
                 -2.216306e+02                 -2.736258e+00
                         Limit                         Cards
                  1.097916e-01                  1.024519e+01
                    StudentYes                 Income:Rating
                  1.503437e+02                 -1.163501e-02
                  Income:Cards                    Income:Age
                 -5.653834e-02                 -6.737762e-03
            Income:Education          Income:GenderFemale
                 -4.321065e-02                 -9.194263e-02
           Income:StudentYes                  Limit:Rating
                 -1.379862e+00                  2.641437e-04
             Limit:Cards               Limit:Education
                  1.024745e-03                  3.739166e-04
        Limit:GenderFemale             Limit:StudentYes
                  2.283491e-04                  7.133722e-02
      Limit:EthnicityAsian            Cards:GenderFemale
                  8.016329e-04                  3.662637e+00
         Cards:StudentYes      Cards:EthnicityCaucasian
                  3.426239e+00                  2.963773e+00
               Age:Education            Age:EthnicityAsian
                 -3.083158e-02                  9.465336e-02
    Education:GenderFemale          Education:StudentYes
                 -9.200723e-01                  1.881524e-01
 Education:EthnicityCaucasian        StudentYes:MarriedYes
                  7.575334e-01                  4.515216e+00
MarriedYes:EthnicityCaucasian
                 -7.114727e+00
```

```
> length(lasso.coef[lasso.coef!=0])-1
[1] 26
```

By using lasso.coef, it could be seen that 39 of the 65 coefficient estimates are exactly zero (In the Appendix). And the resulting coefficient estimates are sparse in lasso. Here, we just use lasso.coef[lasso.coef!=0] to show the selected features. From above result, we could see the lasso model with λ chosen by cross-validati on contains only 26 variables.

5. Compare and discuss the final form of the model from the linear regression, ridge regression, and lasso regression.

	The final form of the model
linear regression	RSS= $\sum_{i=1}^{200}(y_i - \beta_0 - \sum_{j=1}^{65} x_{ij}\beta_j)^2$
Ridge regression	RSS+ $1.023531\sum_{j=1}^{65} {\beta_j}^2$
lasso regression	RRS+ $1.232847\sum_{j=1}^{26} \lvert\beta_j\rvert$

Compare and discuss: linear regression estimates parameters by minimizing the Residual Sum of Squared (RSS) in a given set of data/observations. The estimation procedure is often called lest squares (coefficient) estimation. However, if the $\widehat{\beta}$- s are unconstrained, they can explode and hence are susceptible to very high variance. Compared with that, ridge regression and the lasso shrink the regression coefficients towards zero, both them are penalised methods for regression, but the form of the penalty term is different. Ridge regression adds a shrinkage penalty term (regularized L2 norm) to the calculation of minimizing RSS, while Lasso (Least absolute shrinkage and selection operator) adds an L1 norm as a penalty constraint in the calculation of RSS minimization. Here, $1.023531\sum_{j=1}^{65} {\beta_j}^2$ and $1.232847\sum_{j=1}^{26} \lvert\beta_j\rvert$ are shrinkage penalty terms of Ridge and lasso respectively.

6. Compare the test errors for the linear model, ridge regression model, and lasso model.
Set seed 12345.

```
> # linear regression
> set.seed(12345)
> linear.mod = lm(y[train]~X[train,])
> linear.pred = coef(linear.mod)[1]+X[test,] %*% coef(linear.mod)[-1]
> mean((linear.pred-y[test])^2)
[1] 7827.05
```

```
> bestlam=cv.out$lambda.min
> ridge.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((ridge.pred-y[test])^2)
[1] 5782.066
```

```
> bestlam=cv.out$lambda.min
> lasso.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((lasso.pred-y[test])^2)
[1] 4686.504
```

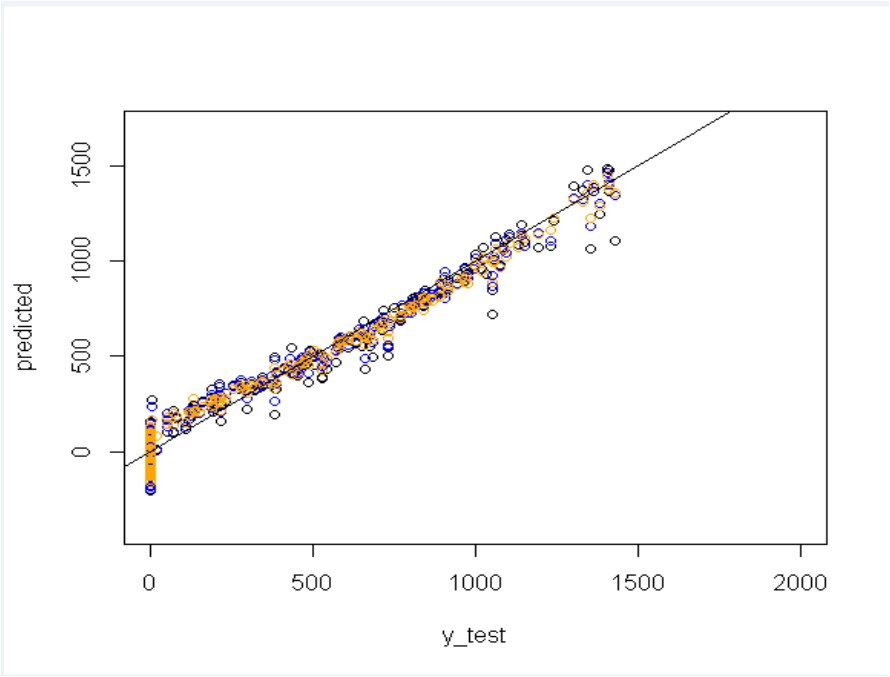
	The test errors of models
linear regression	7827.05
Ridge regression	5782.066
lasso regression	4686.504

Lasso regression model has the lowest test errors, 4686.504. The performance of the ridge regression is 5782.066, which is lower than lasso but higher than liner regression. Also, it can be seen that the fitting effect of the linear model is not very ideal with the highest test error, 7827.05. Lasso has the best performance, which maybe it does variable selection. Since the lasso can do automatic feature selection by setting some parameter estimates to zero. Therefore, lasso works well processing data with multicollinearity like this Credit data which has many pairwise interacted features.

7. Plot a comparison of the test predictions for the three approaches.

```
> #plot a comparison of the test predictions for three models
> plot(y[test],linear.pred,ylim=c(-400,1700),xlab="y_test",ylab="predicted")
> points(y[test],ridge.pred,col="blue")
> points(y[test],lasso.pred,col="orange")
> abline(0,1)
```

The plotting is as follows:



The test predictions: Linear model represents by black points. Blue dots are from the Ridge regression model and orange points are from Lasso model.

Part 2 Generalised additive models

1. What happens to the test mean squared error as the degrees of freedom of the natural spline for age is varied?
Set random seed 123456.
We choose the degrees of freedom (df) number from one to ten for age using the natural spline to show the result of the test mean squared error.

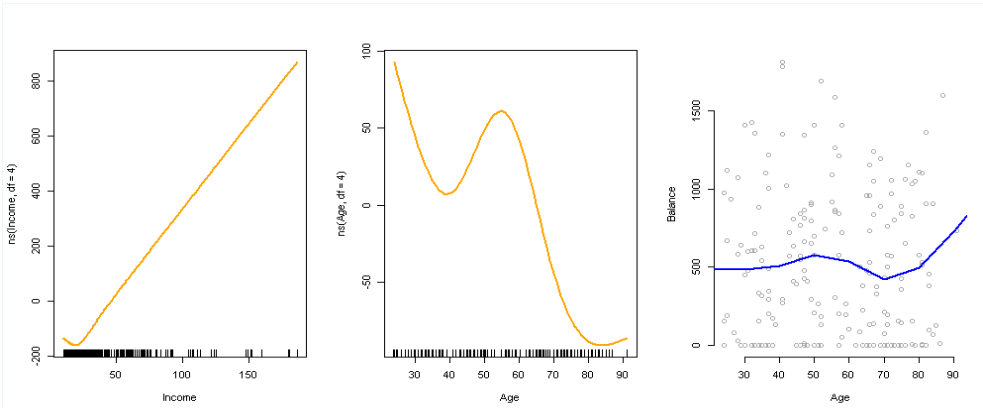
```
> set.seed(123456)
> library(gam)
> train = sample(1:nrow(Credit),nrow(Credit)/2)
> test = -train
> gam.mod1 = gam(Balance~ ns(Income,df=4)+ ns(Age,df=1)+Student,data=Credit[train,])
> gam.mod2 = gam(Balance~ns(Income,df=4)+ns(Age,df=2)+Student,data=Credit[train,])
> gam.mod3 = gam(Balance~ns(Income,df=4)+ns(Age,df=3)+Student,data=Credit[train,])
> gam.mod4 = gam(Balance~ ns(Income,df=4)+ ns(Age,df=4)+Student,data=Credit[train,])
> gam.mod5 = gam(Balance~ns(Income,df=4)+ns(Age,df=5)+Student,data=Credit[train,])
> gam.mod6 = gam(Balance~ns(Income,df=4)+ns(Age,df=6)+Student,data=Credit[train,])
> gam.mod7 = gam(Balance~ns(Income,df=4)+ns(Age,df=7)+Student,data=Credit[train,])
> gam.mod8 = gam(Balance~ns(Income,df=4)+ns(Age,df=8)+Student,data=Credit[train,])
> gam.mod9 = gam(Balance~ns(Income,df=4)+ns(Age,df=9)+Student,data=Credit[train,])
> gam.mod10 = gam(Balance~ns(Income,df=4)+ns(Age,df=10)+Student,data=Credit[train,])
> pred.mod1 = predict(gam.mod1,newdata=Credit[test,])
> pred.mod2 = predict(gam.mod2,newdata=Credit[test,])
> pred.mod3 = predict(gam.mod3,newdata=Credit[test,])
> pred.mod4 = predict(gam.mod4,newdata=Credit[test,])
> pred.mod5 = predict(gam.mod5,newdata=Credit[test,])
> pred.mod6 = predict(gam.mod6,newdata=Credit[test,])
> pred.mod7 = predict(gam.mod7,newdata=Credit[test,])
> pred.mod8 = predict(gam.mod8,newdata=Credit[test,])
> pred.mod9 = predict(gam.mod9,newdata=Credit[test,])
> pred.mod10 = predict(gam.mod10,newdata=Credit[test,])
> mse1 = mean((pred.mod1-Credit $Balance[test])^2)
> mse2 = mean((pred.mod2-Credit $Balance[test])^2)
> mse3 = mean((pred.mod3-Credit $Balance[test])^2)
> mse4 = mean((pred.mod4-Credit $Balance[test])^2)
> mse5 = mean((pred.mod5-Credit $Balance[test])^2)
> mse6 = mean((pred.mod6-Credit $Balance[test])^2)
> mse7 = mean((pred.mod7-Credit $Balance[test])^2)
> mse8 = mean((pred.mod8-Credit $Balance[test])^2)
> mse9 = mean((pred.mod9-Credit $Balance[test])^2)
> mse10 = mean((pred.mod10-Credit $Balance[test])^2)
> c(mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10)
 [1] 148591.0 148871.3 149093.1 147980.3 151120.5 154338.7 154394.8 155402.6
 [9] 156119.1 160770.3
```

```
> y1= c(mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10)
> x1= c(1,2,3,4,5,6,7,8,9,10)
> plot(x1,y1,type = "o",pch = 15,col = "red", main = "The Natural Spline",
+      xlab = "Degrees of freedom - Age",ylab = "The test mean squared error")
```

As from the above figure, the test mean squared error as the degrees of fre edom of the natural spline for age is varied. Error first slowly increases and the n decreases, reaching the minimum at df=4 (error: 147980.3), and after that, pre sents a rising trend along with the df increasing.

2. Choose the model with the lowest test error. Describe the effect of age in this model.

From previous results, we can see that gam.mod4 has the smallest test error, which is:
gam.mod4 = gam(Balance~ ns(Income, df=4)+ ns(Age, df=4)+ Student, data=Credit[train,])



From above figures, compared with the limit and ns plot, Age and ns plot is quite different, which has some fluctuation. When Age is between 38 and 58, the ns increases with age. After being over 80 years old, the ns also shows a slight increase in age. For other age groups (23-38, 58-80), the ns is negatively correlated with age. It could be seen in the last plot, the independent variable Age and responded Balance are indeed curvilinear relations.
As varied age, gam.mod4 shows to be the best model. The effect of age is that Age contributes nonlinearly to Balance.

3. How does the square root of the mean square error (RMSE) compare to the typical size/value of balance? Does your best model seem like a good model and why?

```
> sqrt(mse4)
[1] 384.682
> median(Credit$Balance)
[1] 459.5
```

This model does not seem like a good model. RMSE is 384.682, which is used to measure the deviation between the observed value and the predicted value. The typical size of balance here used is median, which is 459.5. RMSE is significant. Since the average error is used, and the average error is more sensitive to abnormal points, if the regression value of a certain point is unreasonable, its error is relatively large, so it will have a greater impact on the value of RMSE, that is, the average value is not robust.
While for the median, no matter how the maximum number or the minimum number changes in the data set, the median will not change, it is robust to outliers.

Part 3 Regression with high-dimensional data

1.Confirm that a linear model can fit the training data exactly and provide the evidence. Discuss this model is going to be useful or not and why?
Set seed: 123456789

```
> set.seed(123456789)
> Parkinsons = read.csv("D:/comp2020/424/parkinsons.csv",header=TRUE,row.names=
1)
> X =model.matrix(UPDRS~., Credit)[,-1]
> y =Parkinsons$UPDRS
> X = scale(X)
> train = sample(1:nrow(X), 30)
> test = -train
> dim(Parkinsons[train,])
[1] 30 98
> dim(Parkinsons[test,])
[1] 12 98
> linear.mod = lm(y[train]~X[train,])
> summary(linear.mod)

Call:
lm(formula = y[train] ~ X[train, ])

Residuals:
ALL 30 residuals are 0: no residual degrees of freedom!

Coefficients: (68 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)       168.4         NA      NA       NA
X[train, ]X1   -14256.8         NA      NA       NA
X[train, ]X2    13767.3         NA      NA       NA
X[train, ]X3    11991.6         NA      NA       NA
X[train, ]X4     2096.3         NA      NA       NA
X[train, ]X5    42266.9         NA      NA       NA
X[train, ]X6   -25667.1         NA      NA       NA
X[train, ]X7     1492.9         NA      NA       NA
X[train, ]X8    -5698.5         NA      NA       NA
X[train, ]X9     9569.4         NA      NA       NA
X[train, ]X10   -3151.3         NA      NA       NA
X[train, ]X11  -22713.0         NA      NA       NA
X[train, ]X12   10818.8         NA      NA       NA
X[train, ]X13 -513397.8         NA      NA       NA
X[train, ]X14 -481472.3         NA      NA       NA
X[train, ]X15 -121571.5         NA      NA       NA
X[train, ]X16  493831.0         NA      NA       NA
X[train, ]X17 -515015.0         NA      NA       NA
X[train, ]X18    7887.2         NA      NA       NA
X[train, ]X19   12136.4         NA      NA       NA
X[train, ]X20  -20373.3         NA      NA       NA
X[train, ]X21    1780.8         NA      NA       NA
X[train, ]X22    7785.6         NA      NA       NA
X[train, ]X23  -15534.0         NA      NA       NA
X[train, ]X24    4258.9         NA      NA       NA
X[train, ]X25  515097.5         NA      NA       NA
X[train, ]X26  486650.6         NA      NA       NA
X[train, ]X27  103917.0         NA      NA       NA
X[train, ]X28 -499297.7         NA      NA       NA
X[train, ]X29  512698.0         NA      NA       NA
X[train, ]X30        NA         NA      NA       NA
X[train, ]X31        NA         NA      NA       NA
X[train, ]X32        NA         NA      NA       NA
X[train, ]X33        NA         NA      NA       NA
X[train, ]X34        NA         NA      NA       NA
X[train, ]X35        NA         NA      NA       NA
X[train, ]X36        NA         NA      NA       NA
X[train, ]X37        NA         NA      NA       NA
X[train, ]X38        NA         NA      NA       NA
X[train, ]X39        NA         NA      NA       NA
X[train, ]X40        NA         NA      NA       NA
X[train, ]X41        NA         NA      NA       NA
X[train, ]X42        NA         NA      NA       NA
X[train, ]X43        NA         NA      NA       NA
X[train, ]X44        NA         NA      NA       NA
X[train, ]X45        NA         NA      NA       NA
```

```
x[train, ]x46        NA         NA       NA        NA
x[train, ]x47        NA         NA       NA        NA
x[train, ]x48        NA         NA       NA        NA
x[train, ]x49        NA         NA       NA        NA
x[train, ]x50        NA         NA       NA        NA
x[train, ]x51        NA         NA       NA        NA
x[train, ]x52        NA         NA       NA        NA
x[train, ]x53        NA         NA       NA        NA
x[train, ]x54        NA         NA       NA        NA
x[train, ]x55        NA         NA       NA        NA
x[train, ]x56        NA         NA       NA        NA
x[train, ]x57        NA         NA       NA        NA
x[train, ]x58        NA         NA       NA        NA
x[train, ]x59        NA         NA       NA        NA
x[train, ]x60        NA         NA       NA        NA
x[train, ]x61        NA         NA       NA        NA
x[train, ]x62        NA         NA       NA        NA
x[train, ]x63        NA         NA       NA        NA
x[train, ]x64        NA         NA       NA        NA
x[train, ]x65        NA         NA       NA        NA
x[train, ]x66        NA         NA       NA        NA
x[train, ]x67        NA         NA       NA        NA
x[train, ]x68        NA         NA       NA        NA
x[train, ]x69        NA         NA       NA        NA
x[train, ]x70        NA         NA       NA        NA
x[train, ]x71        NA         NA       NA        NA
x[train, ]x72        NA         NA       NA        NA
x[train, ]x73        NA         NA       NA        NA
x[train, ]x74        NA         NA       NA        NA
x[train, ]x75        NA         NA       NA        NA
x[train, ]x76        NA         NA       NA        NA
x[train, ]x77        NA         NA       NA        NA
x[train, ]x78        NA         NA       NA        NA
x[train, ]x79        NA         NA       NA        NA
x[train, ]x80        NA         NA       NA        NA
x[train, ]x81        NA         NA       NA        NA
x[train, ]x82        NA         NA       NA        NA
x[train, ]x83        NA         NA       NA        NA
x[train, ]x84        NA         NA       NA        NA
x[train, ]x85        NA         NA       NA        NA
x[train, ]x86        NA         NA       NA        NA
x[train, ]x87        NA         NA       NA        NA
x[train, ]x88        NA         NA       NA        NA
x[train, ]x89        NA         NA       NA        NA
x[train, ]x90        NA         NA       NA        NA
x[train, ]x91        NA         NA       NA        NA
x[train, ]x92        NA         NA       NA        NA
x[train, ]x93        NA         NA       NA        NA
x[train, ]x94        NA         NA       NA        NA
x[train, ]x95        NA         NA       NA        NA
x[train, ]x96        NA         NA       NA        NA
x[train, ]x97        NA         NA       NA        NA
```

```
Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:    NaN
F-statistic:   NaN on 29 and 0 DF,  p-value: NA
```

This model is not useful. We can see that here throws an exception: Residual standard error. The difference between these predicted values and the ones used to fit the model are called Residuals. When the residual standard error is exactly 0 then the model fits the data perfectly (likely due to over-fitting). However, we could see that the residual standard error and p-value are NA, which means the residual standard error cannot be shown to be significantly different from the variability in the unconditional response. Thus, it suggests the linear model has not any predictive ability. The reason would be insufficient sample size. In training dataset, it is high-dimensional data with 97 features, where it has 30 samples, 29 + 1 (intercept) = 30 variables, residuals are 0.

2.Now use the lasso to fit the training data, using leave-one-out cross-validation to find the tuning parameter λ. (You will find it convenient to set grid = 10^seq(3, -1; 100) and thresh=1e-10.) What is the optimal value of λ and what is the resulting test error?

Set seed 123456789.

```
> # lasso -LOOCV
> library("glmnet")
> grid = 10^seq(3, -1, length=100)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=30,thresh = 1e-1
0)
> cv.out$lambda.min
[1] 0.6428073
> bestlam=cv.out$lambda.min
> lasso.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((lasso.pred-y[test])^2)
[1] 20.54572
```

The optimal value of λ is 0.6428073 and the resulting test error is 20.54572.

3. State your final model for the UPDRS. How many features have been selected? What conclusions can you draw?

```
> out=glmnet (X,y,alpha =1, lambda =grid)
> lasso.coef=predict (out,type ="coefficients",s=bestlam)[1:98,]
> lasso.coef[lasso.coef!=0]
(Intercept)          X9         X83         X97
 26.6192659   0.2009743   0.5362753   9.1410291
```

y = 26.6192659 + 0.2009743*X9 + 0.5362753*X83 + 9.1410291*X97

Lasso model select three features, which are X9, X83 and X97. And we could see that X97 has significant coefficient score. This would be the most important feature.

4. Repeat your analysis with a different random split into training and test sets. Have the same features been selected in your final model?
We split the data set, with 35 instances in training set and 7 instances in test set.

```
> # a different random split
> train = sample(1:nrow(X), 35)
> test = -train
> dim(Parkinsons[train,])
[1] 35 98
> dim(Parkinsons[test,])
[1]  7 98
> linear.mod = lm(y[train]~X[train,])
> library("glmnet")
> grid = 10^seq(3, -1, length=100)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=30,thresh = 1e-1
0)
> cv.out$lambda.min
[1] 1.484968
> bestlam=cv.out$lambda.min
> lasso.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((lasso.pred-y[test])^2)
[1] 10.38167
> out=glmnet (X,y,alpha =1, lambda =grid)
> lasso.coef=predict (out,type ="coefficients",s=bestlam)[1:98,]
> lasso.coef[lasso.coef!=0]
(Intercept)         X97
  26.619266    8.463798
```

y = 26.619266 + 8.463798*X97

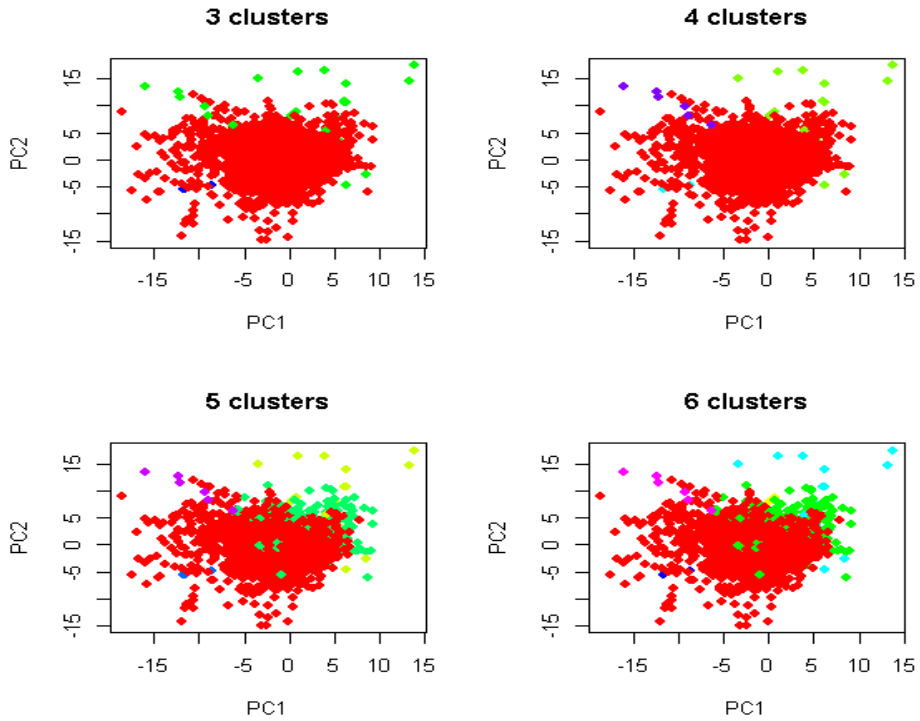
	Feature number	Feature name
Question3	3	X9, X83 and X97
Question4	1	X97

As can be seen from the above results, by using the final model used in Question 3, the selected features are different when a different random segmentation data set is selected. Three features (X9, X83 and X97) are selected in Question 3, while only one feature (X97) is selected in Question 4, which keeps the most important one.

Part 4 Clustering

1. Carry out hierarchical clustering with Euclidean distance and complete linkage. Describe the resulting clustering for 3-6 clusters. Hint: table() also works with only one argument. Plot the first 2 principal components against each other with the colour argument set equal to the cluster labels. What can you deduce/observe about the clustering?

```
> table(cutree(hc.complete,3))

   1    2    3
6792   35    3
> table(cutree(hc.complete,4))

   1    2    3    4
6792   29    3    6
> table(cutree(hc.complete,5))

   1    2    3    4    5
6559   29  233    3    6
> table(cutree(hc.complete,6))

   1    2    3    4    5    6
6559    7  233   22    3    6
```

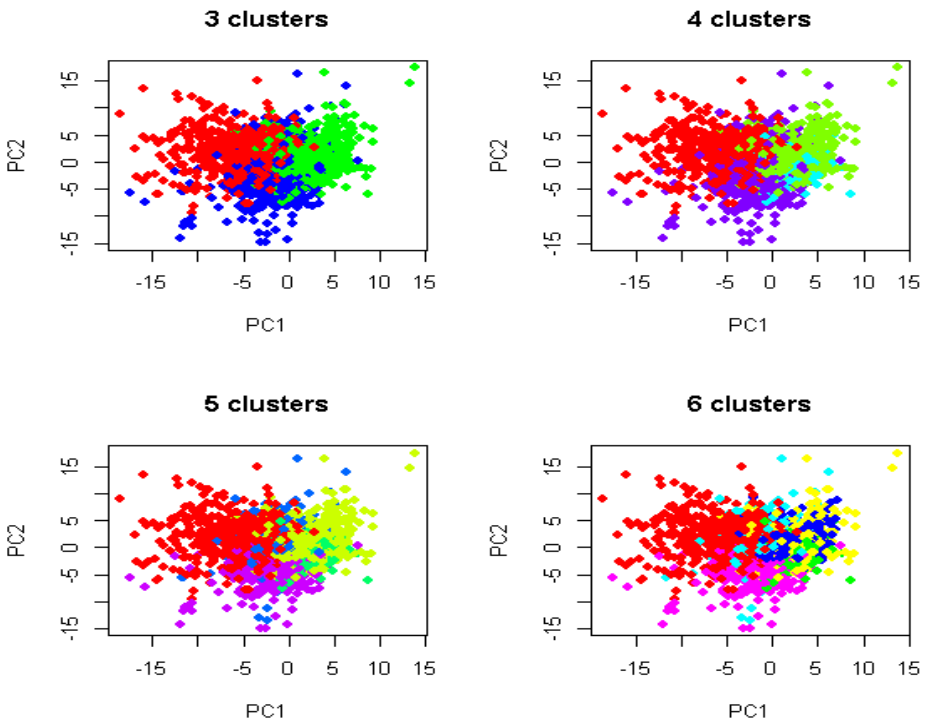


The above figure and results show the cluster class results generated by this method in different number of clusters (3-6). The results showed that most genes were concentrated in one cluster. The quality of complete hierarchical clustering is affected by the inability to adjust the merging or decomposition that has been done. It tends to look

for a compact class. When the number of clusters increases from 3 to 6, its result does not change much. The hierarchical clustering with Euclidean distance and complete linkage performs not well in this dataset.
We can see that Euclidean distance + complete linkage hierarchical clustering: this method takes the distance between the two farthest points in two sets as the distance between the two sets. The limit is very large. Even though the two clusters are very close, as long as there are points that do not match, they will not be merged. So it's not a good way.

2.Repeat the cluster analysis using correlation-based distance and complete linkage. You will need to use as.dist(), which converts a distance matrix into a form recognised by hclust(). As for the distance matrix itself, note that cor(t(X)) gives all pairwise correlations of the rows of X. Compare the clusters with those found above.

```
> table(cutree(hc2,3))

   1    2    3
1778 3991 1061
> table(cutree(hc2,4))

   1    2    3    4
1778 2656 1335 1061
> table(cutree(hc2,5))

   1    2    3    4    5
1778 2656 1335  385  676
> table(cutree(hc2,6))

   1    2    3    4    5    6
1778 1566 1335  385 1090  676
```

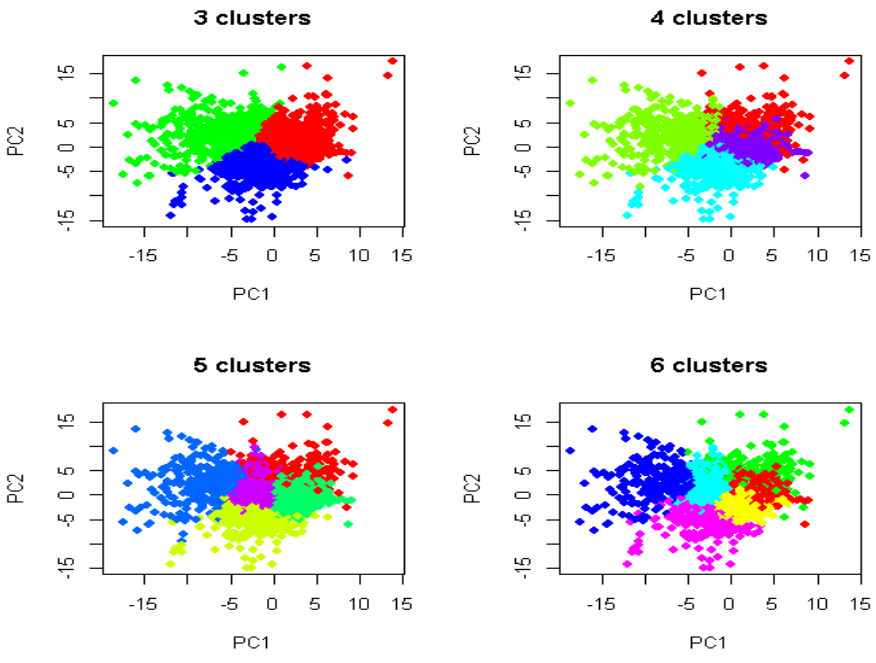


The difference between two hierarchical clustering methods, Euclidean distance + complete linkage and correlation-based distance + complete connection, use different definitions in dissimilarity measures. The former is based on Euclidean distance:

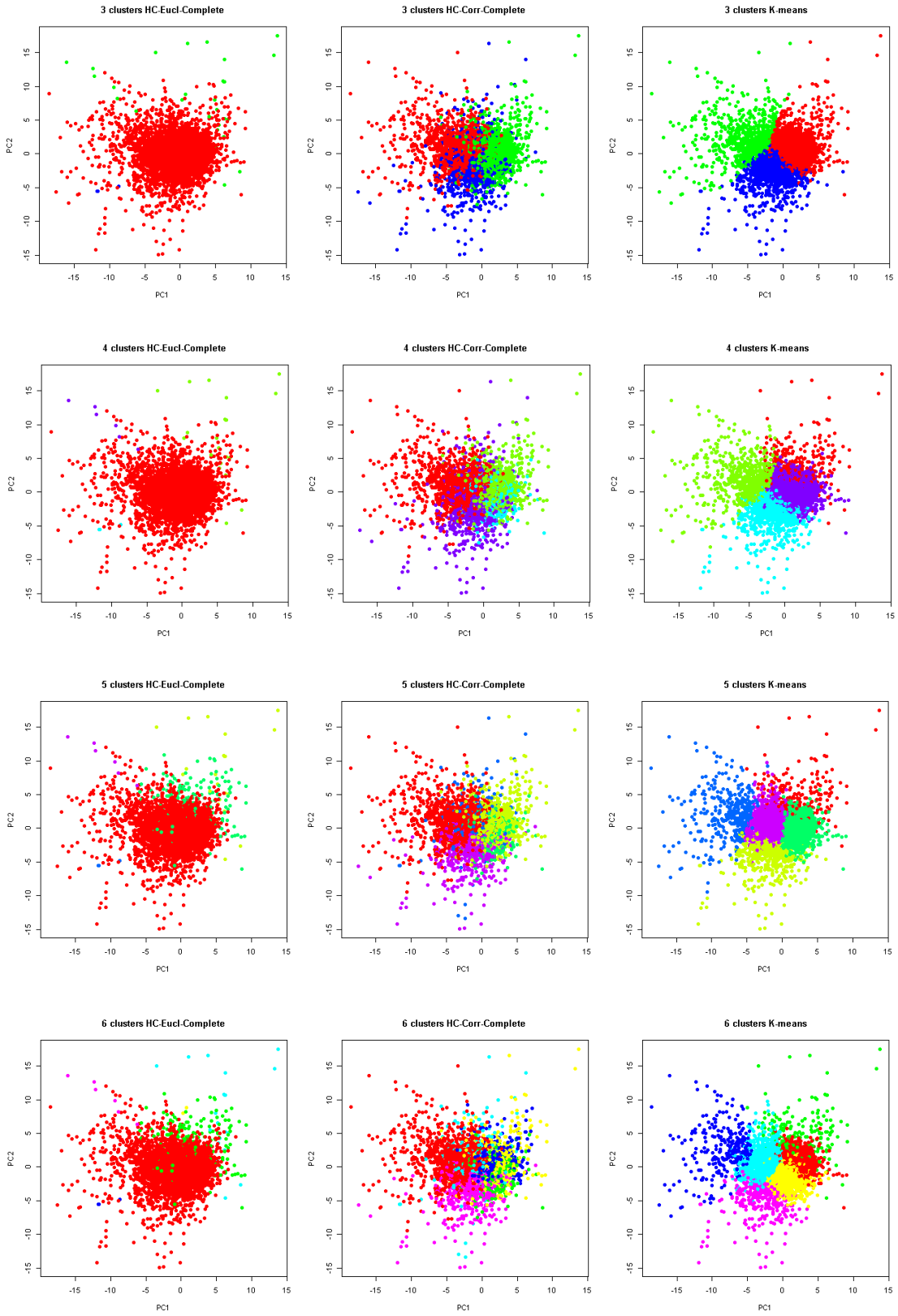
$\sqrt{(x-y)^2}$, while the latter uses Correlation-based distance:$1 - corr(x, y)$. Both of them uses complete linkage which is an extreme method, but the second method considers the use of relevance on this basis, so it performs better than the first method. It can be seen that compare with the hierarchical clustering with Euclidean distance, using correlation-based distance and complete linkage can make cluster classes have a better distribution, rather than focusing on a particular class.

3.Finally, carry out K-means clustering for 3 - 6 clusters. Compared the clusters of K-means with that of the above two approaches, find which of the hierarchical clusterings results is more similar to that of K-means?

```
> km3 = kmeans(X,3,nstart=50)
> table(km3$cluster)

   1    2    3
4668  763 1399
> km4 = kmeans(X,4,nstart=50)
> table(km4$cluster)

   1    2    3    4
 380  711 1034 4705
> km5 = kmeans(X,5,nstart=50)
> table(km5$cluster)

   1    2    3    4    5
 277  554 3790  390 1819
> km6 = kmeans(X,6,nstart=50)
> table(km6$cluster)

   1    2    3    4    5    6
2696 2047  247 1233  280  327
```



Here provides compared plots of three methods in different numbers of clusters as follows:



Comparison: The compared plots show that K-means gets better results than other two methods on this dataset. Each cluster is clearly well separated.

The goal of K-means algorithm is to divide n objects into K clusters with K as the parameter, so that the similarity within the clusters is high, while the similarity between clusters is low. K-means is to find the center, and then calculate the distance; Hierarchical clustering is to merge the samples one by one.

In K-means clustering algorithm, the number of clusters K needs to be given in advance. Hierarchical clustering does not specify the specific number of clusters, but only focuses on the distance between clusters, and eventually forms a tree chart.

The biggest advantage of hierarchical clustering is that it can get the whole clustering process at one time. As long as the above clustering tree is obtained, how many clusters you want to divide can get the results directly according to the tree structure. Changing the number of clusters does not need to calculate the attribution of data points again. The disadvantage of hierarchical clustering is that it needs a large amount of calculation, because it needs to calculate the distance of all data points in multiple clusters every time.

A potential drawback of hierarchical clustering is that clustering obtained by cutting the dendrogram at a certain height is necessarily nested within the clustering obtained by cutting at a greater height. Although hierarchical clustering does not need to determine the number of clusters, once a split or merge is implemented, it cannot be modified and the quality of clustering is limited.

In addition, the initial point selection of K-means is unstable and random, which leads to the instability of clustering results. It can be more sensitive to small changes in the data than the other two approaches.

K-means is more similar to hierarchical clustering with Euclidean distance and complete linkage.
Compared with the hierarchical clustering mentioned above, K-means algorithm has a big difference, that is, it needs the coordinates of data points, because it must take the average, while hierarchical clustering does not need coordinate data in fact, only needs to know the distance between data points. That is to say, K-means is only applicable to the case where Euclidean distance is used to calculate the similarity of data points, because if non-Euclidean distance is used, the cluster center cannot be obtained by simple average.

Part 5 Write a report

It is significate to evaluate application of machine learning in healthcare. Often a tradeoff of the models must be made between accuracy and intelligibility. More accurate models such as neural nets usually are not intelligible, but more intelligible models such as naive-Bayes often have significantly worse accuracy. To achieve the tradeoff, the paper presents two case studies where high-performance generalized additive models with pairwise interactions (GA^{2}Ms) are applied to real healthcare problems (e.g. predicting patients and reducing the cost of medical care in hospitals).

This paper first introduces models: Let $D = \{(x\mathrm{i}, y\mathrm{i})\}_1^N$ denote a training dataset of size N, where $x_i = (x_{i1}, \dots, x_{ip})$ is a feature vector with p features and y_i is the target (response). We use x_j to denote the jth variable in the feature space.
Standard GAMs (Generalized additive models) have the form:

$$g(E[y]) = \beta_0 + \sum f_j\,(x_j)$$

where g is the link function and for each term f_j, $E[f_j] = 0$.

GA^{2}Ms model: adds pairwise interactions on standard GAMs:

$$g(E[y]) = \beta_0 + \sum_j f_j(x_j) + \sum_{i \neq j} f_{ij}\,(x_i, x_j)$$

Note that pairwise interactions are used to improve accuracy, maintaining intelligible (can be visualized). GA^{2}Ms builds the best GAM first and then detects and ranks all possible pairs of interactions in the residuals and then choose the top k pairs.

The first case study is the pneumonia risk prediction. The GA^{2}M model discovered the same asthma pattern that created problems in the CEHC study, provides a simple mechanism to correct this problem, and uncovered other similar problems (chronic lung disease and history of chest pain) that were not recognized in the CEHC study but which warrant further investigation and probably repair. The paper suggests the GA^{2}M model is accurate, intelligible, and repairable.

The other is the 30-day hospital readmission case study. It applies a much larger dataset which contains much features for each patient. It just presents the top 6 terms for each patient (three patients showed) that contributed most to risk. Compared to an unintelligible model such as neural net, the model is fairly transparent, and its predictions can be fully understood whether on each patient or the whole level.

The paper illustrates that each term in the intelligible model returns a risk score. The average risk score for each term averaged across the training set is set to zero by subtracting the mean score. This makes models identifiable and modular for easier to interpret.
In addition, the GA^{2}M model could handle some of its accuracy by shaping continuous features which others (e.g. logistic regression) that could not achieve, and gains more accurately than expert discretization (Table 1). Compared with other approaches, the models

are intelligible enough to provide a window into the data and prediction problem, raise questions and require investigation and further data analysis.

In summary, GA^{2}Ms achieve state-of-the-art accuracy while remaining intelligible. It represents a significant step forward in the tradeoff between model accuracy and intelligibility that should make it easier to deploy high-accuracy learned models in applications where model verification and debuggability are as important as accuracy.

Model	**Pneumonia**	**Readmission**
Logistic Regression	0.8432	0.7523
GAM	0.8542	0.7795
GA^{2}M	0.8576	0.7833
Random Forests	0.8460	0.7671
LogitBoost	0.8493	0.7835

Table 1: AUC for different learning methods on two cases.