STAT 30250_Linear Models 2
HW 03.
ID: 16206265
Name: Minkun Kim

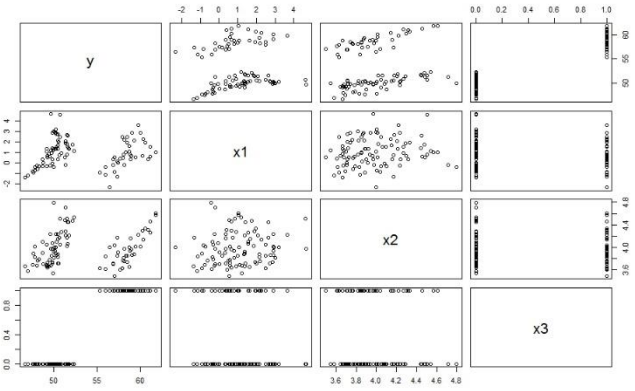
[Q.1]

We want to build the best multiple regression model to fit to the data set. Starting point would be investigating the predictors' contributions to a response variable, and discovering possible interactions between predictors. To finalize our model, it is also necessary to check the model assumptions and ensure the reliability of the model.

First, a correlation matrix helps us detect collinearity between predictors.

```
m.data = read.csv('C:/Users/Minkun/Dropbox/Minkun/Semester 2/30250_Linear Models ii/LAB/data/Q1data.csv', header = T)
m.data = m.data[-1]
head(m.data)
cor(m.data)
pairs(m.data)
```

```
           y          x1          x2          x3
y  1.00000000 0.06983305 0.16950318 0.94720307
x1 0.06983305 1.00000000 0.12688626 -0.11673049
x2 0.16950318 0.12688626 1.00000000 -0.05809609
x3 0.94720307 -0.11673049 -0.05809609 1.00000000
```

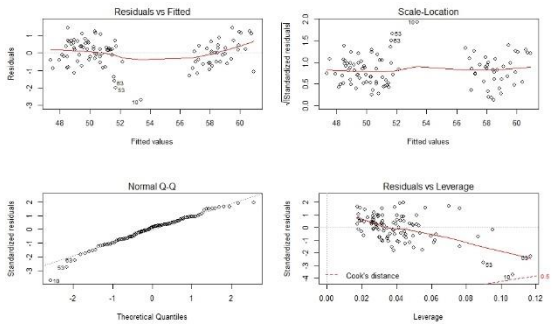


It seems that there is no correlation or interaction between regressors, but we observe non-linearity between predictors and the response variable. What we can notice first is the presence of a binary variable (x3), thus we treat it as a factor then fit the model to the data set.

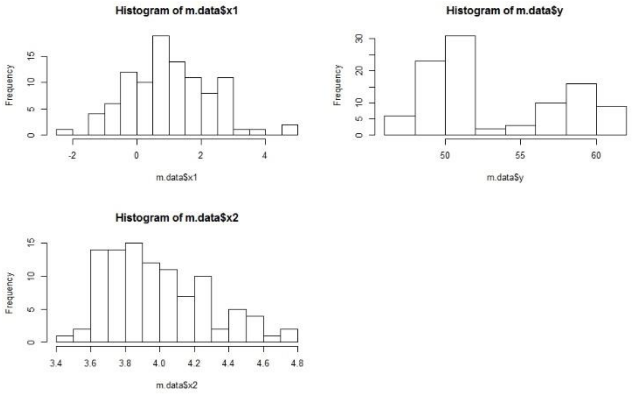
```
m.data$x3 = as.factor(m.data$x3)
is.factor(m.data$x3)
#a single predictor
reg_1 = lm(y ~ x1, data = m.data); summary(reg_1) # Rsq=0.004
reg_2 = lm(y ~ x2, data = m.data); summary(reg_2) # Rsq=0.028
reg_3 = lm(y ~ x3, data = m.data); summary(reg_3) # Rsq=0.897
# double predictors
reg_4 = lm(y ~ x1+x2, data = m.data); summary(reg_4) # Rsq=0.03
reg_5 = lm(y ~ x1+x3, data = m.data); summary(reg_5) # Rsq=0.93
reg_6 = lm(y ~ x2+x3, data = m.data); summary(reg_6) # Rsq=0.94
# triple predictors
reg_7 = lm(y ~ x1+x2+x3, data = m.data); summary(reg_7) # Rsq=0.97
av = aov(reg_7); summary(av)
```

```
            Df  Sum Sq  Mean Sq  F value   Pr(>F)
x1           1   9.6      9.6      16.70    9.07e-05 ***
x2           1  51.8     51.8      89.84    1.97e-15 ***
x3           1 1858.1  1858.1    3222.67   < 2e-16 ***
Residuals   96   55.4      0.6
```

```
layout(matrix(c(1,2,3,4), 2,2))
plot(av)
```




```
hist(m.data$x1, breaks = 10)
hist(m.data$x2, breaks = 10)
hist(m.data$y, breaks = 10)
```



We cannot find any significant issues from the diagnostic plot checking the non-Normality and and non-constant variance in the residuals of the model. Interestingly, their variance appears to be constant; therefore, we consider the transformation of the predictors rather than fitting a polynomial model, expecting the

transformation linearizes the relationship. More importantly, in the plot of a series of histograms, the distribution of 'x2' appears to be skewed to right so if we want to do statistical testing, it would be better to have normality. Accordingly, we consider square root or log - transforming 'x2' predictor. However, unfortunately, the two transformations did not make any significant improvement (manifesting similar R square-values) in dealing with the non-linearity residing between predictors and the response variable in the model, which is also stated by the AIC value.

```
m.data$log.x2 = log(m.data$x2)
hist(m.data$log.x2, breaks = 20)

m.data$sq.x2 = sqrt(m.data$x2)
hist(m.data$sq.x2, breaks = 20)

reg_8 = lm(y ~ x1+log.x2+x3, data = m.data); summary(reg_8) # Rsq=0.97
av = aov(reg_8); summary(av)
layout(matrix(c(1,2,3,4), 2,2))
plot(av)

reg_9 = lm(y ~ x1+sq.x2+x3, data = m.data); summary(reg_9) # Rsq=0.97
av = aov(reg_9); summary(av)
layout(matrix(c(1,2,3,4), 2,2))
plot(av)

fitnull <- lm(y~1, data=m.data); summary(fitnull)
fitfull <- lm(y~x1+x2+x3, data = m.data); summary(fitfull)
final<- step(fitfull, scope = list(lower=fitnull, upper=fitfull), direction='backward', trace = T);summary(final)
Start:  AIC=-51.15
y ~ x1 + x2 + x3
       Df  Sum of Sq    RSS      AIC
<none>                  55.35   -51.146
- x1   1    47.78      103.13    9.085
- x2   1    82.52      137.87   38.117
- x3   1  1858.14     1913.50  301.152

fitfull <- lm(y~x1+log.x2+x3, data = m.data); summary(fitfull)
final<- step(fitfull, scope = list(lower=fitnull, upper=fitfull), direction='backward', trace = T);summary(final)
Start:  AIC=-50.63
y ~ x1 + log.x2 + x3
         Df  Sum of Sq    RSS     AIC
<none>                  55.64  -50.628
- x1      1    46.87     102.51  8.482
- log.x2  1    82.24     137.87  38.117
- x3      1  1856.71    1912.35  301.092
```

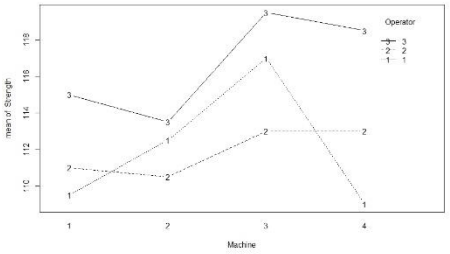
[Q.2]

This experiment should implement a mixed factor effect model because we are only interested in the outputs from the four machines which should be characterized as a fixed factor while three machine operators are randomly selected ,which should be considered as a random factor.

```
te.data = read.csv('C:/Users/Minkun/Dropbox/Minkun/Semester 2/30250_Linear Models ii/LAB/data/FiberStrength.csv', header = T)
colnames(te.data)[1] = 'Operator'
head(te.data)
summary(te.data)
attach(te.data)
Operater = as.factor(Operator)
Machine = as.factor(Machine)
is.factor(Machine)
aov = aov(Strength ~ Machine + Operator + Machine*Operator); summary(aov)
                    Df Sum Sq Mean Sq  F value   Pr(>F)
Machine              3  81.33   27.11    5.361 0.009523 **
Operator             1  85.56   85.56   16.919 0.000814 ***
Machine:Operator     3  42.19   14.06    2.781 0.074790 .
Residuals           16  80.92    5.06
```

Here we cannot rely on F value and P value because here R does not differentiate the fixed factor and random factor, but SS values are still valid; therefore, we can carry on the hypothesis test.

#Test H0: Interaction between Machine and Operator is 0?
F = MSAB/MSE = 14.06/5.06 = 2.778656 --------Definitely close to 1.
F(0.05), (3, 16) = 3.239----------therefore, we fail to reject H0, i.e An interaction effect is not significant.
#this can be seen through the interaction plot as well. The range of strength in this plot is too narrow to say all lines are not parallel.
with(te.data, interaction.plot(Machine,Operator,Strength, type = 'b'))



#Test H0: Fixed effect is 0?
F = MSA/MSAB = 27.11/14.06 = 1.928165------ Definitely close to 1.
F(0.05), (3, 3) = 9.277---------- therefore, we fail to reject H0, i.e there is no significant fixed effect.

#Test H0: Random effect is 0?
F = MSB/MSAB = 85.56/14.06 = 6.085349----- close to 1.
F(0.05), (1, 3) = 10.13----------therefore, we fail to reject H0, i.e there is no significant random effect.

This test does fully convince me because once there is no interaction effect discovered, we can investigate the source of the contribution to the fixed effect of Machine factor as well as the random effect of Operator factor. In other word, the variance of interaction term is not included in MSA of Machine factor and MSB of Operator factor.

Not to mention, we can estimate the value of fixed, random effect and the variances of interaction, fixed, random factor. The estimate of the variance of response variable would be equivalent to that of the variance of residual.

```
library(lme4)
fs.lmer = lmer(Strength ~ Machine + (1|Operator) + (1|Machine:Operator)); summary(fs.lmer)
anova(fs.lmer)
```

REML criterion at convergence: **97.9**

Random effects:

Groups	Name	Variance	Std.Dev.
Machine:Operator	(Intercept)	3.493	1.869
Operator	(Intercept)	6.111	2.472
Residual		**2.750**	1.658

Number of obs: 24, groups: Machine:Operator, 12; Operator, 3

Fixed effects:

	Estimate	Std. Error	t value
(Intercept)	111.8333	1.9130	58.46
Machine2	0.3333	1.8015	0.19
Machine3	4.6667	1.8015	2.59
Machine4	1.6667	1.8015	0.93

Analysis of Variance Table

	Df	Sum Sq	Mean Sq	F value
Machine	3	22.973	7.6576	2.7846

The value of random effect (maximum likelihood) is 97.9
The values of fixed effect is 111.8333, 0.3333, 4.6667, 1.6667 respectively.
The var of fixed factor is 1.91 and 1.8015.
The var of random factor 6.111
The var of interaction term is 3.493
Those values seem insignificant.

The estimate of the variance of response variable is only 2.75. Which one is the main source of this variance? We can determine it by comparing their significance of F-values. But as we've been through above, there is no significant random effect, fixed effect and interaction. If we need to reduce the variability in the response variable (the fiber strength), we can advise that in the case operator is a source of variation, "train operators," in the case machine is the source of variance, "calibrate the machine again."

STAT 30250_Monte Carlo Inference
HW 03.
ID: 16206265
Name: Minkun Kim

[Q.1]

We want to build the best multiple regression model to fit to the data set. Starting point would be investigating the predictors' contributions to a response variable, and discovering possible interactions between predictors. To finalize our model, it is also necessary to check the model assumptions and ensure the reliability of the model.

Notation: Let **'X'** be the RV denoting the no. of absences in a company for the past 12 months (as recorded by the HR department). Let us denote the frequencies of people with $\mathbf{X_i}$ absences by $\mathbf{f_i}$'s, i=0(1)10.

Given: **X~ P(λ)** and for the prior distribution of λ, we say **λ~ exp(θ) (or λ~G(a,b)** in later case**)**. Moreover, it has been given that the ***statistician is 75% sure that λ< 5***. Let us write the table given as follows:

No. of absences	0	1	2	3	4	5	6	7	8	9	10
Frequency	f_0	f_1	$f_2(13)$	$f_3(8)$	$f_4(3)$	$f_5(4)$	$f_6(3)$	$f_7(0)$	$f_8(0)$	$f_9(0)$	$f_{10}(1)$

where $f_0 + f_1 = 18$.

Assumption: The absence of one employer is not influenced by that of another.

Aim:

- ✓ To analyze the posterior distribution of λ.
- ✓ To approximately find the number of employees out of the sample of 50, f_0, who had no absences during the previous 12 months.

Analysis: It is clear from the data that f_0 is the missing data in this case and we need to find out the posterior distribution of λ and use Gibbs sampling to sample from that distribution using data augmentation techniques. Firstly, we need to calculate the value of the hyperparameter from the prior information given. As per the data,

$P_\theta(\lambda < 5) = 0.75.$

» $1 - e^{-5\theta} = 0.75$. (as we know the cdf of exp(θ) is given by $1 - e^{-x\theta}$)

» **θ = 0.277**

The **likelihood function** is given by $f(\lambda | f_0, f_1, f_2, \ldots, f_{10}) \propto \prod_{i=0}^{10} (P(Xi = i))^{fi}$

$$\propto \prod_{i=0}^{10} \left(e^{-\lambda}\lambda^i\right)^{fi}$$

$$\propto e^{-\lambda \sum_{i=0}^{10} fi} \lambda^{\sum_{i=1}^{10} i\, fi}$$

And we know that ***posterior ∝ likelihood * prior,***

$\pi(f_0, \lambda | (f_0 + f_1), f_2, \ldots, f_{10}) \propto f(\lambda | f_0, f_1, f_2, \ldots, f_{10}) * \pi(\lambda)$

$$\propto e^{-\lambda \sum_{i=0}^{10} fi} \lambda^{\sum_{i=1}^{10} i\, fi} * \theta e^{-\lambda\theta}$$

Now, we have the joint distribution of λ and f_0 and we calculate the conditional distribution to use Gibbs sampling.

- ***Posterior Conditional distribution of f_0 given λ, $(f_0 + f_1)$, f_2, …, f_{10} :***

We observe that we have 18 '0' or '1' occurrence of absence with 'p' and '1-p' probability. Clearly, the conditional distribution of f_0 will follow **bin(18, p)** with
1-p = P('0' occurrence of absence) = $e^{-\lambda}$ and
p = P('1' occurrence of absence) = $\lambda e^{-\lambda}$.
From here, we calculate p, $\frac{p}{1-p} = \lambda$

$$p = \frac{\lambda}{1+\lambda}$$

Hence, $\boldsymbol{\pi (f_0 \mid \lambda, (f_0 + f_1), f_2, \ldots, f_{10}) \sim bin\ (18, \frac{\lambda}{1+\lambda})}$

- ***Posterior Conditional distribution of f_0 given λ, $(f_0 + f_1)$, f_2, …, f_{10} :***

Now, $\pi(\lambda \mid f_0, f_1, f_2, \ldots, f_{10}) \propto e^{-\lambda(\sum_{i=0}^{10} fi+\theta)} \lambda^{\sum_{i=1}^{10} i\, fi}$

$$\propto e^{-\lambda(50+\theta)} \lambda^{128-f0}$$

Clearly,

$$\boldsymbol{\pi (\lambda \mid f_0, (f_0 + f_1), f_2, \ldots, f_{10}) \sim G\ (128\text{-}f_0\text{+}1,\ 50\text{+}\vartheta)}$$

Algorithm: Let us choose two initial values $\lambda^{(0)}$, $f_0^{(0)}$. The algorithm is as follows:

1) Conditional on the current parameter $\lambda^{(i)}$, generate a random value for $f_0^{(i+1)}$, from the posterior conditional distribution,

$$\boldsymbol{\pi (f_0^{(i+1)} \mid \lambda^{(i)}, (f_0 + f_1), f_2, \ldots, f_{10}) \sim bin\ (18, \frac{\lambda(i)}{1+\lambda(i)})}$$

2) Conditional on the newly updated parameter $f_0^{(i+1)}$, generate a random value for $\lambda^{(i+1)}$, from the posterior conditional distribution,

$$\boldsymbol{\pi (\lambda^{(i+1)} \mid f_0^{(i+1)}, (f_0 + f_1), f_2, \ldots, f_{10}) \sim G\ (128\text{-}f_0^{(i+1)}\text{+}1,\ 50\text{+}\vartheta)}$$

3) Return Step 1 and repeat the process unless the chain starts to burn-in and converges to a certain value.

R-code:

```
gibbs=function(it,l,f,th)
{
        l0=l;f0=f;
        ll=ff=vector();
        ll[1]=l;
        ff[1]=f;
        for(i in 2:(it+1))
        {
                ff[i]=rbinom(1,18,ll[i-1]/(1+ll[i-1]));
                ll[i]=rgamma(1,shape=129-ff[i],rate=50+th);
        }
        par(mfrow=c(2,2))
        plot(1:(it+1),ll,type='l',xlab='iterations',ylab='simulated lambda values',main='Gibbs sampling for the lambda');
        plot(1:(it+1),ff,type='l',xlab='iterations',ylab='simulated f0 values',main='Gibbs sampling for the f0');
        plot(density(ll),'Density plot for simulated lambda');
        hist(ff,freq=FALSE,main='Histogram for the simulated f0');
        mtext('The trace and density plots for lambda and f0',outer=TRUE,cex=1.5);
        list(ll,ff)
}
theta=-log(0.25)/5;
u=gibbs(500,5,7,theta)

# Calculating posterior mean and variance of the parameters
```

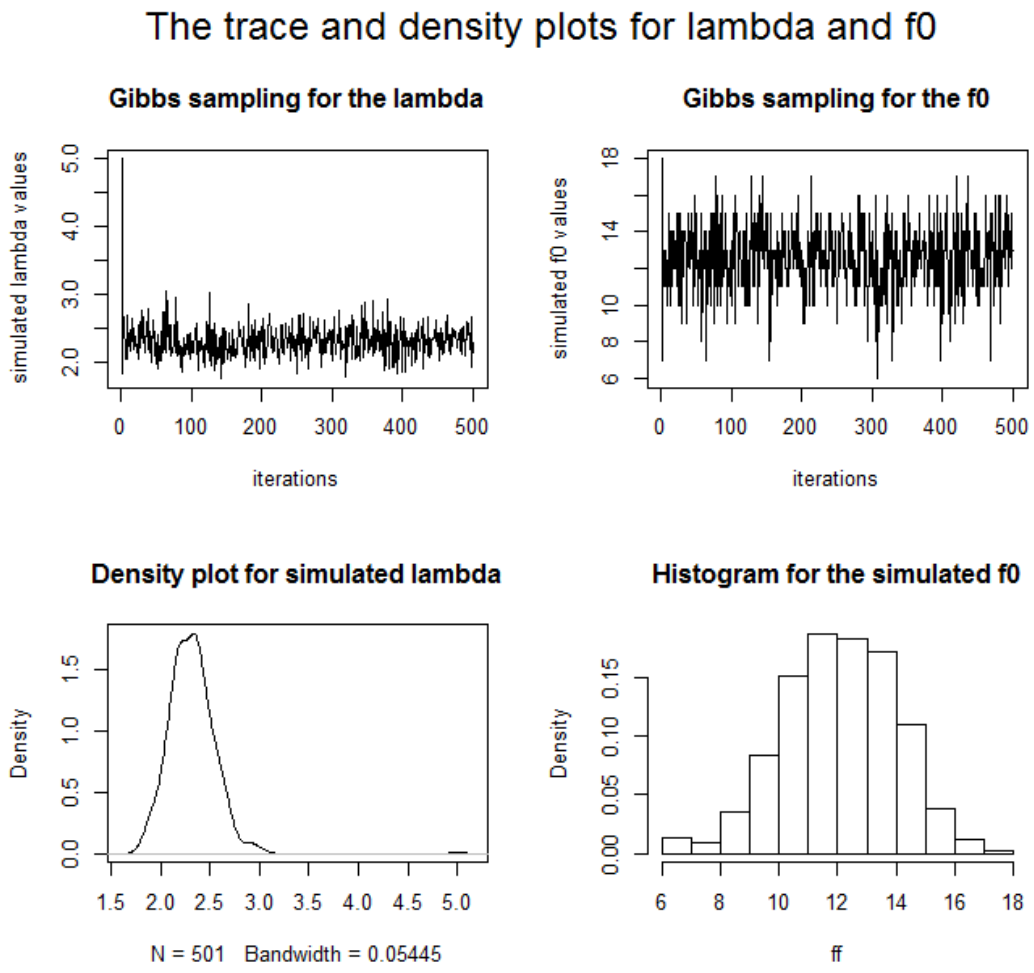
```
lll= u[[1]][-(0:30)];
fff= u[[2]][-(0:30)];
print(array(c(mean(lll),mean(fff),var(lll),var(fff)),c(2,2),dimnames=list(c('lambda','f0'),c('mean','var'))))
message('Credible interval for lambda: [',quantile(lll,0.025),',',quantile(lll,0.975),']')
message('Credible interval for f0: [',quantile(fff,0.025),',',quantile(fff,0.975),']')

# Test for autocorrelation
list(dwtest(lll[-1]~lll[-length(lll)]),dwtest(fff[-1]~fff[-length(fff)]))
```

Results:

	Mean	***Variance***
λ	2.326028	0.04648969
f_0	12.473461	3.63706013

Credible interval for λ: [1.94343148263778,2.73161612912025]
Credible interval for f_0: [9,16]

The **Durbin-Watson test** for autocorrelation revealed a p-value of **0.4904** in case of **λ** and a p-value of **0.4986** in case of **f_0**, which are both less than the level of significance (0.05). Hence, we fail to reject the null hypothesis that autocorrelation is zero, indicating the **chain has converged well to the target distribution**.



STAT 30270_Statistical Data Mining
HW 03. Classifier Accuracy
ID: 16206265
Name: Minkun Kim

Q. Work through various methods for assessing the performance of classifiers and implement these methods on the back pain dataset. Then choose the best classifier from at least three of those:
#A.classification tree
#B.bagging
#C.random Forest
#D.logistic regression
What do you think is an honest measure of classifier accuracy for this classifier on the data that you chose?

0> Intro

We have a categorical variable that consists of 3 classes.
Central Neuropathic: class.01
Nociceptive: class.02
Peripheral Neuropathic: class.03
Before we start, we need to deal with some missing values detected. In this case, random Forest does not work. Accordingly, those values need to be eliminated (N: 464 => 425)

1>**Comparing classifiers** (assessing performance on the same data)

A. classification trees

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	83	10	2
Nociceptive	3	220	12
Peripheral Neuropathic	1	9	85

```
sum(diag(tab))/sum(tab) # accurracy: 0.905 -> 0.913
[1] 0.9129412
```

B. bagging

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	84	3	1
Nociceptive	9	224	8
Peripheral Neuropathic	2	8	86

```
sum(diag(tab))/sum(tab) # accurracy:0.920 -> 0.927
[1] 0.9270588
```

C. random Forest

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	83	10	2
Nociceptive	3	220	12
Peripheral Neuropathic	1	9	85

```
sum(diag(tab))/sum(tab) # accurracy: NA -> 0.913
[1] 0.9129412
```

D. logistic regression

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	95	0	0
Nociceptive	0	235	0
Peripheral Neuropathic	0	0	95

```
sum(diag(tab))/sum(tab) # accurracy: 1.00 -> 1.00
[1] 1
```

Judging from the outputs above, their performance seems quite great, and notably, logistic regression classifier registers 100% accuracy. This seemingly overfitting issue stems from the fact that every data is used to build those classifiers. We know there are several methods to reassess those classifiers and build some confidence in their performance.

_A. General validation: we split the data into three parts – training 50%/validation25%/test25%, and build classifiers based on the training data. Then compare full prediction results yielded by the classifier with validation data and test data from the original so that we can compare their performances.

_B. Bootstrapping validation: We use a bootstrap sample as training data. When we do bootstraping, classifier is built on the dataset that has the same number of observations of the original data, and it does some of observations repeated, whereas in general splittting, classifier is built on smaller dataset. More importantly, the values bootstrapping is missing becomes the test data.

_C. K-fold cross validation: We divide the data into K groups and differentiate one of those groups (test data) from the rest of them (training data), then build the classifiers based on those K-1 groups and compare the prediction results with the test data. This process continues K times.

2>**General validation** (assessing performance on the validate, test data)

A. classification trees

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	17	2	3
Nociceptive	1	51	3
Peripheral Neuropathic	0	2	27

```
sum(diag(tab))/sum(tab) # accurracy: 0.896
[1] 0.8962264
```

B. bagging

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	17	1	0
Nociceptive	3	52	0
Peripheral Neuropathic	2	2	29

```
sum(diag(tab))/sum(tab) # accurracy:0.924
[1] 0.9245283
```

C. random Forest

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	17	2	3
Nociceptive	1	51	3
Peripheral Neuropathic	0	2	27

```
sum(diag(tab))/sum(tab) # accurracy: 0.896
[1] 0.8962264
```

D. logistic regression

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	16	2	4
Nociceptive	0	52	3
Peripheral Neuropathic	1	4	24

```
sum(diag(tab))/sum(tab) # accurracy: 0.868
[1] 0.8679245
```

E. As can be seen from above, bagging shows the best performance here, thus we try bagging on the test data to get an accurate assessment of its performance.

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	21	1	0
Nociceptive	3	59	4
Peripheral Neuropathic	0	2	17

```
sum(diag(tab))/sum(tab) # accurracy:0.906
[1] 0.9065421
```

As expected, in this data set, **bagging** shows the best performance; hence, we can say bagging is the best classification method for the 'back pain' data set.

3>**Bootstrapping validation** (assessing performance on the test data)

A. classification trees

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	21	9	1
Nociceptive	0	81	3
Peripheral Neuropathic	1	11	28

```
sum(diag(tab))/sum(tab)
[1] 0.8387097
```

B. bagging

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	25	0	1
Nociceptive	5	81	9
Peripheral Neuropathic	1	3	30

```
sum(diag(tab))/sum(tab)
[1] 0.8774194
```

C. random Forest

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	24	6	1
Nociceptive	0	80	4
Peripheral Neuropathic	2	4	34

```
sum(diag(tab))/sum(tab)
[1] 0.8903226
```

D. logistic regression

	Central Neuropathic	Nociceptive	Peripheral Neuropathic
Central Neuropathic	24	7	0
Nociceptive	1	76	7
Peripheral Neuropathic	0	10	30

```
sum(diag(tab))/sum(tab)
[1] 0.8387097
```

In this output, **random Forest** shows the best performance.

4>**K-fold cross validation** (assessing performance on the test data) **K=10**

```
table(folds)
folds
 1  2  3  4  5  6  7  8  9 10
43 42 43 42 42 43 43 42 43 42
```

A. classification trees

```
      [,1]
 [1,] 0.9302326
 [2,] 0.9285714
 [3,] 0.8837209
 [4,] 0.8571429
 [5,] 0.9523810
 [6,] 0.9302326
 [7,] 0.9302326
 [8,] 0.8571429
 [9,] 0.8837209
[10,] 0.9761905
```

```
colnames(res)<-c("test")
apply(res,2,summary)
          test
Min.    0.8571
1st Qu. 0.8837
Median  0.9294
Mean    0.9130
3rd Qu. 0.9302
Max.    0.9762
```

B. bagging

```
colnames(res)<-c("test")
apply(res,2,summary)
```

```
              [,1]
 [1,] 0.9302326
 [2,] 0.9285714
 [3,] 0.8837209
 [4,] 0.8571429
 [5,] 0.9523810
 [6,] 0.9302326
 [7,] 0.9302326
 [8,] 0.8571429
 [9,] 0.8837209
[10,] 0.9761905
```

```
                test
Min.    0.8571
1st Qu. 0.8837
Median  0.9294
Mean    0.9130
3rd Qu. 0.9302
Max.    0.9762
```

C. random Forest

```
colnames(res)<-c("test")
apply(res,2,summary)
```

```
              [,1]
 [1,] 0.9767442
 [2,] 0.9285714
 [3,] 0.8837209
 [4,] 0.9047619
 [5,] 0.9761905
 [6,] 0.9534884
 [7,] 0.9534884
 [8,] 0.8809524
 [9,] 0.9767442
[10,] 0.9761905
```

```
                test
Min.    0.8810
1st Qu. 0.9107
Median  0.9535
Mean    0.9411
3rd Qu. 0.9762
Max.    0.9767
```

D. logistic regression

```
colnames(res)<-c("test")
apply(res,2,summary)
```

```
              [,1]
 [1,] 0.8837209
 [2,] 0.8809524
 [3,] 0.7674419
 [4,] 0.8095238
 [5,] 0.8333333
 [6,] 0.8604651
 [7,] 0.8604651
 [8,] 0.9523810
 [9,] 0.9069767
[10,] 0.9285714
```

```
                test
Min.    0.7674
1st Qu. 0.8401
Median  0.8707
Mean    0.8684
3rd Qu. 0.9012
Max.    0.9524
```

In this output, **random Forest** shows the best performance.

All in all, it seems that random Forest is the best classifier to this data set, which is underpinned by those multiple validation procedures. When our data is subject to limited usage, bagging, interestingly, shows the greatest performance; however, once extending the scope of data set, and lifting the usage limit - using bootstrap or K-fold cross validation – we were able to obtain more reliable outcomes.

STAT 40150 Multivariate Analysis_Assignment 01.

ID: 16206265

Name: Minkun Kim

Q1. (a)

(i) Calculate the covariance matrix Σ for the nine chemicals:

```
> pot.data <- read.csv('C:/Users/Minkun/Dropbox/Minkun/Semester 2/40150_Multivariate A/assignment/a1/PotteryData.csv', header = T)
> View(pot.data)
> cov.pot <- cov(pot.data[, 1:9]); cov.pot
```

	Al2O3	Fe2O3	MgO	CaO	Na2O	K2O	TiO2	MnO	BaO
Al2O3	**7.306282828**	-0.907852020	-3.4490767677	0.2845131313	0.007860101	-1.40893182	0.3417348485	-7.171601e-02	2.533990e-03
Fe2O3	-0.907852020	**5.787928586**	1.6480894444	0.7242160101	0.288911162	1.26323182	-0.0630878788	7.544720e-02	1.515611e-03
MgO	-3.449076768	1.648089444	**3.0349497980**	-0.1470693434	0.046438687	1.29850227	-0.2151439394	6.389649e-02	-3.453232e-04
CaO	0.284513131	0.724216010	-0.1470693434	**0.2063688889**	0.041789495	0.02179773	0.0134666667	3.006551e-03	3.376869e-04
Na2O	0.007860101	0.288911162	0.0464386869	0.0417894949	**0.031771010**	0.04919091	0.0013598485	4.451444e-03	1.943990e-04
K2O	-1.408931818	1.263231818	1.2985022727	0.0217977273	0.049190909	**0.72714091**	-0.0926318182	3.394068e-02	1.775000e-04
TiO2	0.341734848	-0.063087879	-0.2151439394	0.0134666667	0.001359848	-0.09263182	**0.0323318182**	-4.573712e-03	1.269697e-04
MnO	-0.071716010	0.075447202	0.0638964949	0.0030065505	0.004451444	0.03394068	-0.0045737121	**2.190346e-03**	2.511919e-05
BaO	0.002533990	0.001515611	-0.0003453232	0.0003376869	0.000194399	0.00017750	0.0001269697	2.511919e-05	**8.891919e-06**

(ii) What are the first two eigenvalues and eigenvectors of Σ? Show that they are indeed eigenvalues and eigenvectors of Σ:

```
> result <- eigen(cov.pot); result
> result$values[1:2]
[1] 10.400328  5.542034

> result$vectors[,1:2]
              [,1]          [,2]
 [1,]  0.7549211696  0.4574027689
 [2,] -0.3833403463  0.8711194478
 [3,] -0.4802916309 -0.0186673666
 [4,]  0.0002777947  0.1439379027
 [5,] -0.0133263940  0.0480784193
 [6,] -0.2249741610  0.0905092573
 [7,]  0.0391953978  0.0179610666
 [8,] -0.0116967456  0.0063836149
 [9,]  0.0001403877  0.0004623287
```

We need to show **Σv = λv**

➔When v is Eigenvector [1], and λ is 10.400328,

```
> cov.pot%*%result$vectors[,1] #(9x9)(9x1)
Al2O3  7.851427768
Fe2O3 -3.986865332
MgO   -4.995190491
```

```
CaO   0.002889156
Na2O  -0.138598869
K2O   -2.339805063
TiO2  0.407644993
MnO   -0.121649990
BaO   0.001460078
> 10.400328%*%result$vectors[,1] #(9x1)
          [,1]      [,2]      [,3]        [,4]      [,5]      [,6]     [,7]      [,8]        [,9]
[1,] 7.851428 -3.986865 -4.99519 0.002889156 -0.1385989 -2.339805 0.407645 -0.12165 0.001460078
```

➔When v is Eigenvector [2], and λ is 5.542034,

```
> cov.pot%*%result$vectors[,2] #(9x9)(9x1)
Al2O3  2.534941665
Fe2O3  4.827773536
MgO   -0.103455179
CaO    0.797708741
Na2O   0.266452231
K2O    0.501605375
TiO2   0.099540840
MnO    0.035378210
BaO    0.002562241
> 5.542034%*%result$vectors[,2] #(9x1)
          [,1]      [,2]       [,3]      [,4]      [,5]      [,6]       [,7]       [,8]        [,9]
[1,] 2.534942 4.827774 -0.1034552 0.7977088 0.2664522 0.5016054 0.09954084 0.03537821 0.002562241
```

(iii) Verify that the first two eigenvectors are orthonormal:

➔Two eigenvectors U & V are orthogonal….. if t(U)%*%V = 0 = t(V)%*%U

```
> U=result$vectors[,1]; U
> V=result$vectors[,2]; V
> t(U)%*%V
              [,1]
[1,] -6.741391e-16
> t(V)%*%U
              [,1]
[1,] -6.741391e-16
```

➔Two eigenvectors U & V are **orthonormal**….. if they are orthogonal and t(U)%*%U = 1 = t(V)%*%V

```
> t(U)%*%U
     [,1]
[1,]    1
> t(V)%*%V
     [,1]
[1,]    1
```

(iv) Would you advise standardizing the pottery data prior to analysis? Explain your reasoning.

This dataset needs to be standardized before we carry out the analysis so that we can make the source data internally consistent. When carefully investigating the dataset, one can see that the magnitudes of values seem to be consistent when they only belong to the same variable. This results from the difference of the unit of measure that is used.

Q1. (b) E[X_1]=10, var[X_1]=11, E[X_2]=5, var[X_2]=4, cov(X_1, X_2)=2

(i) Calculate the expected value and variance of 2(X_1)−(X_2)

```
> E.x_1=10
> var.x_1=11
> E.x_2=5
> var.x_2=4
> cov.x_1_x_2=2
> 2*E.x_1 - E.x_2
[1] 15
```

```
> 4*var.x_1 + var.x_2 - 4*cov.x_1_x_2
[1] 40
```

(ii) Calculate the correlation of U =2(X_1)−(X_2) and V = (X_1) + 2(X_2)

```
> E.x_1sq=111
> E.x_2sq=29
> E.x_1.by.x_2=52
> E.U=15
> E.V=20
> cov.U.V = 2*E.x_1sq - 2*E.x_2sq + 3*E.x_1.by.x_2 - E.U*E.V
> var.U = 4*var.x_1 + var.x_2 - 4*cov.x_1_x_2
> var.V = var.x_1 + 4*var.x_2 + 4*cov.x_1_x_2
> cor.U.V = cov.U.V / sqrt(var.U*var.V); cor.U.V
[1] 0.5345225
```

Q2. (a) K-mean clustering

How many clusters would you suggest are present in this data set? Detail any decisions you make when running this procedure. Provide details of your reasoning and fully commented R code.

First, we need to drop unnecessary data in the dataset.

```
pot.data2 = pot.data[, -10]; pot.data2
```

We want to standardize the raw data prior to analysis.

```
SD <- apply(pot.data2, 2, sd); SD
```

We divide each column in our data matrix by its standard deviation..

```
sd.pot <- sweep(pot.data2, 2, SD, '/'); sd.pot
```

We want to set k=5 because we know the data come from 5 different areas(kiln).

```
WGSS = rep(0,5)
n = nrow(sd.pot)
```

if k=1, add up all SS...what we calculate here is variances so need to delete the bottom of var.

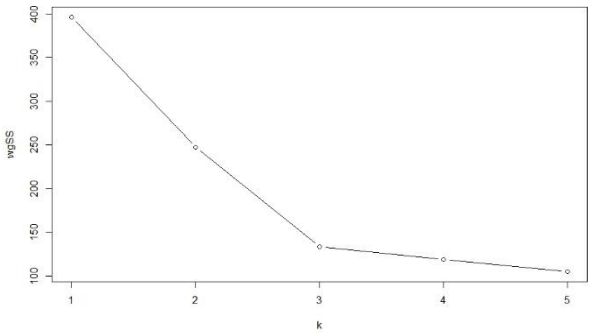
```
WGSS[1] = (n-1)*sum(apply(sd.pot, 2, var))
```

#WGSS was designed to store up each SS according to different k!

```
for(k in 2:5){
  WGSS[k] <- sum(kmeans(sd.pot, centers = k)$withinss) # Within cluster sum of squares by cluster..
}
```

#We want to plot k versus WGSS. we can find the best k that is shown by the elbow in the curve.

```
plot(1:5, WGSS, type = 'b', xlab = 'k', ylab = 'wgSS')
```



#the k versus WGSS plot suggests k = 3 is a good clustering solution.

#k=3 means cluster takes the dataset.

```
kmcl.sd.pot <- kmeans(sd.pot, centers = 3); kmcl.sd.pot
```

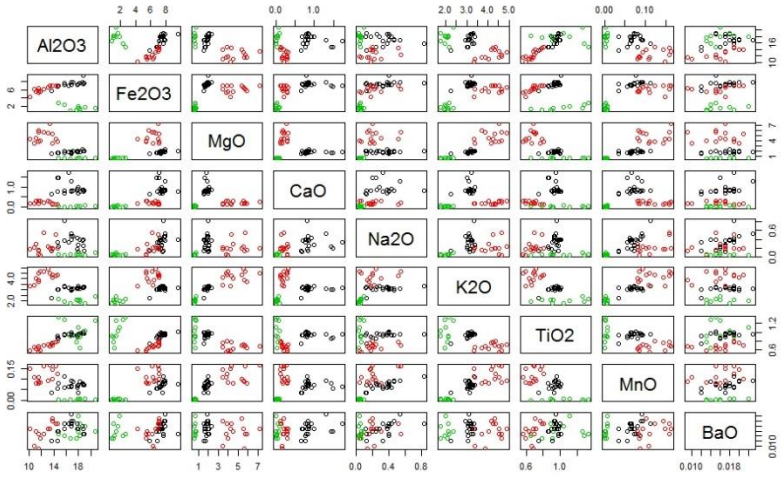
```
K-means clustering with 3 clusters of sizes 21, 14, 10
Cluster means:
    Al2O3    Fe2O3     MgO      CaO      Na2O     K2O      TiO2     MnO        BaO
1 6.259327 3.0877613 1.0575567 2.06711911 1.9395537 3.638752 5.214484 1.52010958 5.748909
2 4.600685 2.5803590 2.7425680 0.47170568 1.2663202 4.911143 3.797649 2.51367517 5.341695
3 6.566744 0.6700442 0.3673704 0.08585043 0.2861242 2.370047 5.672638 0.06837441 5.365649

Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1] 55.91185 49.30545 27.71633
 (between_SS / total_SS =  66.4 %)

Available components:

[1] "cluster"     "centers"     "totss"       "withinss"    "tot.withinss" "betweenss"   "size"        "iter"
[9] "ifault"
```

#We can get a graphical 'picture' of the clustering structure of the observations by using a scatter plot.

```
plot(pot.data2, col=kmcl.sd.pot$cluster)
```

#we add the centers of the clusters and use a different plotting character (pch)

```
points(kmcl.sd.pot$centers, col=1:k, pch=8)
```



Q2. (b) Hierarchical clustering

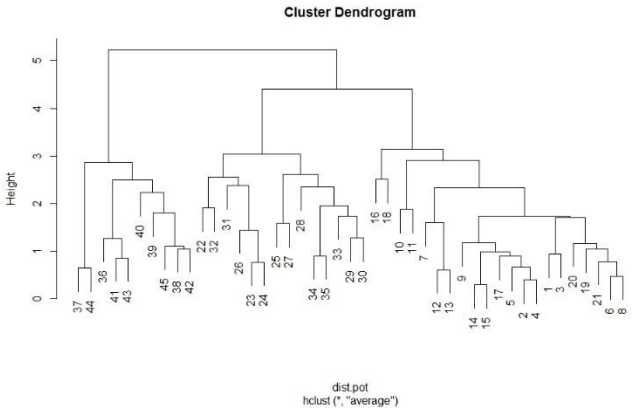
Cluster the 45 pots using hierarchical clustering and average linkage. From the dendrogram, how many clusters would you suggest are present in this data set? Cut the dendrogram at the desired number of clusters.

First, we need a dissimilarity matrix. we use a 'Euclidean dissimilarity' measure.

```
dist.pot <- dist(sd.pot, 'euclidean', diag = T); dist.pot
```

We cluster the matrix using the 'average linkage measure,' and plot it.

```
cl.pot <- hclust(dist.pot, 'average') #let's cluster the matrix using the 'average linkage measure.'
plot(cl.pot)
```



#the dendogram suggests 3 clusters are appropriate.

We cut the dendogram and check the size of clusters

```
hcl.pot <- cutree(cl.pot, k=3); hcl.pot
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

table(hcl.pot)
hcl.pot
 1  2  3
21 14 10
```

Q2. (c)

Compare the cluster solutions obtained in (a) and (b) using an appropriate measure and comment on the agreement between the two solutions.

```
hcl.pot <- cutree(cl.pot, k=3); hcl.pot
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

kmcl.sd.pot <- kmeans(sd.pot, centers = 3); kmcl.sd.pot
Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2

tab <- table(hcl.pot, kmcl.sd.pot$cluster); tab
 hcl.pot  1  2  3
    1    21  0  0
    2     0  0 14
    3     0 10  0
library(mclust)
adjustedRandIndex(hcl.pot, kmcl.sd.pot$cluster)
[1] 1
```

It seems that two clustering solutions produced exactly same result. Given that the order of clusters is pointless – we can swap any rows or columns –, we can assume that the two solutions show perfect agreements of size in each cluster – 21, 14, 10. The adjusted Rand Index function offered by 'mclust' package produced the value of 1 as well.

Q2. (d)

Comment on the relationship between both clustering solutions and the 'kiln' variable. Have you any concerns about the reproducibility of your clustering solutions?

As having seen from the two clustering results, it is obvious that we can cluster all pottery data into 3 different groups. In the original reference of the chemical composition of 45 pots, however, the data are broken down into 5 different locations (the kiln) at which the pottery was found. This might be able to cast the doubts on the reliability of the analysis. Assuming that the chemical composition patterns must be associated with the locations where they were found, it seems fair to expect to discover 5 different clusters that represent the kiln. This might raise the concerns of the reproducibility of our clustering solution because it is known that the

partition with SS within cluster variance is reproducible under repeated initialisations and this is generally the case for small numbers of clusters.

Q3. (a)

Calculate the quadratic discriminant function δk(x) for k =1,2 where x is the set of variables for a new member of the Pima Indian population. Does the new population member suffer from diabetes?

x = (npreg, glu, bp, skin, bmi, ped, age: 7, 187, 50, 33, 33.9, 0.826, 30)

```
pima.data = read.csv('C:/Users/Minkun/Dropbox/Minkun/Semester 2/40150_Multivariate A/assignment/a1/Pima.csv', header = T)
```

#P(g|x) = (group proportion) x (Normal density function with mean and variance). We need to maximize log{P(g|x)}. And it is assumed the groups are characterised by multivariate normal densities with mean vectors and covariance matrices.
#we fit the QDA model to the dataset.

```
library(MASS)
qda.pima <- qda(type~npreg+glu+bp+skin+bmi+ped+age, data=pima.data);qda.pima
Prior probabilities of groups:
        No       Yes
0.6660342 0.3339658

Group means:
        npreg      glu       bp     skin      bmi       ped      age
No  2.908832 110.1225 69.92308 27.23647 31.45185 0.4471368 29.16239
Yes 4.676136 142.9659 74.70455 32.98864 35.77330 0.6178011 36.37500
```

Now we want to construct QDA-function so that we can find the maximum of log{P(g|X)}. We already discovered the group proportions and group means which were offered by the model fitting above. However, we need covariance matrices to make the QDA function. The problem is that the assumption of an equal cov-matrix does not hold here, thus we introduce a new term which is -0.5*log|∑g| and which requires the estimation of more parameters.

First, we need to compute the group specific cov-matrices. We extract data corresponding to its group – No, Yes or K=1, 2

```
d.No = subset(pima.data, type=='No'); d.No
d.Yes = subset(pima.data, type=='Yes'); d.Yes
```

Then we find cov-matrices.

```
cov_No = cov(d.No[,-8]); cov_No
cov_Yes = cov(d.Yes[,-8]); cov_Yes
```

Finally we can compute the Quadratic Discriminant Function for the target observation x !
we build the new function that returns the QDF value.

```
qdf <- function(x, prior, mu, covar){
  x = matrix(as.numeric(x), ncol=1)
  log(prior) + (-0.5*t(x-mu)%*%solve(covar)%*%(x-mu)) - (0.5*log(det(covar)))
}
```

#We define the target observation x.

```
new.x = c(7, 187, 50, 33, 33.9, 0.826, 30)
```

#We store the resulting QDF value in a vector called dfs.

```
G = length(levels(pima.data$type))
dfs = rep(0, G)
```

#Since, in QDA, the shared cov-matrix is pointless, we calculate the QDF value on each occasion: k=1, 2, then choose the maximum value. And when k=1, i.e when the group is No, this returns QDF value.

```
for(g in 1:G) {
```

```
  dfs[g] <- qdf(new.x, qda.pima$prior[g], qda.pima$means[g,], cov_No)
} ;dfs
[1] -22.13318  -18.74559
```

When k=2, i.e when the group is Yes, this returns QDF value.

```
for(g in 1:G) {
  dfs[g] <- qdf(new.x, qda.pima$prior[g], qda.pima$means[g,], cov_Yes)
} ;dfs
[1] -19.68051  -17.85404
```

and we can find which group the target observation belongs to.

```
levels(pima.data$type)[dfs==max(dfs)]
[1] "Yes"
```

and its probability is…

```
round(exp(dfs)/sum(exp(dfs)), 4)
[1] 0.1387  0.8613
```

Therefore, we can conclude that the new population member would suffer from diabetes with a probability of 86%.

Q3. (b)

The five members of the Pima Indian population were initially set aside as a test set. Using the model fitted in part (a), calculate the misclassification rate of your quadratic discriminant analysis model.

#The five members of the Pima Indian population were initially set aside as a test set.

```
npreg <- c(2,9,10,5,1)
glu <- c(88,170,101,121,93)
bp <- c(58,74,76,72,70)
skin <- c(26,31,48,23,31)
bmi <- c(28.4,44.0,32.9,26.2,30.4)
ped <- c(0.766,0.403,0.171,0.245,0.315)
age <- c(22,43,63,30,23)
type <- c('No','Yes','No','No','No')
pima.test <- data.frame(npreg,glu,bp,skin,bmi,ped,age,type)
```

#we add the new data into the initial reference.

```
pima.data2 <- merge(pima.data, pima.test, all = T)
```

#We fit the model again concerning with cross validation

```
qda.pima <- qda(type~npreg+glu+bp+skin+bmi+ped+age, CV=T, data=pima.data2);qda.pima
```

#then compare the classes predicted by the model to the true known classes by examining a cross-tabulation...

```
table(pima.data2$type, qda.pima$class)
```

#before adding the new five observations

```
     No Yes
No  297  54
Yes  74 102 : misclassification rate: (54+74) / 527 = 0.243
```

#after adding the new five observations

```
     No Yes
No  301  54
Yes  72 105 : misclassification rate: (54+72) / 532 = 0.237
```

#It seems that an increase in the size of observations tends to diminish the misclassification rate.

STAT 30270_Statistical Data Mining
PROJECT.
ID: 16206265
Name: Minkun Kim

Q.2 Are there any interesting groupings of images in the data?

Q.3 Can image features be used to accurately predict which images are advertisements?

1. Intro

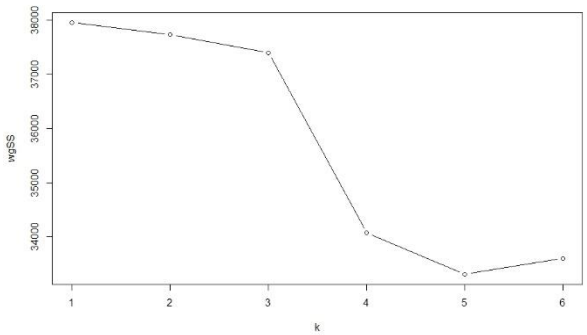
We have a categorical variable that consists of 2 classes.
>class.01: **Ad.**
>class.02: **Nonad.**
But we want to investigate further if there is any specific pattern in the dataset that prescribes the membership of these two classes. If we succeed in discovering the relationship or pattern that resides between predictors (images) and this categorical variable (Ad/NonAd), it becomes easier to predict if the new feed of data in the future would be classified as advertisements or not.

2. Method (K-means clustering + random Forest classification)

We can make two approaches to discovering the relationships. First, we do clustering analysis then see if the result corresponds to the two classes – Ad/NonAd. Implementing K-mean clustering algorithm enables us to compare a series of different results that are produced by parameterizing K value where 'K' indicates the number of grouping. In this process, we can reaffirm if the two classes – Ad/NonAd – are real clusters that fully reflect the features of the data (predictors), or there are other undiscovered clusters that deliver new meaning of the data. Second, using some classification algorithms is another way to unravel the hidden relations right off the bat. We will test various classifiers – classification trees, bagging, and random Forest – and choose the one that shows the best performance on our data set. These classifiers will allow us to identify the real connections between predictors (images) and the two classes (Ad/NonAd), articulating the precise criteria for prediction by assessing and comparing homogeneity of each possible group.

2A. Clustering Analysis
_how many "K" do we need? We want to make our clusters as tight as possible in order to improve the accuracy of our clustering output and this goal seems to be achievable by minimizing Sum of Squares. We plot the K values against SS, then see at which K value, SS is reduced dramatically.

```
WGSS = rep(0,6)
n = nrow(A)
WGSS[1] = (n-1)*sum(apply(A, 2, var))
for(k in 2:6){
  WGSS[k] <- sum(kmeans(A, centers = k)$withinss)
}
plot(1:6, WGSS, type = 'b', xlab = 'k', ylab = 'wgSS')
```



As seen from this plot, when K= 2, 3, and 4, SS appears to drop radically; therefore, we run K-means algorithms on each case.

K=4 (n=3278 obs)

```
Within cluster sum of squares by cluster:
[1]   920.950 24756.441  6662.275     0.000
 (between_SS / total_SS =  14.8 %)
table(kmcl.A$cluster)
   1    2    3    4
  80 2621  524   53
tab <- table(colvec.ad, kmcl.A$cluster); tab
              1    2   3   4
NonAd        64 2621 524  53
Ad           16    0   0   0
classAgreement(tab)
$diag
[1] 0.0195241
$kappa
[1] -0.008915968
$rand
[1] 0.6749611
$crand
[1] 0.03718947
```

```
Within cluster sum of squares by cluster:
[1]  7649.868 24756.441   920.950
 (between_SS / total_SS =  12.2 %)
table(kmcl.A$cluster)
   1    2    3
 577 2621   80
tab <- table(colvec.ad, kmcl.A$cluster); tab
              1    2   3
NonAd       577 2621  64
Ad            0    0  16
classAgreement(tab)
$diag
[1] 0.176022
$kappa
[1] -0.003707443
$rand
[1] 0.6801318
$crand
[1] 0.03805874
```

K=3 (n=3278 obs)

K=2 (n=3278 obs)

```
Within cluster sum of squares by cluster:
[1] 27202.857  7649.868
 (between_SS / total_SS =  8.2 %)
table(kmcl.A$cluster)
   1    2
2701  577
tab <- table(colvec.ad, kmcl.A$cluster); tab
              1    2
NonAd      2685  577
Ad           16    0
> classAgreement(tab)
$diag
[1] 0.819097
$kappa
[1] -0.009589745
$rand
[1] 0.7035554
$crand
[1] -0.007482111
```

>Result: This output tells us K=2 clustering is the most appropriate solution for this data set and which is underpinned by the Rand Index value (0.7). Interestingly, comparing this grouping pattern with the actual class division('ad.') in the data set, one can see that certain 16 observations are consistently grouped in the same cluster throughout all attempts, but it's not to say this consistent

grouping correspond to the class division of the original dataset in a precise manner because the size of Ad class and NonAd class is 458 and 2820 respectively. In this respect, we can say that there is a certain relationship between its predictors and categorical variable, but we cannot say that the class division in our data set does fully explain this relationship.

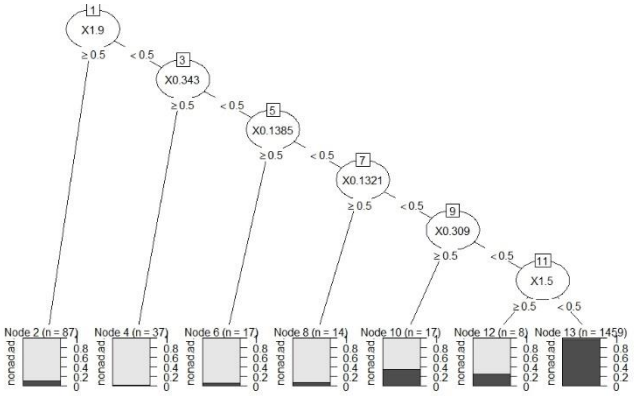
2B. Classification Analysis

_ We can investigate this relationship further with the help of classification analysis. We expect that this algorithm would drill down and narrow their results by creating a series of grouping trees. This meticulousness helps predict what feature of data determine its membership to certain cluster. In other words, we can determine if image features can be used to accurately predict which images are advertisements in our data. The algorithms - classification trees, bagging, random Forest – rise as possible tools for this analysis. Once fitting each model to the dataset, in order to select the best classifier to use, we compare the performances of them using diverse validation procedures. Division of training and test data is the key to carry out more reliable validation. However, given that the size of our data is so massive, we decided to conduct this test within our own data set instead of implementing other data extension methods such as bootstrapping.

2B_a. fitting models and assessing performance on the validation data.

>1.classification trees

```
tab <- table(ad.data$ad.[ind.valid], pred[ind.valid]); tab
          ad.  nonad.
ad.       99      28
nonad.    8      684
sum(diag(tab))/sum(tab)
[1] 0.956044
```



>2.bagging

```
tab <- pred$confusion; tab
          ad.  nonad.
ad.       97         6
nonad.  30       686
sum(diag(tab))/sum(tab)
[1] 0.956044
```

>3.random Forest

```
tab <- table(ad.data$ad.[ind.valid], pred[ind.valid]); tab
          ad.  nonad.
ad.       104        23
nonad.    2       690
sum(diag(tab))/sum(tab)
[1] 0.969475
```

As we expect, random Forest registers the best accuracy, thus accordingly, we try randomForest on the test data to get an accurate assessment of its performance.

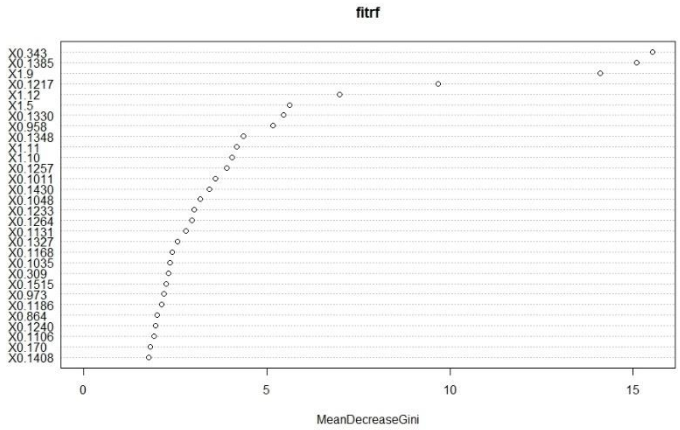
3B_b. fitting models and assessing performance on the test data.

>random Forest

```
tab <- table(ad.data$ad.[ind.test], pred[ind.test]); tab
          ad.  nonad.
ad.       105        26
nonad.    2       687
sum(diag(tab))/sum(tab)
[1] 0.9658537
```

>Result: random Forest proposes certain data division criteria which can be useful in determining if certain data with the image feature are advertisements or not.

```
varImpPlot(fitrf)
```



This plot gives us idea that which predictors are important and shows score per variable aggregated across the whole forest. In this way, random Forest predicts the membership of each data.