OLLSCOIL NA hÉIREANN, CORCAIGH
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Examination Session and Year	Winter 2022
Module Code	ST4060 ST6040
Module Name	Statistical Methods for Machine Learning I Machine Learning and Statistical Analytics I
Paper Number	Paper Number: 1
External Examiner	Mr. Andrew Maclaren
Head of School	Dr. Kevin Hayes
Internal Examiner(s)	Dr. Eric Wolsztynski
Instructions to Candidates	• Please answer all questions. • Provide all your answers in the Word document provided. • Paste your R code into the Word document at the end of each question. • Submit a pdf version of your final Word document for Canvas submission. Note: if you do not manage to answer a question item, provide the R code you would have used, or a comment on the answer you would expect for that question, as relevant.
Duration of Paper	3 hours

List of required R libraries:

```
glmnet
splines
```

List of (possibly) useful R functions:

```
apply()
approx()
as.numeric()
boxplot()
bs()
cbind()
coef()
colnames()
cor.test()
density()
fitted()
lines()
lm()
matrix()
mean()
median()
na.omit()
nls()
nrow()
numeric()
order()
par()
plot()
points()
predict()
quantile()
sample()
sd()
seq()
set.seed()
smooth.spline()
sqrt()
sum()
summary()
which()
```

Question 1 [15 marks]

Lucien is an analyst interested in the theoretic distribution of the sum of five Bernoulli random variables

$$X_i \overset{iid}{\sim} Bernoulli(p_i), \quad p_i = p \text{ (constant)}, \quad i = 1, \ldots, 5$$

To help him, implement a Monte Carlo simulation to evaluate the expected value and variance of this compound random variable

$$S = \sum_{i=1}^{5} X_i$$

Using $M = 1,000$ Monte Carlo resamples, each of size $N = 100$, compute Monte Carlo estimates of the mean and variance of S for all values of $p \in \{0.1, 0.2, \ldots, 0.9\}$ successively. Set the random seed to 4060 (`set.seed(4060)`) before running your analysis.

Note: one can generate a random sample of N realisations of a Bernoulli random variable with probability p in `R` as follows: `x = rbinom(N, size=1, prob=p)`.

(a) Quote your Monte Carlo estimate of the mean of S, for each value of p.

(b) Quote your Monte Carlo estimate of the variance of S, for each value of p.

(c) In a two-frame figure, with the two frames side by side, plot the boxplots of the Monte Carlo distributions of estimates of the means and variances of S, respectively, with respect to p.

(d) Based on the above results, briefly discuss whether S could be thought to follow a *Poisson binomial* distribution with $E(S) = \sum_{i=1}^{5} p_i = 5p$ and $Var(S) = \sum_{i=1}^{5}(1-p_i)p_i = 5p(1-p)$.

Solution:
R code:

```
# Monte Carlo illustration of Le Cam's Theorem
N = 100
p = .1
set.seed(4060)
M = 1000
means = vars = NULL
Ps = seq(.1,.9,by=.1)
for(p in Ps)
m = v = numeric(M)
for(i in 1:M)
X = matrix(rbinom(5*N,1,p),nrow=N,ncol=5)
S = apply(X,1,sum)
m[i] = mean(S)
v[i] = var(S)

means = cbind(means, m)
vars = cbind(vars, v)

boxplot(means, names=Ps, col='cyan')
abline(h=5*Ps, col=8)
```

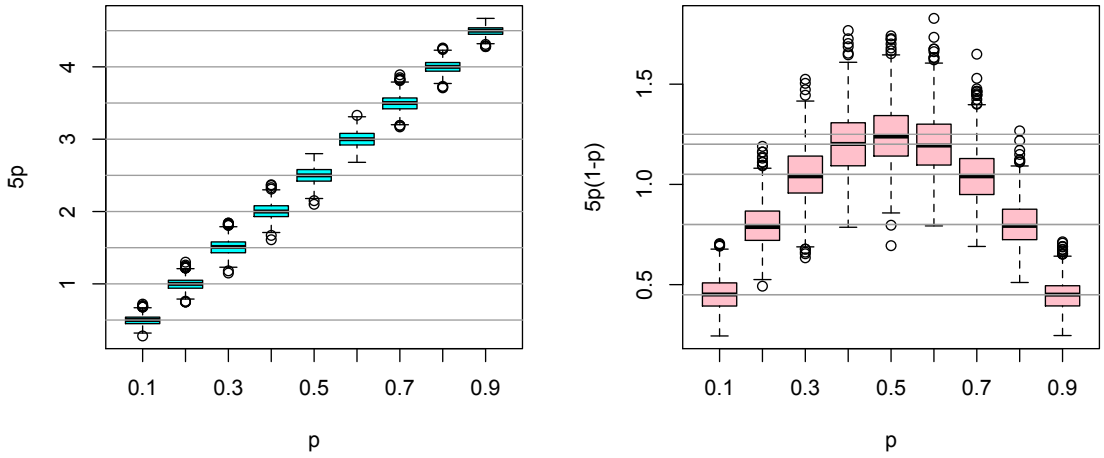
(a) Mean MC estimates of the mean should be something like the following:

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$E(S)$	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
$\hat{\mu}$	0.5010	0.9986	1.5047	2.0051	2.4989	3.0011	3.4952	4.0002	4.4966

(b) Mean MC estimates of the variance should be something like the following:

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$Var(S)$	0.45	0.80	1.05	1.20	1.25	1.20	1.05	0.80	0.45
$\hat{\sigma}^2$	0.4509	0.7946	1.0500	1.2047	1.2485	1.2027	1.0480	0.8020	0.4507

(c) Boxplots of MC distributions:



(d) There is clear alignment between estimates $\{\hat{\mu}\}_{i=1}^{M}$ and $E(S)$ on one hand, and between estimates $\{\hat{\sigma}^2\}_{i=1}^{M}$ and $Var(S)$ on the other hand, in other words the analysis suggests that for M large,

$$\frac{1}{M}\sum_{i=1}^{M}\hat{\mu} \approx E(\mu) = E(S)$$

and

$$\frac{1}{M}\sum_{i=1}^{M}\hat{\sigma}^2 \approx E(\sigma^2) = Var(S)$$

Question 2 [30 marks]

Load the `commute.csv` dataset into your `R` session as follows:

```
dat = read.csv('commute.csv', stringsAsFactors=TRUE)
```

This dataset contains 500 observations of workers in a major US city on their age (in years), gender (M or F), and distance (in miles) and time (in minutes) of their commute to work each day. Each of the 500 respondents worked somewhere other than home.

Use $B = 1,000$ bootstrap resamples and set the random seed to 4060 (`set.seed(4060)`) before running each of your bootstrap analyses.

(a) Quote the bootstrap estimates of the mean commute time and of its associated standard error.

(b) Bootstrap the p-value of the correlation test (`cor.test()`) between the commuters' age and their commute time.

(c) What proportion of the bootstrap tests in (b) were significant at the 5% level?

(d) Compute and quote the bootstrap bias of the p-value from (b).

(e) Compute and quote a 95% confidence interval for the p-value estimation in (b), using the bootstrap mean and standard error and the normal approximation to formulate the confidence interval.

(f) Compute and quote an *ordinary* nonparametric bootstrap 95% confidence interval for the p-value estimation in (b).

(g) Briefly discuss whether the normal approximation is a reasonable approach for calculation of the confidence interval for the p-value of the correlation test.

Solution:
R code:

```
dat = read.csv('data/commute.csv',stringsAsFactors=TRUE)

set.seed(4060)
B = 1000
n = nrow(dat)
samples = matrix(sample(dat$Time, size=B*n, replace=TRUE), B, n)
bmeans = apply(samples, 1, mean)
time.se = sd(bmeans)
# BS expected value
mean(bmeans)
# BS std error
time.se

set.seed(4060)
pb = numeric(B)
for(b in 1:B)
```

```
ib = sample(1:n,n,replace=TRUE)
datb = dat[ib,]
pb[b] = cor.test(datb$Age,datb$Time)$p.value

mean(pb)
mean(pb<.05)
# statistic of original sample:
p0 = cor.test(dat$Age,dat$Time)$p.value
# BS bias:
mean(pb)-p0
# 95% Normal CI:
mean(pb)+c(-1,1)*1.96*sd(pb)
# BS-adjusted CI:
2*p0 - quantile(pb,c(.975,.025))
# Normal CI not appropriate given the distribution:
hist(pb)
```

(a) Bootstrap mean commute time: **29.11 minutes**.
Bootstrap standard error of the sample mean: **0.95 minutes**.

(b) Bootstrap p-value: **0.41**.

(c) Rate of bootstrap p-values belowe 0.05: **13%**.

(d) Bootstrap bias for the p-value: $\bar{p^*} - p^0 = \mathbf{-0.139}$

(e) Approximate 95% normal CI:

$$\bar{p^*} \pm 1.96 SE(p^*) = (\mathbf{-0.176}, \mathbf{0.999})$$

(alternative using actual bootstrap correction also accepted.)

(f) Nonparametric bootstrap 95% CI (q^*_α denoting quantile bootstrap p-value):

$$(2p^0 - q^*_{0.975}, 2p^0 - q^*_{0.025}) = (\mathbf{0.132}, \mathbf{1.098})$$

(g) Looking at the histogram of bootstrap p-values will show an almost uniform-looking distribution, in any case far from the normal pattern, which is sufficient to rule out the normal approximation baed on bootstrap statistics used in (e).

Question 3 [30 marks]

Load the `blood_pressure.csv` dataset into your `R` session as follows:

```
dat = read.csv(file='blood_pressure.csv')
x = dat$BMI
y = dat$Systolic
```

In this question we only use body mass index `x` (BMI, in kg/m^2) and systolic blood pressure `y` (in mmHg) from this dataset of 75 clinical observations of patients.

Set the random seed to 4060 (`set.seed(4060)`) before running your analysis.

(a) Fit a univariate GLM model to the whole dataset, to describe systolic blood pressure as a function of BMI. Compute and quote the root mean square error (RMSE) from this fit.

(b) Fit a kernel density estimator (KDE) to the *scaled* residuals from this fit. Quote the bandwidth that was automatically set for this KDE.

(c) Compute and quote the RMSE between the KDE obtained in (b) and a standard normal probability density function (pdf) evaluated at the same points.

(d) Compute and quote the RMSE between the KDE obtained in (b) and a Student pdf with two degrees of freedom evaluated at the same points.

(e) Based on (c) and (d), briefly discuss which model better describes the residuals, and what this tells you about your GLM fit obtained in (a).

(f) Using `library(splines)`, fit a B-spline to the data (`x`,`y`), using quantiles (0.15, 0.60, 0.70, 0.80) of `x` as knots. Compute and quote the RMSE from this fit.

(g) Briefly discuss whether you would use the GLM fit from (a) or the B-spline from (f) to analyse the data (`x`,`y`), using relevant output to support your opinion.

Solution:
R code:

```
library(glmnet)
dat = read.csv(file='data/blood_pressure.csv')
x = dat$BMI
y = dat$Systolic

glmo = glm(Systolic~BMI, data=dat)
sqrt(mean(residuals(glmo)^2))

kde = density(scale(glmo$residuals))
kde$bw

f.ref = dnorm(kde$x,0,1)
sqrt(mean((kde$y-f.ref)^2))

f.t = dt(kde$x,2)
```

```
sqrt(mean((kde$y-f.t)^2))

library(splines) # contains function bs()
KN = quantile(dat$BMI, c(0.15, 0.60, 0.70, 0.80))
BM = bs(dat$BMI, knots=KN)
B.spline = lm(y~BM)
sqrt(mean(residuals(B.spline)^2))

plot(dat$BMI,dat$Systolic,pch=20)
lines(dat$BMI,fitted(glmo))
os = order(dat$BMI)
lines(dat$BMI[os],fitted(B.spline)[os],col=3,lwd=2)
```

(a) GLM fit RMSE: **13.24**

(b) KDE bandwidth $h =$ **0.3280512**

(c) Distance from KDE to normal: **0.0180**

(d) Distance from KDE to Student: **0.0380**

(e) The KDE is clearly closer to the standard normal, indicating the model residuals are reasonably in line with the model assumption of normally distributed noise, which suggests a reasonable fit and model.

(f) B-spline fit RMSE: **12.62**

(g) The B-spline seems to yield a lower error (with a 4.7% decrease in RMSE over GLM), however plotting the fit will show an unreasonably complex structure; the B-spline overfits the data and the GLM is likely preferable here.

Question 4 [25 marks]

Load the `blood_pressure.csv` dataset into your `R` session as follows:

```
dat = read.csv(file='blood_pressure.csv')
x = model.matrix(Systolic~.+0, data=dat)
y = dat$Systolic
```

This dataset contains 75 observations of patients on five variables including their age (in years), waist circumference (in cm), systolic blood pressure (in mmHg), cholesterol level (in mg/dL) and body mass index (BMI, in kg/m^2).

Set the random seed to 4060 (`set.seed(4060)`) before running each of the below analyses.

(a) Implement 10-fold cross-validation of a lasso model of systolic blood pressure `y` as a function of the other four covariates `x`, using a fixed value $\lambda = 10$ for the regularisation parameter. Compute and store the training and test root means square errors (RMSEs) from the cross-validated fits. Quote the mean estimates of training and test prediction errors.

(b) Carry out the same analysis as in (a) but using a cross-validated value for the regularisation parameter. Compute and store the training and test RMSEs from the corresponding cross-validated fits. Quote the means of these training and test RMSEs.

(c) Carry out the same analysis as in (a) but using a multivariate GLM instead of the lasso. Compute and store the training and test RMSEs from the corresponding cross-validated fits. Quote the means of these training and test RMSEs.

(d) Plot the boxplots of all four sets of errors from (a) and (b) in a single plot frame. Briefly comment on these results, explaining any difference you may find.

(e) Based on the above results, briefly comment on each model performance and indicate whether one model is preferable over the other, and why.

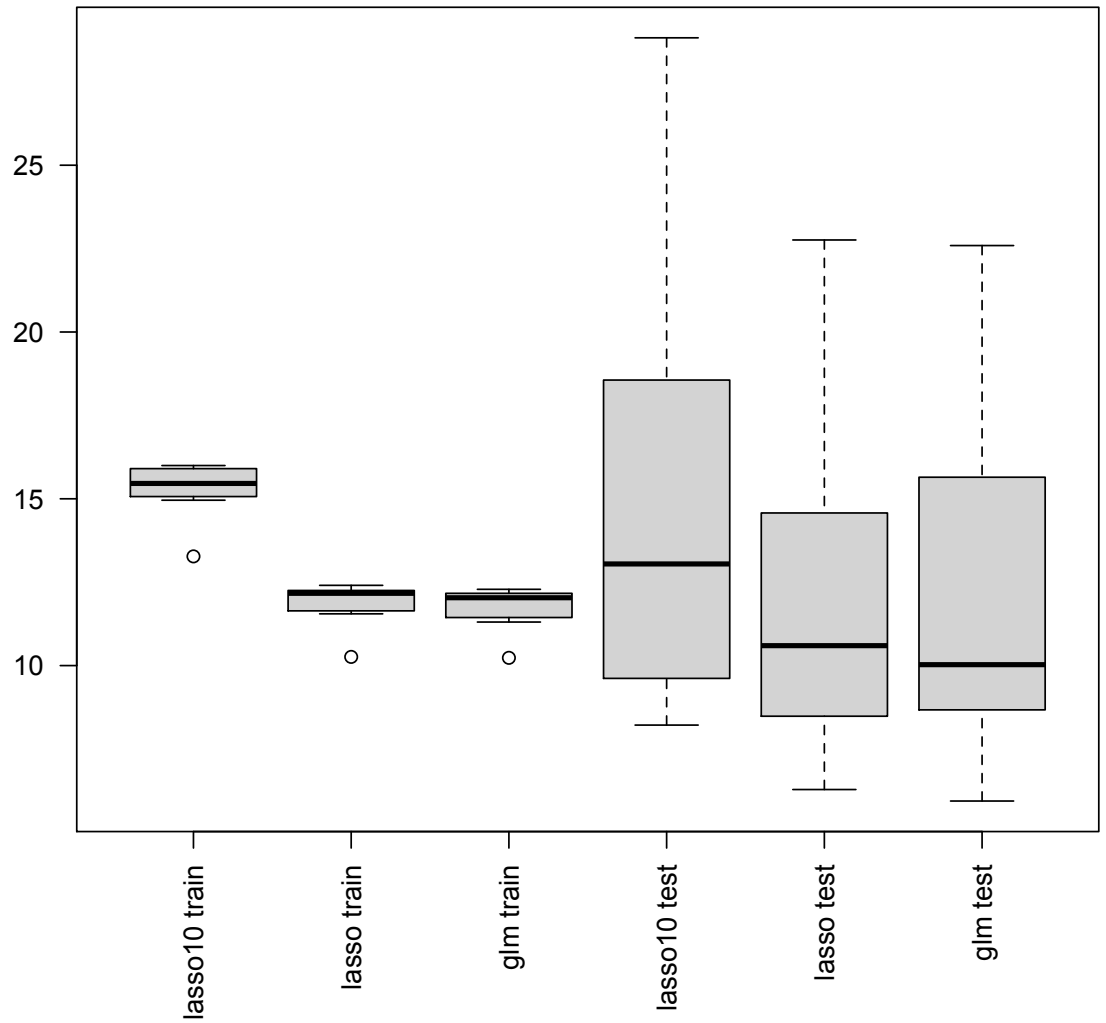
Solution:

(a) Naive lasso train RMSE: **15.3**, test RMSE: **14.6**.

(b) Tuned lasso train RMSE: **11.9**, test RMSE: **11.9**.

(c) GLM train RMSE: **11.8**, test RMSE: **11.9**.

(d) Boxplots:



(e) The naive lasso clearly underfits and is therefore not reliable for prediction. The tuned lasso and the traditional GLM perform comparably. There seems to be either little need for regularisation, or need to use an alternative strategy such as ridge regression to tackle colinearity in the data, but the GLM so far seems satisfactory. It would also make for a simpler model to use in a routine setting with non-experts.

R code:

```
library(glmnet)
dat = read.csv(file='data/blood_pressure.csv')
x = model.matrix(Systolic~.+0,data=dat)
y = dat$Systolic
n = nrow(x)
K = 10
folds = cut(1:n,K,labels=FALSE)
fit.rmse.lasso = p.rmse.lasso = numeric(K)
```

```
fit.rmse.lasso.cv = p.rmse.lasso.cv = numeric(K)
fit.rmse.glm = p.rmse.glm = numeric(K)
lam.cv = numeric(K)
set.seed(4060)
for(k in 1:K)
i.train = which(folds!=k)
i.test = which(folds==k)
x.train = x[i.train,]
y.train = y[i.train]
x.test = x[i.test,]
y.test = y[i.test]
#
lam = 10
lasso = glmnet(x.train,y.train,lambda=lam)
y.hat = predict(lasso,x.train)[,1]
fit.rmse.lasso[k] = sqrt(mean((y.hat-y.train)^2))
y.p = predict(lasso,x.test)[,1]
p.rmse.lasso[k] = sqrt(mean((y.p-y.test)^2))
#
lam.cv[k] = cv.glmnet(x.train,y.train)$lambda.min
lasso.cv = glmnet(x.train,y.train,lambda=lam.cv[k])
y.hat.cv = predict(lasso.cv,x.train)[,1]
fit.rmse.lasso.cv[k] = sqrt(mean((y.hat.cv-y.train)^2))
y.p.cv = predict(lasso.cv,x.test)[,1]
p.rmse.lasso.cv[k] = sqrt(mean((y.p.cv-y.test)^2))
#
glmk = glm(Systolic~.,dat[i.train,],family='gaussian')
glm.hat = predict(glmk,dat[i.train,])
fit.rmse.glm[k] = sqrt(mean((glm.hat-y.train)^2))
y.p.glm = predict(glmk,dat[i.test,])
p.rmse.glm[k] = sqrt(mean((y.p.glm-y.test)^2))

nms = c("lasso10","lasso","glm")
boxplot(fit.rmse.lasso,fit.rmse.lasso.cv,fit.rmse.glm,
p.rmse.lasso,p.rmse.lasso.cv,p.rmse.glm,
names=c(paste(nms,"train"),paste(nms,"test")),
las=2)
```