ST4060 – Statistical Methods for Machine Learning I
ST6015 – Computer Analytical Techniques for Actuarial Applications
ST6040 – Machine Learning and Statistical Analytics I

Continuous Assessment 2 2021-22

Name: Aditya Vikram Dash
Student ID: 120224162
Programme (FMAS4, RAS4, MScDSA, etc.): MScDSA

Please provide your answers to each question below, and provide any relevant R code with your answers to each question. Submit a pdf version of this document as your unique submission document on Canvas.

Remember to save your work regularly.

Your answer to Question 1:

(a)

Question	Answer	Justification
(i) Regression or classification?	Classification	Straight up classification problem.
(ii) Can the scientist use MSE?	No	MSE is for regression, not classification.
(iii) Supervised or unsupervised?	Supervised	We have all the data readily available for the output we want to predict which can be used to train our model – so, supervised.

(b)

Statement	True or False?	Justification
(i)	False	ErrTest cannot be smaller than trained error.
(ii)	True	Regularisation is used to reduce overfitting.
(iii)	True	Ridge regression has the least overfitting of data.
(iv)	False	ErrTest-ErrTrain has a better Ridge value than LASSO.

(c)

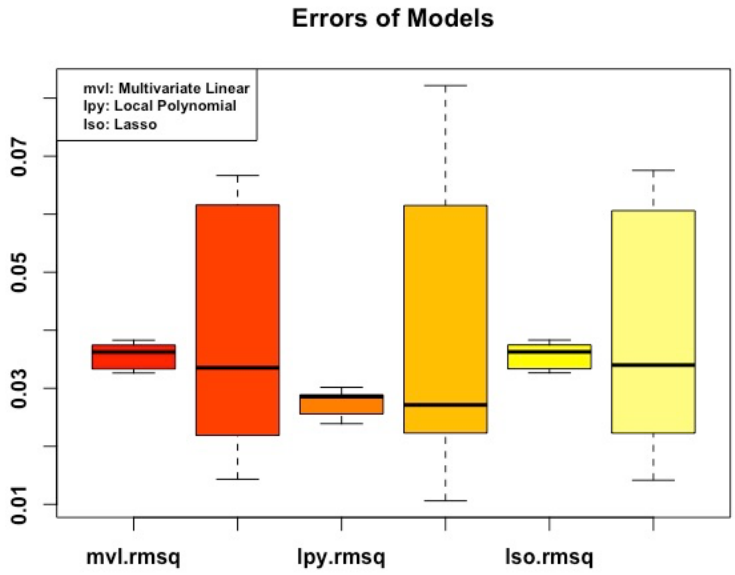
Statement	True or False?	Justification
(i)	True	Error 2 boxplot of errors is much lesser and compact than other two. Error 1 and 3 are for Test Data.
(ii)	False	Test Error can't be smaller than Training Error.
(iii)	False	Test Error can't be smaller than Training Error.
(iv)	True	As Error spread is lower for Error 1 compared to Error 3.

Your answers to Question 2:

(a)

Model	Linear regression	Polynomial regr.	LASSO
CV training error	0.03575279	0.02754139	0.03577952
CV test error	0.03809638	0.03865921	0.0381408

(b) Boxplots:



(c) On comparison, we observe that Lasso has less error as it reduces coefficients, Linear Regression has more since it causes overfitting and its coefficients are not minimized; Local Polynomial has highest due to high degree.

Relevant R code for Question 2:

```r
for(k in 1:K){
  # split the data
  train = which(folds!=k)

  # linear regression
  mvl = lm(y~., data=x, subset=train)
  # local polynomial
  lpy = loess(y~., data=x, subset=train, control=loess.control(surface='direct'))
  # lasso
  lso.cv = cv.glmnet(xm[train,], y[train], grouped=F)
  lso = glmnet(xm[train,], y[train], alpha=1, lambda=lso.cv$lambda.min)

  # fitted values
  # linear regression
  mvl.yh = mvl$fitted.values
  # local polynomial
  lpy.yh = lpy$fitted
  # lasso
  lso.yh = predict(lso, newx=xm[train,])

  # rmsq
  # linear regression
  mvl.rmsq[k] = mean((mvl.yh-y[train])^2)
  # local polynomial
  lpy.rmsq[k] = mean((lpy.yh-y[train])^2,na.rm=T)
  # lasso
  lso.rmsq[k] = mean((lso.yh-y[train])^2,na.rm=T)

  # bootstrap points
  test = which(folds==k)

  # predictions
  # mvl
  lrp = predict(mvl, newdata=x[test,])
  # lpy
  lpp = predict(lpy, newdata=x[test,], na.rm=T)
  # lasso
  lap = predict(lso, newx=xm[test,], na.rm=T)

  # prediction errors
  # linear regression
  mvl.pred.err[k] = mean((lrp-y[test])^2)
  lpy.pred.err[k] = mean((lpp-y[test])^2, na.rm=T)
  lso.pred.err[k] = mean((lap-y[test])^2, na.rm=T)
}
```

```r
# linear regression errors
mean(mvl.rmsq)
mean(mvl.pred.err)

# local polynomial errors
mean(lpy.rmsq)
mean(lpy.pred.err)

# lasso
mean(lso.rmsq)
mean(lso.pred.err)

# boxplot
boxplot(mvl.rmsq,mvl.pred.err,lpy.rmsq,lpy.pred.err,lso.rmsq,lso.pred.err,
        col=heat.colors(6), main="Errors of Models",
        names=c('mvl.rmsq','mvl.pred.err','lpy.rmsq','lpy.pred.err','lso.rmsq','lso.pred.err'))
legend("topleft", legend=c("mvl: Multivariate Linear", "lpy: Local Polynomial","lso: Lasso"),
       cex=0.7)
```

Your answers to Question 3:

(a)

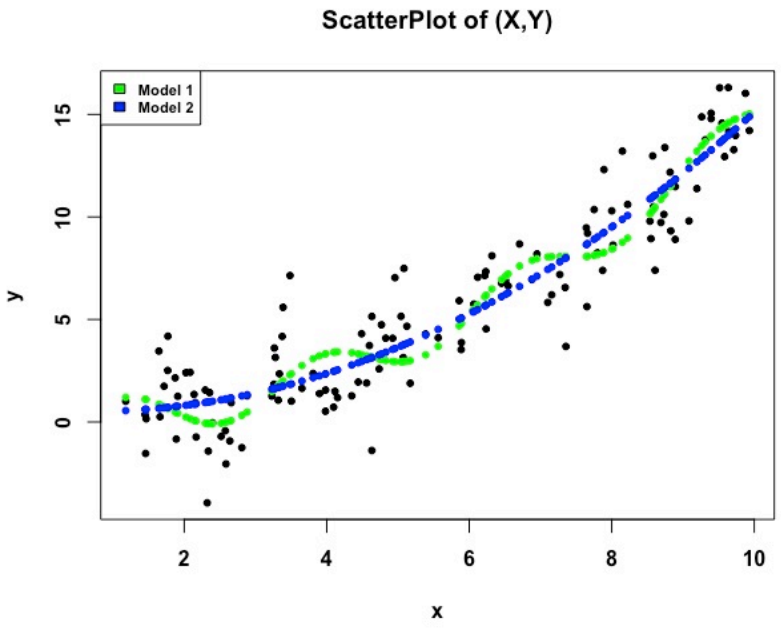
Optimal value for a:	0.15
Optimal value for b:	0.2

(b)

Coefficient	a	b	c
Fit for model 1	0.146885	-1.070910	2.264759
Fit for model 2	0.16781	-0.22710	0.58810

RMSE for model 1:	1.843127
RMSE for model 2:	1.799604

(c) Scatterplot:



(d)

Coefficient	a	b	c
Bootstrap coefficient estimates:	0.1642713	-0.1889164	0.5173158

Relevant R code for Question 3:

```r
L=20
a.grid=seq(0.01,1,by=.01)
a.grid
length(a.grid)
b.grid=seq(0.1,2,by=0.1)
b.grid
length(b.grid)
grid_matrix= matrix(NA,nrow=length(a.grid),ncol=length(b.grid))
for (i in 1:length(a.grid)){
  for(j in 1:length(b.grid)){
    grid_matrix[i,j]=mean((y-model(c(a.grid[i],b.grid[j]),x ))^2)
  }
}
grid_matrix
persp(a.grid,b.grid,grid_matrix,col="green",theta=45)
contour(a.grid,b.grid,grid_matrix)
min(grid_matrix)
optimal=which.min(grid_matrix)
optimal = arrayInd(optimal, .dim=dim(grid_matrix))
optimal

a.optimal = a.grid[optimal[1]]
a.optimal
b.optimal = b.grid[optimal[2]]
b.optimal
```

```r
nlm_1 = nls(y~a*x^2+sin(b+c*x),
         start=list(a=0.05,b=0.4,c=2))
nlm_2 = nls(y~a*x^2+b*x+c,
          start=list(a=0.05,b=0.4,c=2))

# estimates
summary(nlm_1)
summary(nlm_2)

# rmse
nlm_1.rmsq = sqrt(mean( (y - (0.146885*x^2+sin(-1.070910 +2.264759 *x)))^2  ))
nlm_1.rmsq = sqrt(mean( (y - (0.16781*x^2-0.22710 *x+0.58810))^2 ) )
```

```r
B = 100
ia = ic = ib = numeric(B)
xb = yb = numeric(N)
set.seed(4060)
for (b in 1:B){
  i = sample(1:N, size=N, replace=T)
  xb = dat$x[i]
  yb = dat$y[i]
  nlm_2 = nls(yb~a*(xb^2)+(b*xb)+c,
              start=list(a=0.05,b=0.4,c=2))
  ia[b] = coef(nlm_2)[1]
  ib[b] = coef(nlm_2)[2]
  ic[b] = coef(nlm_2)[3]
}

# bootstrap estimates
mean(ia)
mean(ib)
mean(ic)
```