Question 1 - Linear regression model

a) The scatterplots and correlation matrix (see **Figure** 1) show that there is a negative correlation between response variable(`mpg`) and `cylinders`, `displacement`, `horsepower` and `weight`. There is a positive correlation between response variable and `acceleration`, `year` and `origin`. In the boxplots of class conditional distributions, none seems to have nearly identical distributions for all classes, indicating that both variables `cylinders` and `origin` may be associated with the response `mpg`

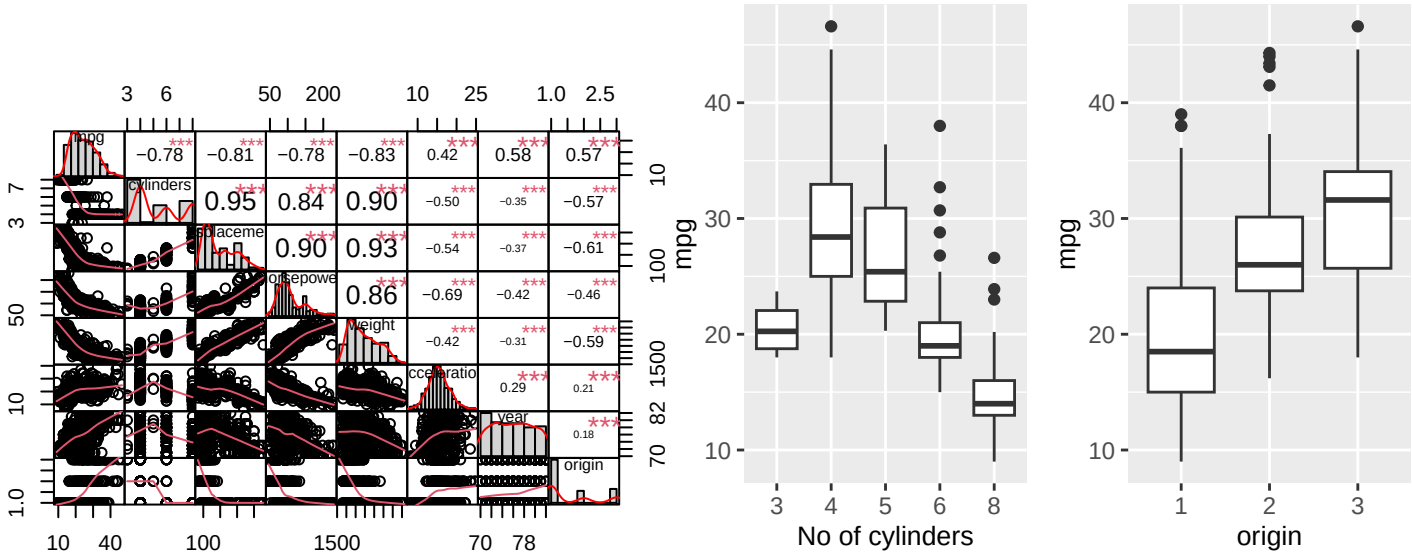

Figure 1: *Exploratory analysis. Left:Scatterplot matrix for Auto data. Middle:Boxplot of mpg vs cylinders, Right:Boxplot of mpg vs origin*

b) **Table** 1 represent the regression coefficients of fitted linear regression model for predicting `mpg` using all the remaining variables.Note that all neumarical predictor variables are standardized. LOOCV estimate of test MSE for this model is 11.37113.

	Coefficients from lm	Coefficients from deep learning model
(Intercept)	23.4459	11.1850200
cylinders	-0.8416	-1.0172445
displacement	2.082	-1.0522574
horsepower	-0.6525	-1.1918335
weight	-5.4991	-2.1132698
acceleration	0.2223	-0.3746326
year	2.7656	2.4453557
origin	1.1488	1.0254807
Test error rate	11.37113	311.6921

Table 1: Summary for the linear regression models

c) Please refer to Quesion 1 part (c) in the `R` code section for the code. The calculated LOOCV test error rate is 11.37113

d) Please refer to Quesion 1 part (d) in the `R` code section for the code. The calculated LOOCV test error rate is 311.6921. **Table** 1 represent the regression coefficients obtained from deep learning model for predicting `mpg` using all the remaining variables.

e) MSE values are identical for part b) and part c). But MSE and regression coefficients are quite different when compare models from part b) and part d). It could be due to small number of epochs used in part d). When I check the loss function at last run model is still improving (training loss is getting smaller and smaller). This implies that if we increase the number of epochs deep learning model will give some what similar results to the model obtain in part b)

Question 2 - Logistic regression model

a) The response variable `default` is a binary variable with levels No and Yes indicating whether the customer defaulted on their debt. About 96.67% of the outcomes are 'No' and 33.3% are 'Yes'. In the boxplots of class conditional distributions (see **Figure** 2), `balance` seems to have different distributions for two response classes, indicating that variable `balance` may be associated with the response `default`. There seem not to have association between variables `default` and `students` according to the stacked barplot.

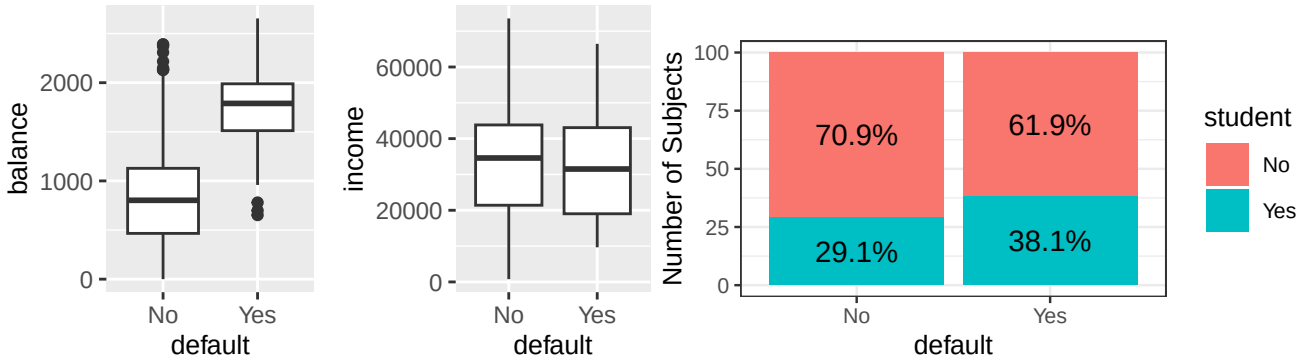

Figure 2: *Exploratory analysis. Left:Boxplot of balance vs default. Middle:Boxplot of income vs default, Right:Stacked barplot of default and student*

b) **Table** 2 represent the coefficients of fitted logistic regression model for predicting `default` using all the remaining variables. Note that all neumarical predictor variables are standardized. 5 fold CV estimate of test error for this model is 0.027.

	Coefficients from glm	Coefficients from deep learning model
(Intercept)	-5.97524	-4.4681
studentYes	-0.64678	-1.4490322
balance	2.77483	2.0443287
income	0.04046	-0.3389541
Test error rate	0.02680	0.028

Table 2: Summary for the logistic regression models

c) Please refer to Quesion 2 part (c) in the `R` code section for the code. The calculated 5 fold estimate of test error rate is 0.027.

d) Please refer to Quesion 2 part (d) in the `R` code section for the code. The calculated 5 fold test error rate is 0.028. **Table** 2 represent the logistic regression coefficients obtained from deep learning model.

e) Test errors are identical for part b) and part c). Moreover test errors are close for models in part b) and part d). Increase the number of epochs of deep learning model may give identical results to the model obtain in part b).

Question 3 - Multinomial regression model

a) In the boxplots of class conditional distributions (see **Figure** 3), none seems to have nearly identical distributions for three response classes, indicating that all variables may be associated with the response `Species`

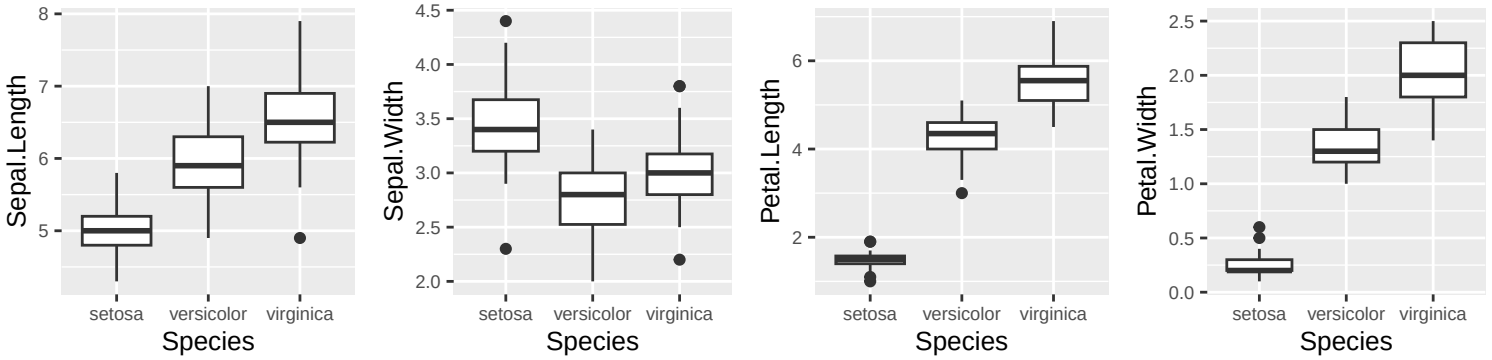

Figure 3: *Exploratory analysis. Boxplots of balance vs other variables*

b) **Table** 3 represent the coefficients of fitted multinomial regression model for predicting `Species` using all the remaining variables.Note that all neumarical predictor variables are standardized. 5 fold CV estimate of test error for this model is 0.02667.

Coefficients	Coefficients from multinom		Coefficients from deep learning model		
	versicolor	**virginica**	**setosa**	**versicolor**	**virginica**
(Intercept)	9.023056	-11.021851	-0.10877903	0.04390925	0.06489126
Sepal.Length	-3.345346	-5.380110	-0.5303301	0.4563077	0.5329779
Sepal.Width	-3.039437	-5.945487	-0.7120199	0.3313866	-0.2143498
Petal.Length	13.69517	30.29748	-0.5843883	0.9941038	1.0456575
Petal.Width	8.090794	21.980253	-0.0954898	-0.6549985	-0.4770274
Test error rate	0.02667		0.393		

Table 3: Summary for the multinomial regression models

c) Please refer to Quesion 3 part (c) in the `R` code section for the code. The calculated 5 fold cv test error rate is 0.393. **Table** 3 represent the coefficients obtained from deep learning model.

d) Test errors and coefficients are quite different when compare models from part b) and part d). It could be due to small number of epochs used in part d). When I check the loss function at last run model is still improving (training loss is getting smaller and smaller). This implies that if we increase the number of epochs deep learning model will give some what similar results to the model obtain in part b).

Section 2: R code

```
## ----setup, include=FALSE----------------------------------
knitr::opts_chunk$set(echo = TRUE)
options(xtable.comment = FALSE)
knitr::opts_chunk$set(dev = 'pdf')


## ----echo=FALSE--------------------------------------------
library(knitr)
opts_chunk$set(comment="",warning = FALSE, message=FALSE,tidy.opts=list(keep.blank.line=TRUE, width.cutoff=120),


## ----include=FALSE-----------------------------------------
## calling required libraries
library(ISLR)
library(PerformanceAnalytics)
library(caret)
library(boot)
library(dplyr)
library(ggplot2)
library(boot)
library(nnet)
library(keras)

#################################################
#                  Question 1
#################################################

######## 1a) ############

## ----q1a, echo=FALSE, fig.cap="\\textit{Exploratory analysis. Left:Scatterplot matrix for Auto data. Middle:Bo
chart.Correlation(Auto_dat)
plot1<-ggplot(Auto_dat, aes(x=as.factor(cylinders), y=mpg)) +
    geom_boxplot() + labs(x = "No of cylinders") + theme(legend.position = "none",axis.title.x = element_text(si
plot2<-ggplot(Auto_dat, aes(x=as.factor(origin), y=mpg)) +
    geom_boxplot() + labs(x = "origin") + theme(legend.position = "none",axis.title.x = element_text(size=10), a

require(gridExtra)
grid.arrange(plot1, plot2, ncol=2)


## ----include=FALSE-----------------------------------------
Auto_dat <- Auto[,-9]
mean <- apply(Auto_dat[,-1], 2, mean)
std <- apply(Auto_dat[,-1], 2, sd)
scaledX<- scale(Auto_dat[,-1], center = mean, scale = std)
Auto_dat <- data.frame(cbind(mpg=Auto[,1], scaledX))


######## 1b) ############

## ----include=FALSE-----------------------------------------
#fitting model
lm.fit <- lm(mpg ~ . ,data = Auto_dat)
summary(lm.fit)

#calculating LOOCV estimate of test MSE using cv.glm
lm.cv.err <- cv.glm(Auto_dat, glm(mpg ~ ., data = Auto_dat))
lm.cv.err$delta
```

```r
#calculating LOOCV estimate of test MSE using caret package
lm.fit1 <- train(
    form = mpg ~ . ,
    data = Auto_dat,
    trControl = trainControl(method="LOOCV",
        seeds = set.seed(1)),
    method = "lm")
lm.fit1$results$RMSE^2


######## 1c) ############

## ----include=FALSE-----------------------------------------
set.seed(1)

loocv<-function(i)
{
  test<-Auto_dat[i,]
  training<-Auto_dat[-i,]
  model<-lm(mpg ~ . , data = training)
  pred <- predict(model,test)
  err = (test[, "mpg"] - pred )^2
  return(err)
}

# If we are performing loocv more than one time we must rearrange data
#re_Auto<-Auto_dat[sample(nrow(Auto_dat)),]
error <- sapply(1:nrow(Auto_dat), FUN = loocv)
cat("MSE:",mean(error),"\n")



######## 1d) ############

## ----include=FALSE-----------------------------------------
Auto_dat <- Auto[,-9]
mean <- apply(Auto_dat[,-1], 2, mean)
std <- apply(Auto_dat[,-1], 2, sd)
train_data <- data.frame(scale(Auto_dat[,-1], center = mean, scale = std))

train_targets <- Auto_dat[,1]
train_data <- as.matrix(train_data)

# Define a simple model
model <- keras_model_sequential()
model %>%
  layer_dense(units = 1, input_shape = dim(train_data)[2], activation = 'linear')

mse <- c()

#Perform loocv
for(i in 1:nrow(train_data)){
  val_data <- train_data[i,]
  val_data <- matrix(val_data,nrow=1)
  val_targets <- train_targets[i]

  partial_train_data <- train_data[-i,]
  partial_train_targets <- train_targets[-i]
```

```r
  # Compile the model
  model %>% compile(
    loss = 'mse',
    optimizer = "rmsprop"
  )

  #Train model
  history <- model %>% fit(
    partial_train_data, partial_train_targets,
    epochs = 5,
    batch_size = 64,
    verbose = 0
  )

  #make predictions
  ypred <- predict(model, val_data)
  mse[i] <- (val_targets - ypred)^2
}

weights <- model %>% get_weights()
mean(mse)


##############################################
#                 Question 2
##############################################


######## 2a) ############

## ----q2a, echo=FALSE, fig.cap="\\textit{Exploratory analysis. Left:Boxplot of balance vs default. Middle:Boxpl
plot1<-ggplot(Default, aes(x=default, y=balance)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element
plot2<-ggplot(Default, aes(x=default, y=income)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element

require(gridExtra)
grid.arrange(plot1, plot2, ncol=2)

Default_df = Default

Default_df %>%
  count(default, student) %>%
  group_by(default) %>%
  mutate(pct= prop.table(n) * 100) %>%
  ggplot() + aes(default, pct, fill=student) +
  geom_bar(stat="identity") +
  ylab("Number of Subjects") +
  geom_text(aes(label=paste0(sprintf("%1.1f", pct),"%")),
            position=position_stack(vjust=0.5)) +
  ggtitle(" ") +
  theme_bw(base_size = 9.5)


## ----include=FALSE----------------------------------------
mean <- apply(Default[,3:4], 2, mean)
std <- apply(Default[,3:4], 2, sd)
scaledX<- scale(Default[,3:4], center = mean, scale = std)
Default_dat <- data.frame(cbind(Default[,1:2], scaledX))
```

```
######## 2b) ############

## ----include=FALSE-----------------------------------------
glm.fit <- glm(default ~ ., data = Default_dat, family = "binomial")

#calculating 5 fold CV estimate of test MSE using cv.glm
# Note: By default cost (loss) is MSE, which needs to be
# changed to misclassification rate for qualitative response
cost <- function(r, pi = 0){mean(abs(r - pi) > 0.5)}

set.seed(1)
glm.cv.err <- cv.glm(Default_dat, glm.fit, cost = cost, K = 5)
glm.cv.err$delta

#calculating 5 fold CV estimate of test MSE using caret package
glm.fit1 <- train(
    form = default ~ .,
    data = Default_dat,
    trControl = trainControl(method = "cv", number = 5,
        seeds = set.seed(1)),
    method = "glm",
    family = "binomial")
glm.fit1


######## 2c) ############

## ----include=FALSE-----------------------------------------
indices <- sample(1:nrow(Default_dat))
err.rate <-c()
#Perform 10 fold cross validation
folds <- cut(indices,breaks=5,labels=FALSE)
for(i in 1:5){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  test <- Default_dat[testIndexes, ]
  training <- Default_dat[-testIndexes, ]
  model<-glm(default ~ . ,family = binomial, data = training)
  pred.prob <- predict(model,test, type = "response")
  pred <- ifelse(pred.prob >= 0.5, "Yes", "No")
  err.rate[i] = 1 - mean(pred == test[, "default"])
}
mean(err.rate)


######## 2d) ############

## ----include=FALSE-----------------------------------------
def_dat<-Default_dat
def_dat$default <- as.numeric(def_dat$default=="Yes")
def_dat$student <- as.numeric(def_dat$student=="Yes")
def_dat <- as.matrix(def_dat)
train_data <- def_dat[,2:4]
train_targets <- def_dat[,1]

build_model <- function(){
  model <- keras_model_sequential() %>%
    layer_dense(units = 1, input_shape = c(3), activation = "sigmoid")
```

```
  ### compile the model
  model %>% compile(
    optimizer = "rmsprop",
    loss = "binary_crossentropy", # loss function to minimize
    metrics = c("accuracy") # monitor classification accuracy
  )
}

k <- 5
indices <- sample(1:nrow(def_dat))
folds <- cut(indices, breaks = k, labels = FALSE)
num_epochs <- 100
all_scores <- c()

#Kfold cross validation
for (i in 1:k){
  cat("Processing fold #", i, "\n")

  val_indices <- which(folds == i, arr.ind = TRUE)
  val_data <- train_data[val_indices,]
  val_targets <- train_targets[val_indices]

  partial_train_data <- train_data[-val_indices,]
  partial_train_targets <- train_targets[-val_indices]

  model <- build_model() # use precompiled model function

  model %>% fit(partial_train_data, partial_train_targets,
                epochs = num_epochs,
                batch_size = 128,
                verbose = 0) # trains the model in silent mode (verbose = 0)

  # evaluate model on the validation data
  results <- model %>% evaluate(val_data, val_targets, verbose = 0)
  all_scores <- c(all_scores, results[2])
}

weights <- model %>% get_weights()
all_scores
mean(all_scores)


###############################################
#                 Question 3
###############################################


######## 3a) ############

## ----q3a, echo=FALSE, fig.cap="\\textit{Exploratory analysis. Boxplots of balance vs other variables}",fig.hei
plot1<-ggplot(iris, aes(x=Species, y=Sepal.Length)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element
plot2<-ggplot(iris, aes(x=Species, y=Sepal.Width)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element
plot3<-ggplot(iris, aes(x=Species, y=Petal.Length)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element
plot4<-ggplot(iris, aes(x=Species, y=Petal.Width)) +
    geom_boxplot() + theme(legend.position = "none",axis.title.x = element_text(size=10), axis.title.y = element
require(gridExtra)
```

```r
grid.arrange(plot1, plot2, plot3, plot4,ncol=4)


## ----include=FALSE-----------------------------------------
mean <- apply(iris[,1:4], 2, mean)
std <- apply(iris[,1:4], 2, sd)
scaledX<- scale(iris[,1:4], center = mean, scale = std)
iris_dat <- data.frame(Species = iris[,5], scaledX)
iris_dat <- within(iris_dat, {
  Species <- factor(Species, labels  = 0:2, levels = c("setosa","versicolor","virginica"))
})


######## 3b) ############

## ----include=FALSE-----------------------------------------
iris.fit <- train(
    form = Species ~ .,
    data = iris_dat,
    trControl = trainControl(method = "cv", number = 5,
        seeds = set.seed(1)),
    method = "multinom",
    trace = FALSE)
iris.fit

iris_nw = iris_dat
iris_nw$Species <- relevel(iris_nw$Species, ref = "0")
mult.fit <- multinom(Species ~ . , data = iris_nw)
summary(mult.fit)


######## 3c) ############

## ----include=FALSE-----------------------------------------
mean <- apply(iris[,1:4], 2, mean)
std <- apply(iris[,1:4], 2, sd)
scaledX<- scale(iris[,1:4], center = mean, scale = std)
iris_dat <- data.frame(scaledX,Species = iris[,5])
iris_dat <- within(iris_dat, {
  Species <- factor(Species, labels  = 0:2, levels = c("setosa","versicolor","virginica"))
})

train_targets <- iris_dat[,5]
train_data <- iris_dat[,1:4]
train_data <- as.data.frame(train_data)
train_data <- as.matrix(train_data)

train_labels <- to_categorical(train_targets)

build_model <- function(){
  # specify the model
  model <- keras_model_sequential() %>%
    layer_dense(units = 3, input_shape = dim(train_data)[2], activation = "softmax")

  model %>% compile(
    optimizer = "rmsprop",
    loss = "categorical_crossentropy",  # loss function to minimize
    metrics = c("accuracy") # monitor classification accuracy
  )
```

```
}

#5 fold cv
k <- 5
indices <- sample(1:nrow(iris_dat))
folds <- cut(indices, breaks = k, labels = FALSE)
num_epochs <- 50 # number of epochs
all_scores <- c()
for (i in 1:k){
  cat("Processing fold #", i, "\n")

  val_indices <- which(folds == i, arr.ind = TRUE)
  val_data <- train_data[val_indices,]
  val_targets <- train_labels[val_indices,]

  partial_train_data <- train_data[-val_indices,]
  partial_train_targets <- train_labels[-val_indices,]

  model <- build_model() # use precompiled model function

  model %>% fit(partial_train_data, partial_train_targets,
                epochs = num_epochs,
                batch_size = 32,
                verbose = 0) # trains the model in silent mode (verbose = 0)

  # evaluate model on the validation data
  results <- model %>% evaluate(val_data, val_targets)
  all_scores <- c(all_scores, results[2])
}
weights <- model %>% get_weights()
1- mean(all_scores)
```