Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2022-2023)
EC433 – Artificial Intelligence and Machine Learning

Lab ISE Programming Tutorial in R

Name: Abhinav Pallayil **Roll No. – 2019120045** **Date: 7/11/2022**

Objective: Program the following algorithm in RStudio:

1. Implement the PCA for feature extraction
2. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points while selecting appropriate data set for your experiment and draw graphs.
3. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.
4. Lab 7 Tutorial: Deep Learning for text/Linear regression/ Linear classification

Theory:

Principal component analysis (PCA) is a popular technique for analyzing large datasets containing a high number of dimensions/features per observation, increasing the interpretability of data while preserving the maximum amount of information, and enabling the visualization of multidimensional data. PCA is a dimensionality reduction technique that has four main parts: feature covariance, eigen decomposition, principal component transformation, and choosing components in terms of explained variance.

Locally weighted linear regression is a non-parametric algorithm, that is, the model does not learn a fixed set of parameters as is done in ordinary linear regression. Rather parameters θ are computed individually for each query point x. While computing θ, a higher “preference” is given to the points in the training set lying in the vicinity of x than the points lying far away from x.

In machine learning, backpropagation is a widely used algorithm for training feedforward neural networks. Generalizations of backpropagation exist for other artificial neural networks (ANNs), and for functions generally. These classes of algorithms are all referred to generically as "backpropagation". In fitting a neural network, backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input–output example, and does so efficiently, unlike a naive direct computation of the gradient with respect to each weight individually. This efficiency makes it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss; gradient descent, or variants such as stochastic gradient descent, are commonly used.

Text classification is one of the popular tasks in NLP that allows a program to classify free-text documents based on pre-defined classes. The classes can be based on topic, genre, or sentiment. Today’s emergence of large digital documents makes the text classification task more crucial, especially for companies to maximize their workflow or even profits. Recently, the progress of NLP research on text classification has arrived at the state-of-the-art (SOTA). It has achieved terrific results, showing Deep Learning methods as the cutting-edge technology to perform such tasks.

Code:

```
#https://medium.com/analytics-vidhya/feature-extraction-and-brand-perceptual-map-using-pca-in-r-31477002fdf2

sink("./Output.txt",append=T)

#read data
rating <- read.csv("http://goo.gl/IQl8nc")

#scale data
rating.sc <- rating
rating.sc[,1:9] <- scale(rating[,1:9])

#compute mean
brand.mean <- aggregate(rating.sc[,1:9], list(rating.sc[,10]), mean)

rownames(brand.mean) <- paste("",letters,sep="")[1:10]
(brand.mean <- brand.mean[,-1] )
rating.pc <- prcomp(brand.mean, scale=TRUE)
summary(rating.pc)

jpeg("rating_pc.jpeg",quality = 100)
plot(rating.pc, type="l")
dev.off()

#positioning
jpeg("Brand_Positioning_1.jpeg",quality = 100)
biplot(rating.pc, main="Brand Positioning", cex=c(1,0.7))
dev.off()

brand.dist <- dist(brand.mean)
(brand.mds <- cmdscale(brand.dist))

#differentiation
jpeg("Brand_Positioning_2.jpeg",quality = 100)
plot(brand.mds, type="n", main="Brand Differentiation")
rownames <- paste("",letters,sep="")[1:10]
text(brand.mds, rownames, cex=1)
dev.off()

sink()
```

Code for Question 1

```
#https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf

sink("./Output.txt",append=T)

library(faraway)
data(eco)

#jpeg("Proportion US born vs. Mean Annual Income 1.jpeg",quality = 100)
```

```
plot(income~usborn, data=eco, xlab="Proportion US born",ylab="Mean Annual Income")
#dev.off()

g <- lm(income~usborn, eco)
summary(g)

#("Proportion US born vs. Mean Annual Income 2.jpeg",quality = 100)
plot(income~usborn, data=eco, xlab="Proportion US born",ylab="Mean Annual
Income",xlim=c(0,1),ylim=c(15000,70000),xaxs="i")
#dev.off()

abline(g$coef)
data(chicago)
chicago
ch <- data.frame(chicago[,1:4],involact=chicago[,6],income=chicago[,7]/1000)
ch
summary(ch)
par(mfrow=c(2,3))
for(i in 1:6) hist(ch[,i],main=names(ch)[i])
for(i in 1:6) boxplot(ch[,i],main=names(ch)[i])
pairs(ch)
summary(lm(involact~race,data=ch))
g <- lm(involact~race + fire + theft + age + log(income), data = ch)
plot(g$fit,g$res,xlab="Fitted",ylab="Residuals",main="Residual-Fitted plot")
abline(h=0)
qqnorm(g$res)
gi <- lm.influence(g)
for(i in 1:5) qqnorml(gi$coef[,i+1],main=names(ch)[-5][i])
qqnorml(rstudent(g),main="Jacknife Residuals")
qt(0.05/(2*47),47-6-1)
halfnorm(cooks.distance(g),main="Cook-Statistics")
ch[c(6,24),]
g <- lm(involact~race + fire + theft + age + log(income),ch,subset=(1:47)[-c(6,24)])
summary(g)
g2 <- lm(involact~race + poly(fire,2) + poly(theft,2) + poly(age,2) + poly(log(income),2),
ch, subset=(1:47)[-c(6,24)])
anova(g,g2)
y <- ch$inv[cooks.distance(g) < 0.2]
x <- cbind(ch[,1:4],linc=log(ch[,6]))
x <- x[cooks.distance(g) < 0.2,]
library(leaps)
a <- leaps(x,y)
Cpplot(a)
g <- lm(involact~race + fire + theft + age, ch, subset=(1:47)[-c(6,24)])
summary(g)
galt <- lm(involact~race+fire+log(income),ch,subset=(1:47)[-c(6,24)])
summary(galt)
galt <- lm(involact~ race+fire,ch,subset=(1:47)[-c(6,24)])
summary(galt)
g <- lm(involact~race + fire + theft + age, data=ch)
```

```
summary(g)
data(chiczip)
g <- lm(involact~race + fire + theft +age, subset=(chiczip == "s"), ch)
summary(g)
g <- lm(involact~ race + fire + theft +age, subset=(chiczip == "n"), ch)
summary(g)

sink()
```

Code for Question 2

```
##https://rpubs.com/jrnorton11/FeedForwardBackPropagation

sink("./Output.txt",append=T)

beta <- 0.45
alpha <- 0.9
input <- N0 <- matrix(c(1,1))
t <- 1
w0 <- matrix(c(.4,-.1,.1,-.1), nrow=2)
w1 <- matrix(c(0.06, -0.4), nrow=1)

sprintf("Input is:")
print(input)
sprintf("Target is: %f",t)
sprintf("w0 is:")
print(w0)
sprintf("w1 is:")
print(w1)

sigma <- function(t) 1/(1+exp(-t))

N1 <- sigma(w0 %*% input)
sprintf("N1 is:")
print(N1)

N2 <- sigma(w1 %*% N1)

N2.0.error <- N2 * (1-N2) * (1-N2)
sprintf("N2.0.error is: %f",N2.0.error)

w1.Rate = (beta * N2.0.error[1,1]) * t(N1)
sprintf("w1.Rate is:")
print(w1.Rate)

sprintf("w1 is:")
print(w1)
w1.new <- w1 + w1.Rate + alpha*(t-1)
sprintf("w1.new is:")
print(w1.new)

N1.0.error <- N2.0.error %*% w1.new
```

```
w0.Rate = t(beta * N1.0.error) * (N0)
sprintf("w0.Rate is:")
print(w0.Rate)

w0.Rate <- matrix(c(w0.Rate[1,1], w0.Rate[2,1], w0.Rate[1,1], w0.Rate[2,1]), nrow=2)
sprintf("w0.Rate is:")
print(w0.Rate)

w0.new <- w0 + w0.Rate + alpha*(t-1)
sprintf("w0.new is:")
print(w0.new)

w0 <- w0.new
N1 <- sigma(w0 %*% input)

w1 <- w1.new
N2 <- sigma(w1 %*% N1)

sprintf("N2 is:")
print(N2)

N2.1.error <- N2 * (1-N2) * (1-N2)
sprintf("N2.1.error is: %f",N2.1.error)

sink()
```

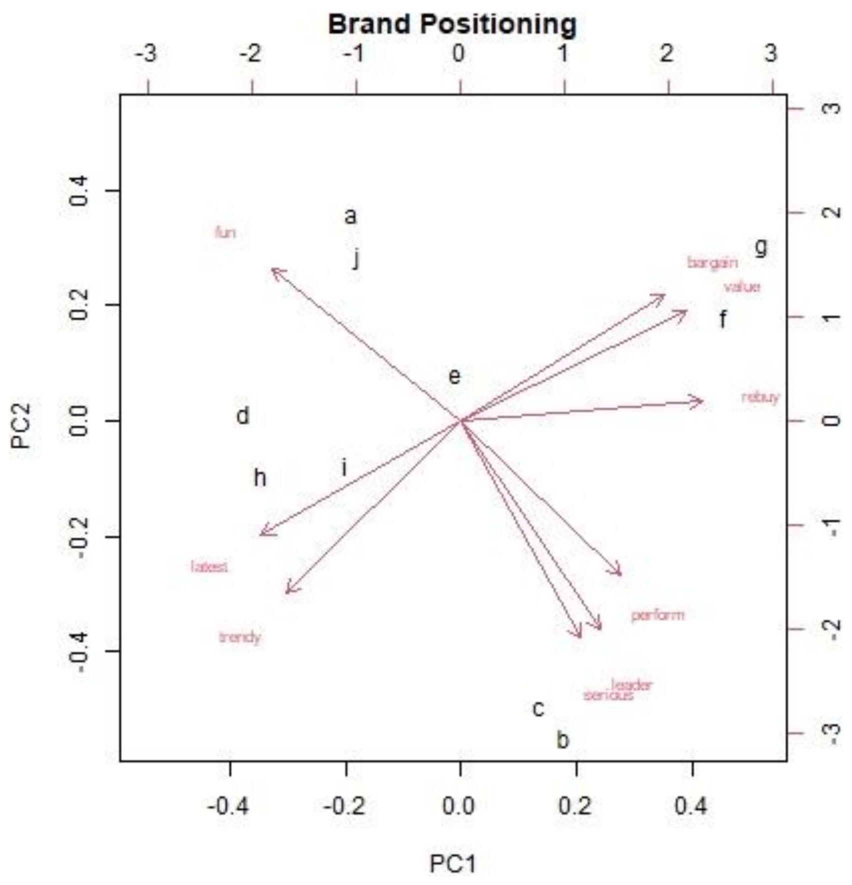
Code for Question 3

```
#https://blogs.rstudio.com/ai/posts/2017-12-07-text-classification-with-keras/

sink("./Output.txt",append=T)

library(keras)
imdb <- dataset_imdb(num_words = 10000)
train_data <- imdb$train$x
train_labels <- imdb$train$y
test_data <- imdb$test$x
test_labels <- imdb$test$y
str(train_data[[1]])
train_labels[[1]]
max(sapply(train_data, max))

# Named list mapping words to an integer index.
word_index <- dataset_imdb_word_index()
reverse_word_index <- names(word_index)
names(reverse_word_index) <- word_index

# Decodes the review. Note that the indices are offset by 3 because 0, 1, and
# 2 are reserved indices for "padding," "start of sequence," and "unknown."
decoded_review <- sapply(train_data[[1]], function(index) {
  word <- if (index >= 3) reverse_word_index[[as.character(index - 3)]]
  if (!is.null(word)) word else "?"
})
cat(decoded_review)

vectorize_sequences <- function(sequences, dimension = 10000) {
  # Creates an all-zero matrix of shape (length(sequences), dimension)
  results <- matrix(0, nrow = length(sequences), ncol = dimension)
  for (i in 1:length(sequences))
    # Sets specific indices of results[i] to 1s
    results[i, sequences[[i]]] <- 1
  results
}

x_train <- vectorize_sequences(train_data)
x_test <- vectorize_sequences(test_data)

str(x_train[1,])

y_train <- as.numeric(train_labels)
y_test <- as.numeric(test_labels)

library(keras)

model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

model %>% compile(
  optimizer = optimizer_rmsprop(lr=0.001),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

model %>% compile(
  optimizer = optimizer_rmsprop(lr = 0.001),
  loss = loss_binary_crossentropy,
  metrics = metric_binary_accuracy
)

val_indices <- 1:10000

x_val <- x_train[val_indices,]
partial_x_train <- x_train[-val_indices,]
```

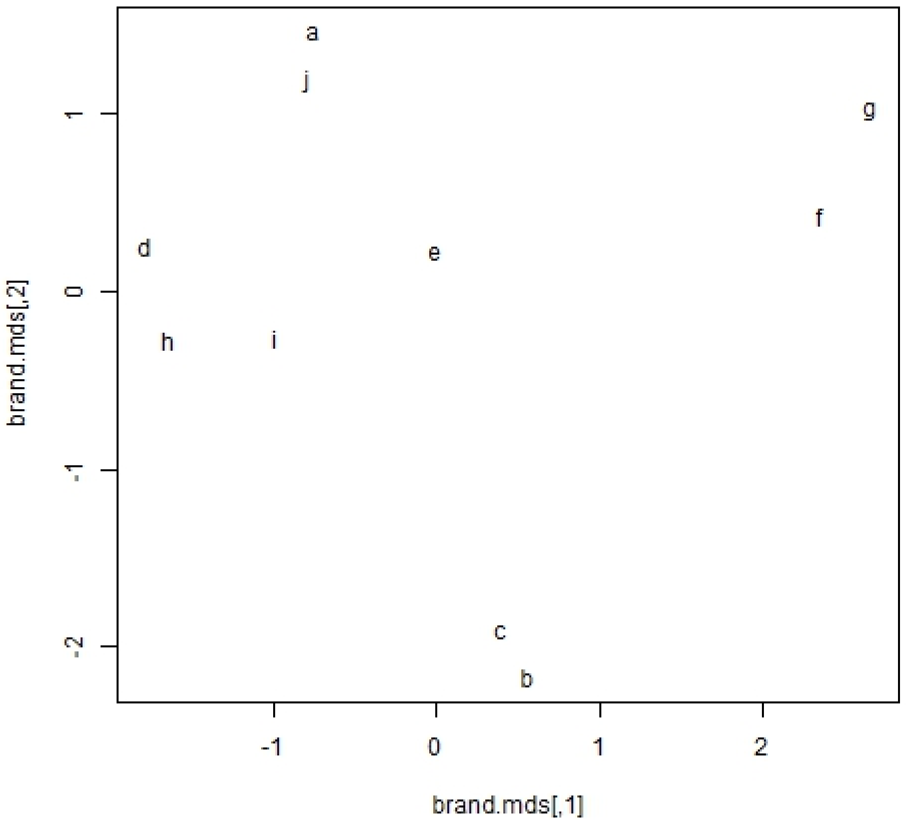
```
y_val <- y_train[val_indices]
partial_y_train <- y_train[-val_indices]

model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

history <- model %>% fit(
  partial_x_train,
  partial_y_train,
  epochs = 20,
  batch_size = 512,
  validation_data = list(x_val, y_val)
)

model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

model %>% fit(x_train, y_train, epochs = 4, batch_size = 512)
results <- model %>% evaluate(x_test, y_test)

results

model %>% predict(x_test[1:10,])

jpeg("History.jpeg",quality = 100)
plot(history)
dev.off()

sink()
```

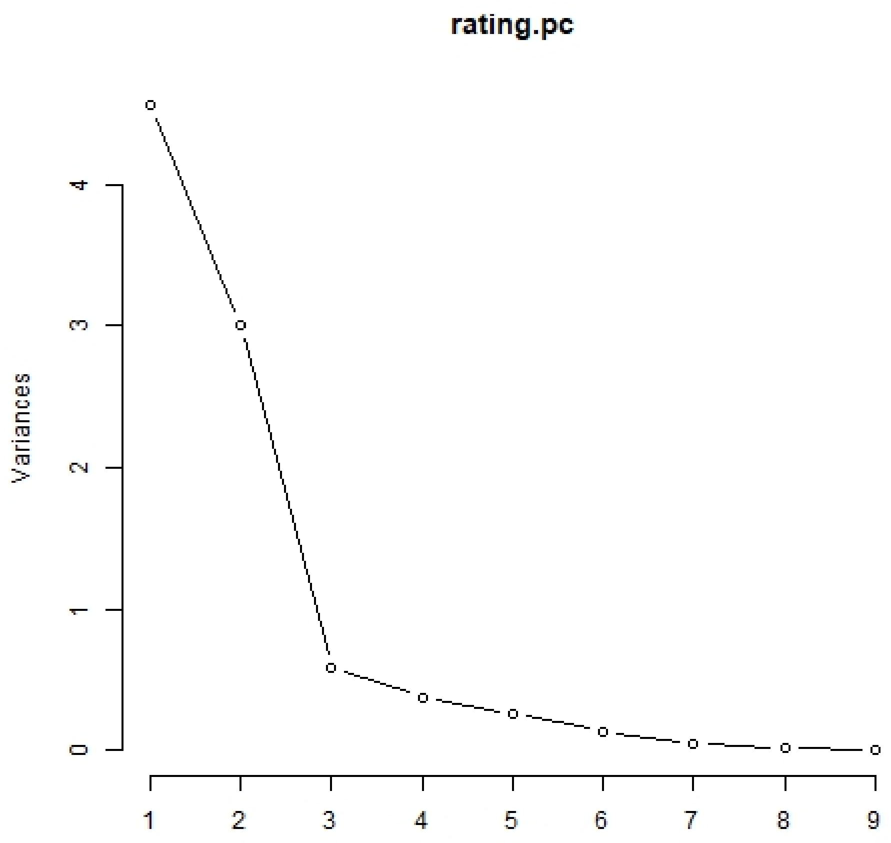
Code for Question 4

Output: (.TEX file added in the folder)
Question 1:

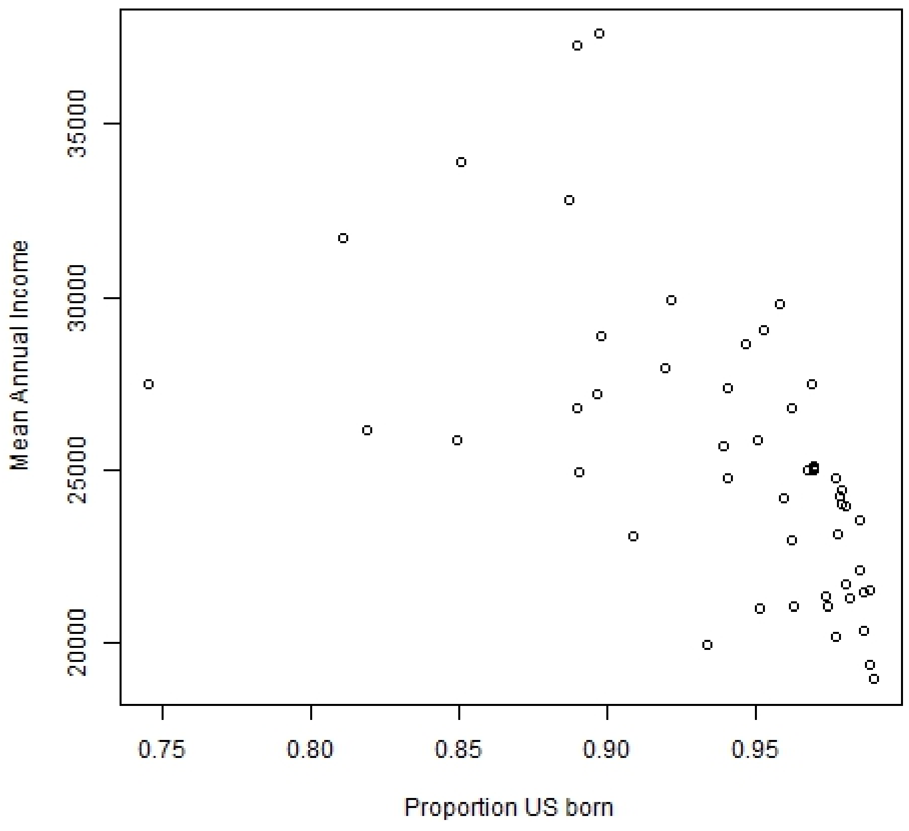


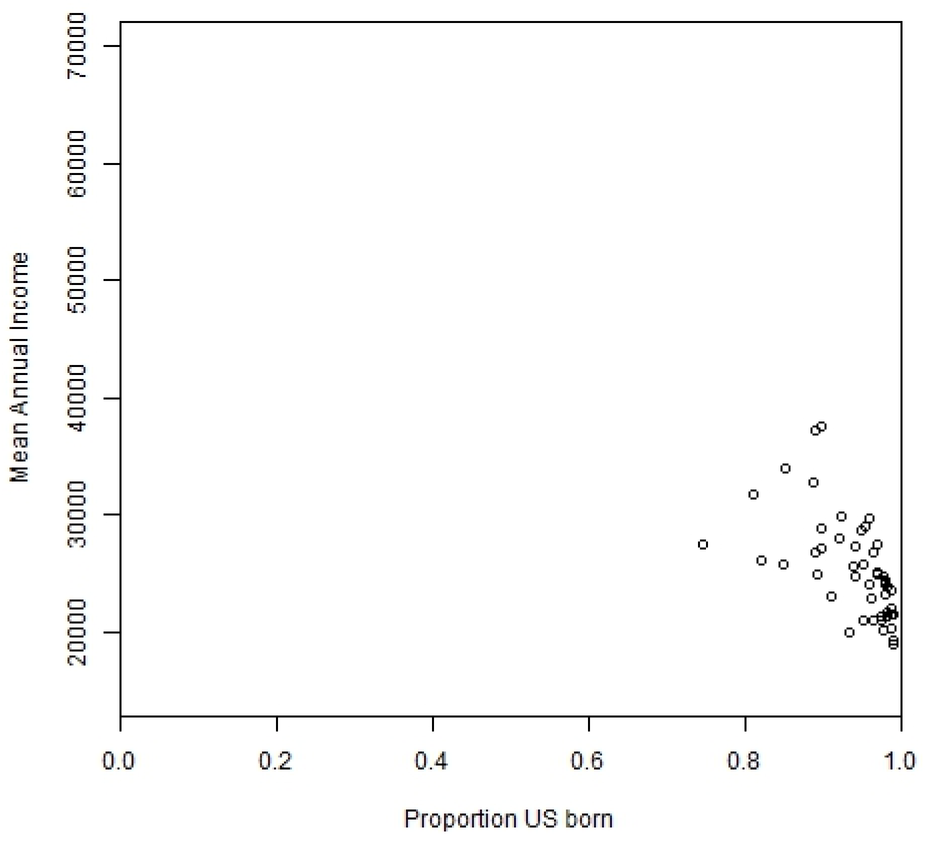
Brand Differentiation

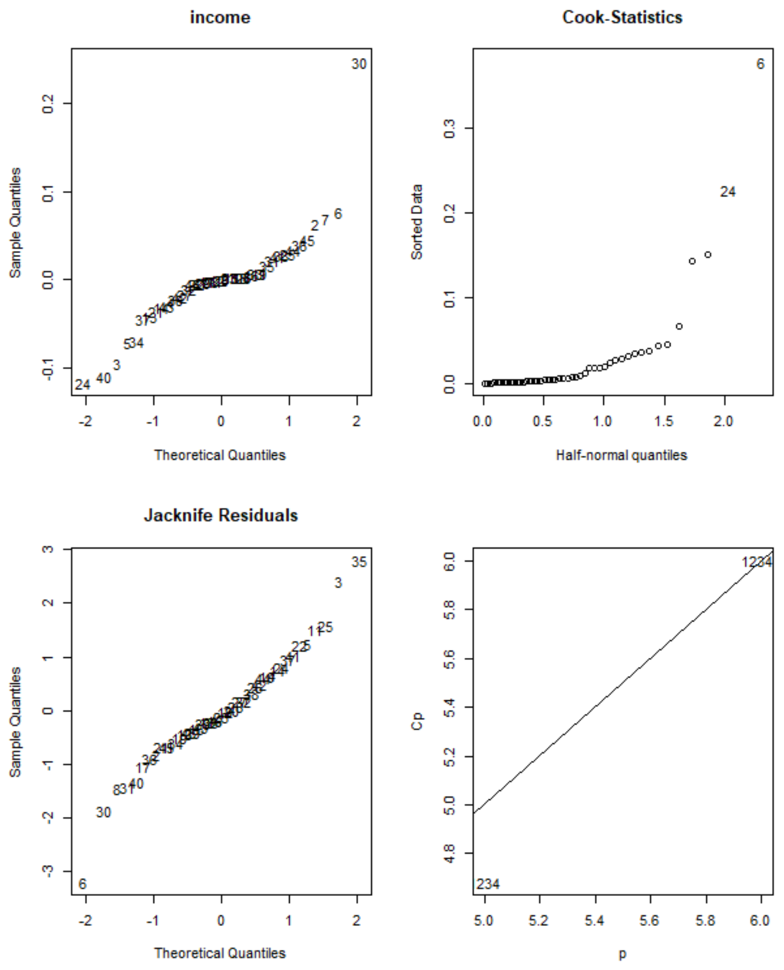



rating.pc
Variances
0
1
2
3
4
1
2
3
4
5
6
7
8
9

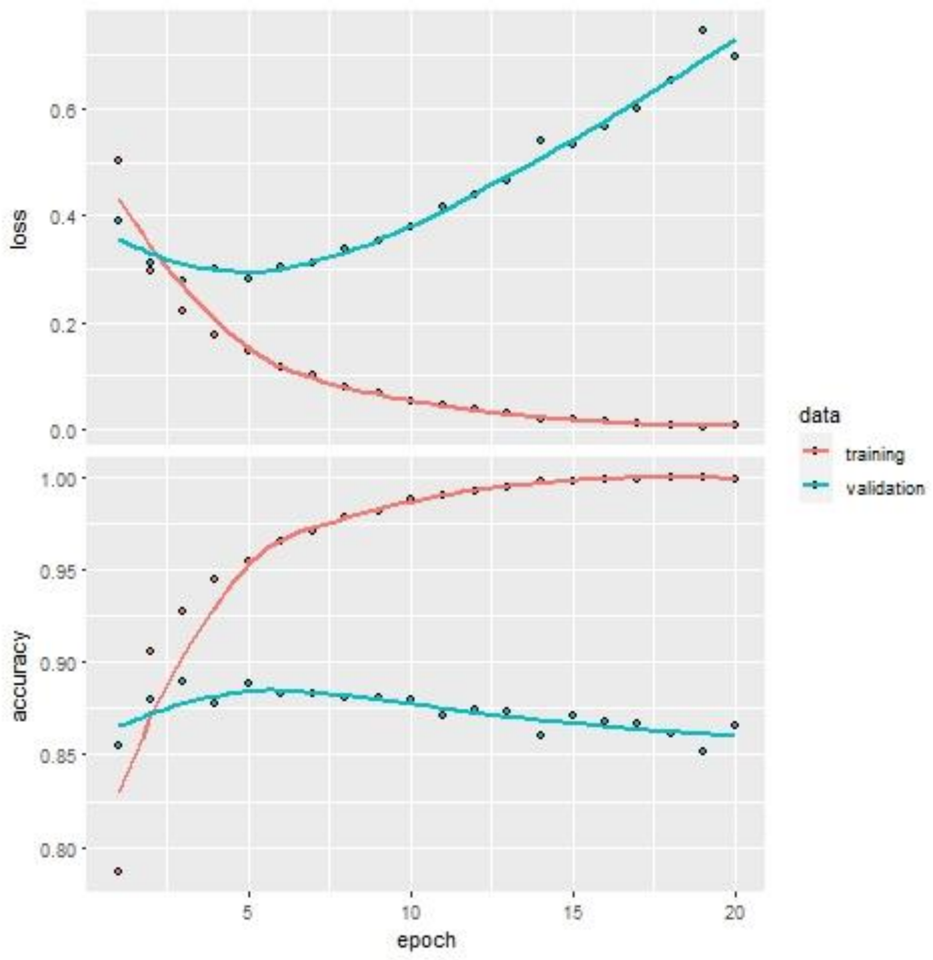
Question 2:

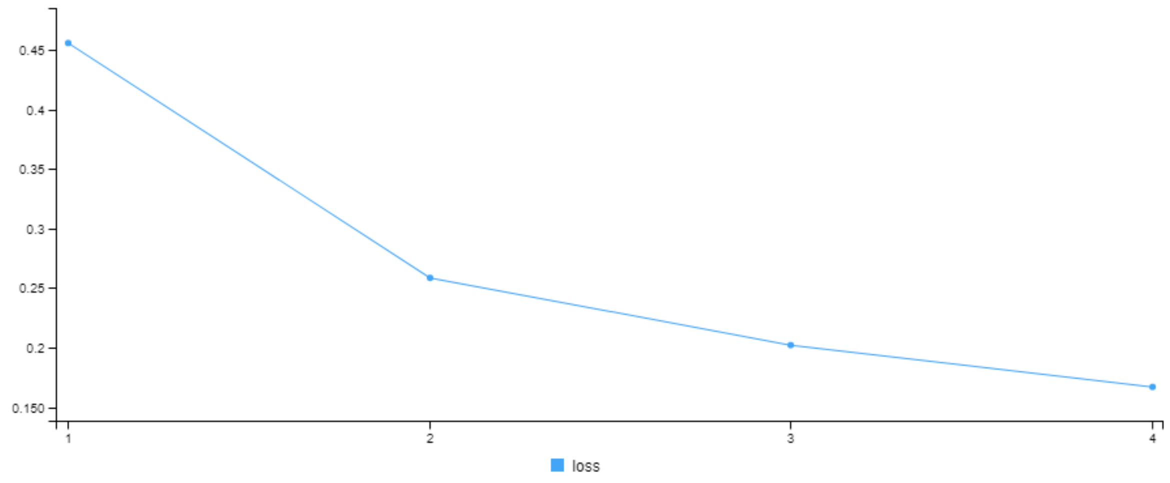

Mean Annual Income
20000
25000
30000
35000
0.75
0.80
0.85
0.90
0.95
Proportion US born

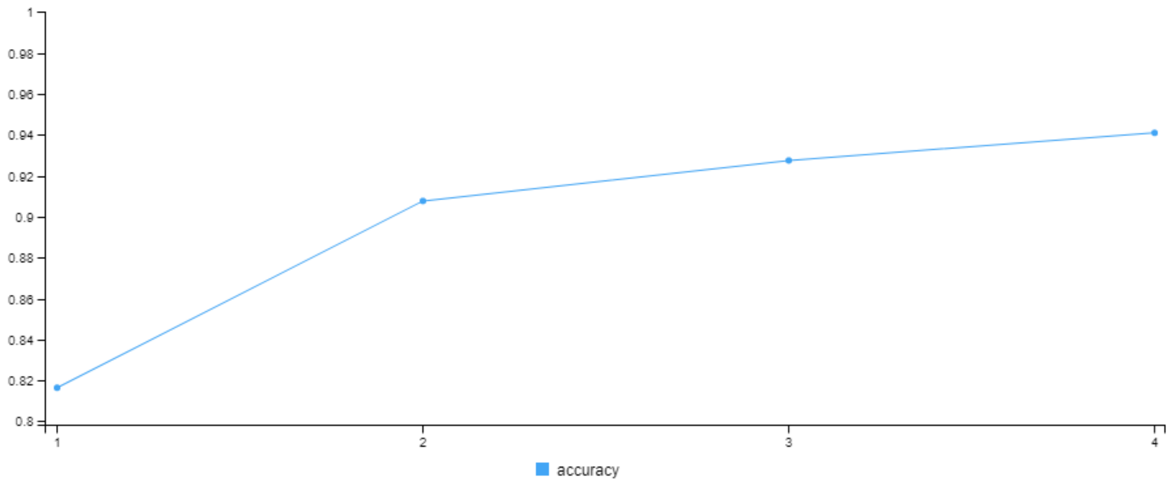

Mean Annual Income
70000
60000
50000
40000
30000
20000
0.0
0.2
0.4
0.6
0.8
1.0
Proportion US born


income
Sample Quantiles
Theoretical Quantiles
Cook-Statistics
Sorted Data
Half-normal quantiles
Jacknife Residuals
Sample Quantiles
Theoretical Quantiles
Cp
p
1234
234

Question 4:


loss
accuracy
epoch
data
training
validation
0.0
0.2
0.4
0.6
0.80
0.85
0.90
0.95
1.00
5
10
15
20


0.45
0.4
0.35
0.3
0.25
0.2
0.150
1
2
3
4
loss


1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
1
2
3
4
accuracy

Interpretation:
Interpretation of the map “Brand Differentiation” is like the map “Brand Positioning.” In conclusion, the closer the positioning of two brands, the more likely they are to compete; the more isolated a brand is on some relevant dimension, the more unique it is. Additionally, considering strength and resources of the company, marketer can consider moving the brand closer to certain target ideal points.
By dividing the data into smaller and smaller subsets it is possible to dilute the significance of any predictor. On the other hand, it is important not to aggregate all data without regard to whether it is reasonable. Clearly a judgment must be made and this often a point of contention in legal cases.
Error improves from 0.1332253 to 0.1310814. This is not much of an improvement, but as noted above, the learning rate will increase and will increase convergence towards the proper weights.
After having trained a network, you will want to use it in a practical setting. You can generate the likelihood of reviews being positive by using the predict method. As you can see, the network is confident for some samples (0.99 or more, or 0.01 or less) but less confident for others (0.7, 0.2).

Conclusion:
1. In this Lab ISE, I learnt about R, RStudio & how to used R Environment.
2. I was also able to implement the PCA for feature extraction.
3. I also did implementation the non-parametric Locally Weighted Regression algorithm in order to fit data points while selecting appropriate data set for your experiment and draw graphs.
4. I built an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.
5. I also built a Deep Learning for Text classification.