CPT\_S 575 Data Science: Assignment 5

Reet Barik

October 21, 2019

## Question 1

1. Estimate the probability that a student who studies for 32 h, has a PSQI score of 12 and has an undergrad GPA of 3.0 gets an A in the class. Show your work.

p = function(x1,x2,x3){ z = exp(-7 + 0.1\*x1 + 1\*x2 - 0.04\*x3); return( round(z/(1+z),4))}  
p(32,3.0,12)

## [1] 0.2176

The chance of a student who studied 32 hours, has a PSQI score of 12 and has undergraduate GPA of 3.0 gets an A in the class is 0.217.

1. How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class? Show your work.

hours = seq(32,50,1)  
probs = mapply(hours, 3.0, 12, FUN=p)  
names(probs) = paste0(hours,"h")  
probs

## 32h 33h 34h 35h 36h 37h 38h 39h 40h 41h   
## 0.2176 0.2351 0.2535 0.2729 0.2932 0.3143 0.3363 0.3589 0.3823 0.4061   
## 42h 43h 44h 45h 46h 47h 48h 49h 50h   
## 0.4305 0.4551 0.4800 0.5050 0.5300 0.5548 0.5793 0.6035 0.6271

It is observed that the student needs to study around 44 to 45 hours to have a 50% chance of getting an A

1. How many hours would a student with a 3.0 GPA and a PSQI score of 3 need to study to have a 50 % chance of getting an A in the class? Show your work.

hours1 = seq(32,45,1)  
probs1 = mapply(hours1, 3.0, 3, FUN=p)  
names(probs1) = paste0(hours1,"h")  
probs1

## 32h 33h 34h 35h 36h 37h 38h 39h 40h 41h   
## 0.2850 0.3058 0.3274 0.3498 0.3729 0.3965 0.4207 0.4452 0.4700 0.4950   
## 42h 43h 44h 45h   
## 0.5200 0.5449 0.5695 0.5939

It is observed that the student needs to study around 41 to 42 hours to have a 50% chance of getting an A with a PSQI score of 3.

## Question 2

library(jsonlite)  
library(plyr)  
require(dplyr)

## Loading required package: dplyr

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(curl)  
library(RCurl)

## Loading required package: bitops

1. Data collection (5%)

api = 'https://content.guardianapis.com/search?'  
key = '9e999f2b-baa3-45e6-87c1-ab38cbc3c9e0'  
url\_test ="https://content.guardianapis.com/search?&section=technology&page-size=100&show-fields=body&api-key=9e999f2b-baa3-45e6-87c1-ab38cbc3c9e0"  
  
webdata = getURL(url\_test)  
categories\_df = fromJSON(webdata)  
lapply(categories\_df,head)

## $response  
## $response$status  
## [1] "ok"  
##   
## $response$userTier  
## [1] "developer"  
##   
## $response$total  
## [1] 52790  
##   
## $response$startIndex  
## [1] 1  
##   
## $response$pageSize  
## [1] 100  
##   
## $response$currentPage  
## [1] 1

sections = c('world','science', 'business','technology','sport','politics')  
fields = 'body'  
pagesize = '100'  
pages = 10  
results = data.frame()  
for(iter in 1:7) {  
 for(section in sections){  
 url = paste(api, '&section=' , section, '&page-size=', pagesize,'&show-fields=', fields,'&api-key=', key, sep ="" )  
 json = fromJSON(url)  
 data = as.data.frame(json$response$results, flatten= TRUE)  
 set = as.data.frame(json$response$results$fields$body, flatten = TRUE)  
 data = subset(data, select = -c(fields))  
 res = cbind(data,set)  
 results = rbind(results, res)  
 print (url)  
 }  
}

colnames(results)[colnames(results) == 'json$response$results$fields$body'] = 'body'

number\_of\_articals = nrow(results)

1. Data cleaning (5%)

results$body = gsub("<.\*?>", "", results$body)   
results$body = gsub("[[:punct:]]", "", results$body)  
results$body = gsub("[[:digit:]]", "", results$body)  
results$body = tolower(results$body)  
  
apply(results, 2, function(x) any(is.na(x)))

## id type sectionId   
## FALSE FALSE FALSE   
## sectionName webPublicationDate webTitle   
## FALSE FALSE FALSE   
## webUrl apiUrl isHosted   
## FALSE FALSE FALSE   
## pillarId pillarName body   
## FALSE FALSE FALSE

number\_of\_articals

## [1] 4200

r\_num = sample(1:number\_of\_articals, 1)  
results$body[r\_num]

## [1] "today’s question is probably the shortest i have ever set find all the ways to arrange four points so that only two distances occur between any two points in other words how many ways are there to draw four dots on a piece of paper such that whichever two dots you choose the distance between these two points is one of two values i’ll give you one solution for free the most obvious one in which the points are arranged in a square the four blue lines are one length and the two green ones are another length with four dots there are six ways to chose a pair of dots so there are six ‘distances’ between dots i learned of this puzzle from maths juggler colin wright who learned it from puzzle maven peter winkler what’s beguiling about the problem is not just the satisfying brevity of the statement but the deceptiveness of the answer “nearly everyone misses at least one solution and for each possible solution it’s been missed by at least one person” said winkler i think it’s a lovely problem because it embraces many levels of difficulty if you struggle with geometry you’ll be overjoyed to find one or two other solutions there are more the business end of the problem however is proving that you have found every solution i’ll be back with the solutions at pm uk time today meanwhile no spoilers although do discuss how you might attack the problem is there a ‘simple’ strategy that gets them all or is it just down to intuition and trialanderror update solution now available here clarifications the four points must be distinct that is no point is allowed to be superimposed on another point for each solution we exclude all reflections rotations and different sizes of that solution also if the two distances that occur between any two of the four points in one solution are a and b then the two distances that appear in another solution are not necessarily a and b i set a puzzle here every two weeks on a monday i’m always on the lookout for great puzzles if you would like to suggest one email me so you think you’ve got problems photograph guardian faber my new book so you think you’ve got problems is out on november it’s a puzzle book but also a book of true stories and mathematical ideas the puzzles i selected all in different ways contain an element of surprise if you enjoy my column i hope you’ll enjoy this book"

1. Tokenization (25%)

suppressMessages(library(quanteda))  
doc.corpus <- corpus(results$body)  
  
# tokenization  
doc.tokens <- tokens(doc.corpus)  
doc.tokens <- tokens(doc.tokens, remove\_punct = TRUE,  
  
remove\_numbers = TRUE)  
  
# removing stopwords  
doc.tokens <- tokens\_select(doc.tokens, stopwords('english'),selection='remove')  
# stemming  
doc.tokens <- tokens\_wordstem(doc.tokens)  
doc.dfm <- dfm(doc.tokens)  
# doc.dfm is a very sparse matrix (99% sparse)  
# we keep only words occurring frequently (top 20%)  
feature\_matrix <- dfm\_trim(doc.dfm, min\_docfreq = 100,  
  
min\_termfreq = 0.2, termfreq\_type = "quantile")  
  
# feature vector of random sample from (b)  
as.vector(feature\_matrix[r\_num, ])

## [1] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0  
## [24] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [47] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [70] 1 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  
## [93] 11 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  
## [116] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [139] 0 0 0 0 0 2 0 0 0 1 0 0 0 0 1 0 10 0 0 0 0 0 0  
## [162] 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 2  
## [185] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0  
## [208] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0  
## [231] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1  
## [254] 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0  
## [277] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  
## [300] 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0  
## [323] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  
## [346] 0 0 0 0 0 0 0 0 0 0 1 0 0 6 0 0 0 0 0 0 0 0 0  
## [369] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0  
## [392] 0 0 0 0 1 0 1 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0  
## [415] 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 1 0 0 0  
## [438] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [461] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0  
## [484] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [507] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0 1 0 0 0 0  
## [530] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0  
## [553] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [576] 2 0 0 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
## [599] 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0  
## [622] 0 0 0 2 0 0 3 0 0 0 2 0 0 0 0 0 0 1 0 0 0 0 0  
## [645] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0  
## [668] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  
## [691] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [714] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [737] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0  
## [760] 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  
## [783] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [806] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [829] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
## [852] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [875] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0  
## [898] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [921] 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [944] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [967] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [990] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1013] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1036] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1059] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1082] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0  
## [1105] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1128] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0  
## [1151] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1174] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1197] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1220] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1243] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0  
## [1266] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1289] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1312] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1335] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1358] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1381] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  
## [1404] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1427] 0 0 0 0 2 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0  
## [1450] 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 1 0 0 0 0 0 0 0  
## [1473] 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 1 1 0  
## [1496] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
## [1519] 0 0 0 1 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0  
## [1542] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0  
## [1565] 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1588] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1611] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0  
## [1634] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1657] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1680] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1703] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0  
## [1726] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0  
## [1749] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1772] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1795] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1818] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  
## [1841] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1864] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1887] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1910] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  
## [1933] 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [1956] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0  
## [1979] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2002] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2025] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2048] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2071] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2094] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2117] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2140] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2163] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2209] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2232] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1  
## [2255] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2278] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0  
## [2301] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2324] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2347] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2370] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0  
## [2393] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0  
## [2416] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2439] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2462] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2485] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [2508] 0 0 0 0 0 0 0 0 0 0 0 0

1. Classification

library(tidytext)   
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(e1071)  
  
#splitting to train adn test data  
# 80% to train data   
matrix = as.matrix(feature\_matrix)  
cor\_Matrix = cor(matrix)  
# find highly correlated features(>=0.8)  
cor\_col\_indices <- findCorrelation(cor\_Matrix, cutoff = 0.80)  
# removing highly correlated features  
matrix <- matrix[, -c(cor\_col\_indices)]  
train\_size <- floor(0.80 \* nrow(matrix)) # 80%-20% split for training and testing  
train\_x <- matrix[1:train\_size,]  
train\_y <- as.factor(results[1:train\_size,]$sectionId)  
test\_x <- matrix[(train\_size+1):nrow(matrix),]  
test\_y <- as.factor(results[(train\_size+1):nrow(matrix),]$sectionId)  
naive\_bayes\_model <- naiveBayes(train\_x, train\_y)  
predictions <- predict(naive\_bayes\_model, test\_x)  
# confusion matrix generated on predictions  
conf\_matrix <- confusionMatrix(predictions, test\_y)  
conf\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business politics science sport technology world  
## business 62 0 0 0 0 0  
## politics 19 196 0 2 7 13  
## science 11 4 99 4 29 11  
## sport 0 0 0 194 0 2  
## technology 7 0 1 0 104 1  
## world 1 0 0 0 0 73  
##   
## Overall Statistics  
##   
## Accuracy : 0.8667   
## 95% CI : (0.8418, 0.8889)  
## No Information Rate : 0.2381   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.836   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: business Class: politics Class: science  
## Sensitivity 0.62000 0.9800 0.9900  
## Specificity 1.00000 0.9359 0.9203  
## Pos Pred Value 1.00000 0.8270 0.6266  
## Neg Pred Value 0.95116 0.9934 0.9985  
## Prevalence 0.11905 0.2381 0.1190  
## Detection Rate 0.07381 0.2333 0.1179  
## Detection Prevalence 0.07381 0.2821 0.1881  
## Balanced Accuracy 0.81000 0.9580 0.9551  
## Class: sport Class: technology Class: world  
## Sensitivity 0.9700 0.7429 0.7300  
## Specificity 0.9969 0.9871 0.9986  
## Pos Pred Value 0.9898 0.9204 0.9865  
## Neg Pred Value 0.9907 0.9505 0.9648  
## Prevalence 0.2381 0.1667 0.1190  
## Detection Rate 0.2310 0.1238 0.0869  
## Detection Prevalence 0.2333 0.1345 0.0881  
## Balanced Accuracy 0.9834 0.8650 0.8643

#precision for each class  
precision = conf\_matrix$byClass[1:6,3]  
print ('Precision:')

## [1] "Precision:"

precision

## Class: business Class: politics Class: science Class: sport   
## 1.0000000 0.8270042 0.6265823 0.9897959   
## Class: technology Class: world   
## 0.9203540 0.9864865

#recall  
recall <- conf\_matrix$byClass[1:6, 1]  
print("Recall:")

## [1] "Recall:"

recall

## Class: business Class: politics Class: science Class: sport   
## 0.6200000 0.9800000 0.9900000 0.9700000   
## Class: technology Class: world   
## 0.7428571 0.7300000