RFL - Choose Your Own Project

Richard Lancaster

20/05/2021

Choose Your Own Project - Richard Lancaster

Predicting Countries of Universities Based on Rankings Indicators

In order to facilitate international collaboration between universities, it is necessary to identify suitable partners, either at the institutional or national level. To facilitate this process, this project aims to predict the country of a university, based on data in 3 indicators of a university rankings data set.

Data set: Times Higher Education (THE) World University Rankings. This contains data on around 1,000 universities around the world, across a wide range of indicators, with the aim of ranking the top universities globally. Data is compiled and released annually.

Indicators: For the purpose of this project, we will use 3 indicators from the data set to make the prediction.

International = a measure of the internationalization at a university, based on data around collaboration. International students = the ratio of international to domestic students Income = the income of the institution

Algorithm: This project will use the k nearest neighbors method to make the predictions.

To start, load the necessary packages and the data set

```
install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/Richard/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)
```

```
##
##   There is a binary version available but the source version is later:
##       binary source needs_compilation
## caret 6.0-86 6.0-88              TRUE
##
##   Binaries will be installed
## package 'caret' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'caret'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\Richard\Documents\R\win-library\4.0\00LOCK\caret\libs\x64\caret.dll to C:
## \Users\Richard\Documents\R\win-library\4.0\caret\libs\x64\caret.dll: Permission
## denied
```

```
## Warning: restored 'caret'
```

```
##
## The downloaded binary packages are in
##  C:\Users\Richard\AppData\Local\Temp\RtmpGiXXeW\downloaded_packages
```

```
install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/Richard/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'dplyr' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'dplyr'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\Richard\Documents\R\win-library\4.0\00LOCK\dplyr\libs\x64\dplyr.dll to C:
## \Users\Richard\Documents\R\win-library\4.0\dplyr\libs\x64\dplyr.dll: Permission
## denied

## Warning: restored 'dplyr'

##
## The downloaded binary packages are in
##  C:\Users\Richard\AppData\Local\Temp\RtmpGiXXeW\downloaded_packages
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

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

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
tinytex::install_tinytex()

THEdata <- read.csv("./timesData.csv")
```

To simplify the calculations, we will limit the data to 1 year (2016, the most recent year), and will use only data on 3 different countries. For the countries, we will select the 1 country with the most universities in the data set, from each of North America, Europe, and Asia.

```r
## order by most universities (per country) in the rankings
THEdata <- THEdata %>% filter(year == 2016) #only using 2016 data
top <- data.frame(table(THEdata$country)) #gives number of universities for each country
top[order(top$Freq, decreasing = TRUE),] #puts into descending order
```

```
##                         Var1 Freq
## 71  United States of America  146
## 70             United Kingdom   77
## 30                      Japan   41
## 10                      China   37
## 19                    Germany   37
## 29                      Italy   34
## 2                   Australia   31
## 18                     France   27
## 8                      Canada   25
## 60                      Spain   25
## 59                South Korea   24
## 63                     Taiwan   24
## 7                      Brazil   17
## 25                      India   17
## 41                Netherlands   13
## 52         Russian Federation   13
## 61                     Sweden   11
## 65                     Turkey   11
## 62                Switzerland   10
## 13             Czech Republic    9
## 17                    Finland    9
## 50        Republic of Ireland    9
## 27                       Iran    8
## 3                     Austria    7
## 6                     Belgium    7
## 21                     Greece    7
## 42                New Zealand    7
## 47                     Poland    7
## 48                   Portugal    7
## 64                   Thailand    7
## 9                       Chile    6
## 14                    Denmark    6
## 22                  Hong Kong    6
## 23                    Hungary    6
## 28                     Israel    6
## 58                South Africa    6
## 38                    Malaysia    5
## 44                      Norway    4
## 51                    Romania    4
## 15                      Egypt    3
## 53                Saudi Arabia    3
## 11                    Colombia    2
## 16                     Estonia    2
## 31                      Jordan    2
## 39                      Mexico    2
## 46                    Pakistan    2
## 55                   Singapore    2
```

```
## 56                  Slovakia    2
## 57                  Slovenia    2
## 67                   Ukraine    2
## 69      United Arab Emirates    2
## 1                   Argentina    1
## 4                  Bangladesh    1
## 5                     Belarus    1
## 12                    Cyprus    1
## 20                     Ghana    1
## 24                   Iceland    1
## 26                 Indonesia    1
## 32                     Kenya    1
## 33                    Latvia    1
## 34                   Lebanon    1
## 35                 Lithuania    1
## 36                Luxembourg    1
## 37                     Macau    1
## 40                   Morocco    1
## 43                   Nigeria    1
## 45                      Oman    1
## 49                     Qatar    1
## 54                    Serbia    1
## 66                    Uganda    1
## 68 Unisted States of America    1
## 72             Unted Kingdom    1
```

The top 3 countries are USA, UK and Japan, which are already one from each of the 3 continents I will use.

Now remove all data except for that of the 3 chosen countries.

```
top3 <- c("United States of America", "United Kingdom", "Japan")
THEdata <- THEdata %>% filter(country %in% top3)
THEdata$country <- factor(THEdata$country)
```

The data contains a lot of NAs, empty cells, and cells with '-'. To avoid errors related to this, all such cells will be removed.

The 'total_score' column contains a lot of empty cells ('-') and is not being used anyway, so this column will be removed first.

```
THEdata <- subset(THEdata, select = -total_score)

## Set all empty cells and those with '-' to NA, which makes it easier to remove them
THEdata[THEdata==""] <- NA
THEdata[THEdata=="-"] <- NA

## Remove all rows with NAs (now including empty cells and '-')
THEdata <- na.omit(THEdata)
```

To allow for the calculations later, the relevant columns need to be changed to numeric, and some characters (%) need to be removed.

```r
set.seed(501)

THEdata$international_students <- as.character(THEdata$international_students)
THEdata$international_students <- strsplit(THEdata$international_students, "%")
THEdata$international_students <- as.numeric(THEdata$international_students)

THEdata$international <- as.character(THEdata$international)
THEdata$international <- as.numeric(THEdata$international)

THEdata$income <- as.character(THEdata$income)
THEdata$income <- as.numeric(THEdata$income)

## Check for NAs in international, international_students, and income (none found)
sum(is.na(THEdata$international))
```

```
## [1] 0
```

```r
sum(is.na(THEdata$international_students))
```

```
## [1] 0
```

```r
sum(is.na(THEdata$income))
```

```
## [1] 0
```

Now we need to create the train, test and validation sets. The validation set will only be used at the final stage.

In order to use the confusion matrix, the train, test and validation sets need to be the same size. Therefore, the split sizes ('p') in createDataPartition are used to get the closest values possible, then the train set and validation set are cut to 109 rows, to match the test set.

```r
## Create validation set (for final test)
use_index <- createDataPartition(y = THEdata$country, times = 1, p = 0.5, list = FALSE)
use_set <- THEdata[-use_index,]
validation_set <- THEdata[use_index,]

## Create test and train sets
test_index <- createDataPartition(y = use_set$country, times = 1, p = 0.9, list = FALSE)
train_set <- THEdata[-test_index,]
test_set <- THEdata[test_index,]

## Check lengths of test, train and validation sets
length(test_set$country) #109
```

```
## [1] 109
```

```r
length(train_set$country) #129
```

```
## [1] 129
```

```r
length(validation_set$country) #119
```

```
## [1] 119
```

```r
## Make test, train and validation sets the same length (109) to avoid problems with confusion matrix
train_set <- train_set[1:109,]
validation_set <- validation_set[1:109,]
```

Now that the data has been wrangled and the necessary sets have been created, we will use the knn algorithm (with default k value) and check the accuracy.

```r
## Predict country based on the values for international students, international score, and income
knn_fit <- knn3(train_set$country ~ train_set$international_students + train_set$international + train_

y_hat_knn <- predict(knn_fit, test_set, type = "class")

## Check accuracy
confusionMatrix(data = y_hat_knn, reference = test_set$country)$overall["Accuracy"]
```

```
##  Accuracy
## 0.4036697
```

```r
###Accuracy = 0.4036697
```

The accuracy is quite low, so we will try to optimize the algorithm by finding the best value of k

```r
## Create empty list to store accuracy values
knn_acc_list = list()

## Use for loop to test k values (from 1 - 40)
for(i in 1:40){
  fit <- knn3(train_set$country ~ train_set$international_students + train_set$international + train_se

y_hat <- predict(fit, train_set, type = "class")

x <-  confusionMatrix(data = y_hat, reference = test_set$country)$overall["Accuracy"]

  knn_acc_list[[i]] = x
}

## Show accuracy for each k value, find max and set max to variable 'k_best'

knn_acc_list
```

```
## [[1]]
##  Accuracy
## 0.4587156
##
## [[2]]
##  Accuracy
## 0.4311927
```

```
##
## [[3]]
## Accuracy
## 0.440367
##
## [[4]]
##  Accuracy
## 0.4311927
##
## [[5]]
##  Accuracy
## 0.4036697
##
## [[6]]
## Accuracy
## 0.412844
##
## [[7]]
##  Accuracy
## 0.4036697
##
## [[8]]
## Accuracy
## 0.412844
##
## [[9]]
##  Accuracy
## 0.4311927
##
## [[10]]
## Accuracy
## 0.440367
##
## [[11]]
##  Accuracy
## 0.4311927
##
## [[12]]
##  Accuracy
## 0.4587156
##
## [[13]]
##  Accuracy
## 0.4587156
##
## [[14]]
##  Accuracy
## 0.4495413
##
## [[15]]
##  Accuracy
## 0.4495413
##
## [[16]]
```

```
##  Accuracy
## 0.4311927
##
## [[17]]
##  Accuracy
## 0.4495413
##
## [[18]]
##  Accuracy
## 0.4495413
##
## [[19]]
##  Accuracy
## 0.4495413
##
## [[20]]
##  Accuracy
## 0.4495413
##
## [[21]]
##  Accuracy
## 0.4495413
##
## [[22]]
##  Accuracy
## 0.4495413
##
## [[23]]
##  Accuracy
## 0.4495413
##
## [[24]]
##  Accuracy
## 0.4495413
##
## [[25]]
##  Accuracy
## 0.4587156
##
## [[26]]
##  Accuracy
## 0.4678899
##
## [[27]]
##  Accuracy
## 0.4770642
##
## [[28]]
##  Accuracy
## 0.4678899
##
## [[29]]
##  Accuracy
## 0.4770642
```

```
##
## [[30]]
##  Accuracy
## 0.4862385
##
## [[31]]
##  Accuracy
## 0.4862385
##
## [[32]]
##  Accuracy
## 0.5045872
##
## [[33]]
##  Accuracy
## 0.4954128
##
## [[34]]
##  Accuracy
## 0.5045872
##
## [[35]]
##  Accuracy
## 0.4954128
##
## [[36]]
##  Accuracy
## 0.5045872
##
## [[37]]
##  Accuracy
## 0.4954128
##
## [[38]]
##  Accuracy
## 0.4954128
##
## [[39]]
##  Accuracy
## 0.4954128
##
## [[40]]
##  Accuracy
## 0.4954128
```

```
which.max(knn_acc_list) #32
```

```
## [1] 32
```

```
k_best <- which.max(knn_acc_list)
```

Now we have the best value of k (=32) we will use this value and perform our final test with the validation set

```r
## Use knn with k = k_best
knn_fit <- knn3(train_set$country ~ train_set$international_students + train_set$international + train_s

y_hat_knn <- predict(knn_fit, validation_set, type = "class")

## Check accuracy
confusionMatrix(data = y_hat_knn, reference = validation_set$country)$overall["Accuracy"]
```

```
##  Accuracy
## 0.4678899
```

```r
## Give full confusion matrix output
confusionMatrix(data = y_hat_knn, reference = validation_set$country)
```

```
## Confusion Matrix and Statistics
##
##                           Reference
## Prediction                 Japan United Kingdom United States of America
##   Japan                        0              2                         0
##   United Kingdom               3             11                        24
##   United States of America     7             22                        40
##
## Overall Statistics
##
##                Accuracy : 0.4679
##                  95% CI : (0.3717, 0.5659)
##     No Information Rate : 0.5872
##     P-Value [Acc > NIR] : 0.99541
##
##                   Kappa : -0.0339
##
##  Mcnemar's Test P-Value : 0.06329
##
## Statistics by Class:
##
##                      Class: Japan Class: United Kingdom
## Sensitivity               0.00000                0.3143
## Specificity               0.97980                0.6351
## Pos Pred Value            0.00000                0.2895
## Neg Pred Value            0.90654                0.6620
## Prevalence                0.09174                0.3211
## Detection Rate            0.00000                0.1009
## Detection Prevalence      0.01835                0.3486
## Balanced Accuracy         0.48990                0.4747
##                      Class: United States of America
## Sensitivity                                   0.6250
## Specificity                                   0.3556
## Pos Pred Value                                0.5797
## Neg Pred Value                                0.4000
## Prevalence                                    0.5872
## Detection Rate                                0.3670
## Detection Prevalence                          0.6330
## Balanced Accuracy                             0.4903
```

```
###Accuracy = 0.4678899
```

The final accuracy is 0.467889

Although this is quite low, as there are 3 possible options (countries) it is significantly better than guessing.

The sensitivity and specificity vary significantly for each of the 3 countries. Further work is necessary to provide more consistency, and to see how this model could be expanded to include all countries (many of which have few institutions in the rankings data).

Following this preliminary project, further work can be done to see how the use of different indicators affects the accuracy of the algorithm.