Statistical Learning final project: analyzing student performances using the Open University Learning Analytics dataset

Camilo Betancourt Nieto, Luisa Fernanda Medina Morales and Esteban Ortega Dominguez

Contents

1. Introduction

This project aims to draw conclusions from a dataset using statistical methods learnt during the Statistical Learning course. This process might involve finding relationships among variables implementing regression or classification models. The regression tasks model the relationship between explanatory variables and an outcome (or response variable). On the other hand, as evident as it may be, the classification tasks aim to

classify data points into groups. In our case, due to the nature of our data, we tackled a classification problem by using five different approaches: logistic classifiers (which involve logistic regressions), linear discriminant analysis, quadratic discriminant analysis, the naive Bayes model and the K-Nearest Neighbors algorithm. However, we not only wanted to classify the data, but we also aimed to understand the information we had. With this in mind, first, we need to talk about our input.

1.1 Dataset obtainance and description

For our project, we are going to work with the Open University Learning Analytics dataset, which is publicly available in https://analyse.kmi.open.ac.uk/open_dataset. It is a database that contains information about seven courses (which are also referenced as modules) that were held in the Virtual Learning Environment (VLE) of the Open University (OU). Additionally, it contains anonymised data about the students and their interactions with the course materials. According to this, **we want to understand which variables can affect the students performance (and how)**.

The database has seven tables: `courses`, `assessments`, `vle`, `studentInfo`, `studentRegistration`, `studentAssessment` and `studentVle` The UML diagram for the database and the detailed documentation with the description of the tables and their columns are found in the link provided. Also, there is an article in which the creators of the dataset describe the information more in depth: https://www.nature.com/articles/sdata2017171. However, in order to make our project easy to understand, we give a brief description of the most relevant information for our work.

1.1.1 `studentInfo`:

This is going to be our most important table, since it contains the demographic information about the students, together with their course final results. According the documentation, the column description is as follows:

- `code_module`: an identification code for a module on which the student is registered.
- `code_presentation`: the identification code of the presentation during which the student is registered on the module. It consists of the year and "B" for the presentation starting in February and "J" for the presentation starting in October.
- `id_student`: a unique identification number for the student.
- `gender`: the student's gender.
- `region`: identifies the geographic region, where the student lived while taking the module-presentation.
- `highest_education`: highest student education level on entry to the module presentation.
- `imd_band`: specifies the Index of Multiple Deprivation band of the place where the student lived during the module-presentation. About this index, the documentation leads us to Wikipedia, where it is stated that it is "essentially a measure of poverty".
- `age_band`: band of the student's age.
- `num_of_prev_attempts`: the number times the student has attempted this module.
- `studied_credits`: the total number of credits for the modules the student is currently studying.
- `disability`: indicates whether the student has declared a disability.
- `final_result`: student's final result in the module-presentation.

The head of the table helps us understand its structure:

```
##   code_module code_presentation id_student gender               region
## 1         AAA             2013J      11391      M  East Anglian Region
## 2         AAA             2013J      28400      F             Scotland
## 3         AAA             2013J      30268      F North Western Region
## 4         AAA             2013J      31604      F    South East Region
```

```
## 5         AAA             2013J      32885      F West Midlands Region
## 6         AAA             2013J      38053      M                Wales
##       highest_education imd_band age_band num_of_prev_attempts studied_credits
## 1      HE Qualification  90-100%     55<=                     0             240
## 2      HE Qualification   20-30%    35-55                     0              60
## 3 A Level or Equivalent   30-40%    35-55                     0              60
## 4 A Level or Equivalent   50-60%    35-55                     0              60
## 5    Lower Than A Level   50-60%     0-35                     0              60
## 6 A Level or Equivalent   80-90%    35-55                     0              60
##   disability final_result
## 1          N         Pass
## 2          N         Pass
## 3          Y    Withdrawn
## 4          N         Pass
## 5          N         Pass
## 6          N         Pass
```

An important consideration regarding this table is that it has a compound primary key, since it is the combination of the `code_module`, `code_presentation` and `id_student` columns. Therefore, each observation is a student's ***course-presentation (or module-presentation)***. A student can participate in different courses and can do it more than one time. In total, we have 32,593 rows. Also, it is worth noting that the `final_result` column contains categorical data, which is suitable for a classification task.

1.1.2 Other tables:

According to the documentation, usually, every course-presentation has a number of assessments followed by the final exam. Moreover, there are three types of assessments: Tutor Marked Assessment (TMA), Computer Marked Assessment (CMA) and Final Exam (Exam). This information is contained in the `assessment_type` column of the `assessments` table.

Also, the `studentAssessment` table contains the column `score`, that is the student's score for a given assessment (identified with the `id_assessment` column, which is present in both tables). The range is from 0 to 100. A score lower than 40 is interpreted as Fail.

At first we were tempted to use the `score` column to carry out a separate regression model, but as it will be explained and justified later on, this option was discarded because of the quality of the data in this regard.

On the other hand, the `studentVle` table has some features that might help us understand how well the students perform in the courses: the `sum_click` column. This is the number of times a student interacts with the material in a given day, which is identified with the `date` column of the same table (the date is measured as the number of days since the start of the module-presentation).

Lastly, the `studentRegistration` table has the `date_registration` column, which corresponds to "the date of student's registration on the module presentation, this is the number of days measured relative to the start of the course presentation (e.g. the negative value -30 means that the student registered to module presentation 30 days before it started)".

Based on this information, we highlight that the `final_result` column, which is categorical -and therefore adequate for a classification task- is going to be our "target variable". This means that, using this column, we would like to measure the performance of the students with respect to their features (which are the other columns).

2. Data preparation and cleaning

In order to do an appropriate analysis, we have to perform an initial exploration and then prepare our data in order to make it functional for our models. We divided this process in three sections. The first one corresponds to the pre-processing of the `student_info` table, while the second one corresponds to a series of procedures made with the `studentAssessment` and `assessments` tables that allowed us to define our scope more precisely. Finally, the last section corresponds to the final steps we took to obtain our "objective table" (the one used for the analysis). However, it is important to note that afterwards, when our complete exploratory data analysis was made, we performed further processing when needed.

2.1 `studentInfo` preparation and cleaning

For the `studentInfo` table, it is useful to do an initial exploration by means of a descriptive statistical summary of the columns (we had already converted the categorical features into factors):

```
##  code_module code_presentation   id_student    gender
##  AAA: 748    2013B: 4684       584077 :    5   F:14718
##  BBB:7909    2013J: 8845       80329  :    4   M:17875
##  CCC:4434    2014B: 7804       157310 :    4
##  DDD:6272    2014J:11260       172965 :    4
##  EEE:2934                      279883 :    4
##  FFF:7762                      387276 :    4
##  GGG:2534                      (Other):32568
##                   region                          highest_education
##  Scotland            : 3446   A Level or Equivalent      :14045
##  East Anglian Region : 3340   HE Qualification           : 4730
##  London Region       : 3216   Lower Than A Level         :13158
##  South Region        : 3092   No Formal quals            :  347
##  North Western Region: 2906   Post Graduate Qualification:  313
##  West Midlands Region: 2582
##  (Other)             :14011
##     imd_band      age_band     num_of_prev_attempts studied_credits  disability
##  20-30% : 3654   0-35 :22944   Min.   :0.0000       Min.   : 30.00   N:29429
##  30-40% : 3539   35-55: 9433   1st Qu.:0.0000       1st Qu.: 60.00   Y: 3164
##  10-20  : 3516   55<= :  216   Median :0.0000       Median : 60.00
##  0-10%  : 3311                 Mean   :0.1632       Mean   : 79.76
##  40-50% : 3256                 3rd Qu.:0.0000       3rd Qu.:120.00
##  50-60% : 3124                 Max.   :6.0000       Max.   :655.00
##  (Other):12193
##       final_result
##  Distinction: 3024
##  Fail       : 7052
##  Pass       :12361
##  Withdrawn  :10156
##
##
##
```

First we focus our attention on the the Index of Multiple Deprivation variable, which is stored in the `imd_band` column as a label that accounts for the band in which the measures fall. This column contains 1,111 rows (3.41% of the data) without a value:

imd_band	row_count	row_count_percentage
0-10%	3311	10.16
10-20	3516	10.79
20-30%	3654	11.21
30-40%	3539	10.86
40-50%	3256	9.99
50-60%	3124	9.58
60-70%	2905	8.91
70-80%	2879	8.83
80-90%	2762	8.47
90-100%	2536	7.78
NA	1111	3.41

It is not a huge portion of the data, yet we still wanted to investigate a bit further. We checked for anomalies by selecting only the rows that have NA in the `imd_band` column and checking the statistics summary per attribute. From this, we did not find any patterns that stood out in relation to the summary of the rest of the table:

```
##  code_module code_presentation   id_student   gender
##  AAA: 23     2013B:177         544649 :   3   F:303
##  BBB: 63     2013J:267         551528 :   3   M:808
##  CCC:254     2014B:272         557700 :   3
##  DDD:280     2014J:395         587576 :   3
##  EEE:117                       2080916:   3
##  FFF:367                       29639  :   2
##  GGG:  7                       (Other):1094
##                    region                        highest_education    imd_band
##  North Region        :731   A Level or Equivalent      :283     0-10%  :   0
##  Ireland             :266   HE Qualification           :286     10-20  :   0
##  South Region        : 48   Lower Than A Level         :396     20-30% :   0
##  West Midlands Region: 39   No Formal quals            : 19     30-40% :   0
##  Scotland            : 12   Post Graduate Qualification:127     40-50% :   0
##  North Western Region:  5                                       (Other):   0
##  (Other)             : 10                                       NA's   :1111
##   age_band   num_of_prev_attempts studied_credits disability      final_result
##  0-35 :672   Min.   :0.0000       Min.   : 30.0   N:1078     Distinction:199
##  35-55:424   1st Qu.:0.0000       1st Qu.: 60.0   Y:  33     Fail       :145
##  55<= : 15   Median :0.0000       Median : 60.0              Pass       :531
##              Mean   :0.1269       Mean   : 79.5              Withdrawn  :236
##              3rd Qu.:0.0000       3rd Qu.: 90.0
##              Max.   :3.0000       Max.   :480.0
##
```

We could replace the missing values for the most frequent value, which in this case is the 20-30% band, however, we will just treat them as missing values for now until we have more information.

On the other hand, since the bands of the Index of Multiple Deprivation represent a quantitative variable whose range goes from 0 to 100, it makes sense for us to treat it as a numeric feature instead of a categorical one. That is why we convert each label into a number that conserves the distance among categories, so we are going to pick the middle point of each band (25% for the 20-30% band, for example). After the transformation, the heads of the columns look like this:

imd_band	imd_band_middle
90-100%	0.95
20-30%	0.25
30-40%	0.35
50-60%	0.55
50-60%	0.55
80-90%	0.85

Alternatively, the `age_band` variable has only three categories ('0-35', '35-55' and '55<=') so it has already lost a lot of granularity. Also, from the descriptive statistical summary, we noticed that the '55<=' age band only accounts for 0.66%, so it has low statistical power on its own. For these reasons, instead of using this column as a numeric vector that tries to mimic the ages, we decided to merge the '35-55' and '55<=' bands in order to treat this predictor as a categorical variable with two groups: '0-35' and '<=35':

```
# We need to convert the column to character because it was read as a factor directly.
# After merging the '35-55' and '55<=' bands, we convert the column back into a factor.
studentInfo$age_band = as.character(studentInfo$age_band)
studentInfo$age_band = fct_collapse(studentInfo$age_band,'35<=' = c('35-55', '55<='))
studentInfo$age_band = as.factor(studentInfo$age_band)
```

Likewise, with respect to the `highest_education` feature, we highlight that the 'No Formal quals' and the 'Post Graduate Qualification' categories only represent 1.06% and 0.96% of the rows, respectively. On the other hand, we consider that this column has an ordinal nature. Taking this into account and using some information published in a governmental webpage that briefly describes the educational levels in the UK (where the Open University is from), we reckon that the correct order of the categories should be: 'Lower Than A Level', 'A Level or Equivalent', and 'HE Qualification', in ascending order. In this ordering, the 'No Formal quals' and the 'Lower Than A Level' labels are merged into the 'Lower Than A Level' category, while the 'HE Qualification' and 'Post Graduate Qualification' labels are consolidated into the 'HE Qualification' group. Then, we convert the column into a an ordered factor variable:

```
# First we regroup the categories
studentInfo$highest_education <- as.character(studentInfo$highest_education)
studentInfo$highest_education <- fct_collapse(studentInfo$highest_education,
                                              'Lower Than A Level'=c('No Formal quals',
                                                                     'Lower Than A Level'))
studentInfo$highest_education <- fct_collapse(studentInfo$highest_education,
                                              'HE Qualification'=c('HE Qualification',
                                                                   'Post Graduate Qualification'))

# Then we convert the column into an ordered factor variable
studentInfo$highest_education = factor(studentInfo$highest_education,
                                       levels = c('Lower Than A Level',
                                                  'A Level or Equivalent',
                                                  'HE Qualification'),
                                       ordered = TRUE)
```

Finally, we center our attention on the `final_result` column, which could be our target variable for a classification task. However, this column originally contains four groups: *Distinction*, *Fail*, *Pass* and *Withdrawn*. Since we would potentially like to compare different classification models and, within the scope of the Statistical Learning course, the logistic regression focuses on two-class problems, we are going to consider only two labels: *Successful* (which comprises *Pass* and *Distinction*) and *Unsuccessful* (which includes both *Fail* and *Withdrawn*).

```r
# Again, we convert the column to character for the update
# and then change it back it into a factor.
studentInfo$final_result <- as.character(studentInfo$final_result)
studentInfo$final_result <- fct_collapse(studentInfo$final_result,
                                         'Successful'= c ('Pass', 'Distinction'))
studentInfo$final_result <- fct_collapse(studentInfo$final_result,
                                         'Unsuccessful' = c('Fail', 'Withdrawn'))
studentInfo$final_result <- as.factor(studentInfo$final_result)
```

This update also leaves us a more balanced variable. Before, the minority class (*Distinction*), corresponded to 9.28% of the observations, while the majority class (*Pass*) represented 37.93% of the data. Now, the *Successful* and *Unsuccessful* shares are 47.2% and 52.8%", respectively.

The rest of the columns can be left as they are.

2.1 Procedures regarding `studentAssesment` and `assessment`

We were curious about the fact that the final results of the modules were given only as a categorical variable with very few labels, not as a continuous variable. However, the database included two tables that were related to the scores of each assessment: `studentAssesment` and `assessment`. For this reason and to gain better understanding of our dataset, we investigated a little about the contents of these tables.

First, when looking at a summary of the `studentAssesment` (the table that collects the scores of the students for each assignment). we simply noticed that we had 173 NAs in the `score` column. In this case, this represents less than 0.1% of the records and is explained by the documentation: "if the student did not submit the assessment, no result was recorded".

id_assessment	id_student	date_submitted	is_banked	score
24295 : 1917	537811 : 28	Min. :-11	Min. :0.00000	Min. : 0.0
34873 : 1859	554881 : 26	1st Qu.: 51	1st Qu.:0.00000	1st Qu.: 65.0
34899 : 1826	632074 : 25	Median :116	Median :0.00000	Median : 80.0
15020 : 1776	358075 : 24	Mean :116	Mean :0.01098	Mean : 75.8
14996 : 1695	405314 : 24	3rd Qu.:173	3rd Qu.:0.00000	3rd Qu.: 90.0
34874 : 1661	443696 : 24	Max. :608	Max. :1.00000	Max. :100.0
(Other):163178	(Other):173761	NA	NA	NA's :173

Given that there is only a very small percentage of rows with NAs (and we also checked that they were not concentrated in any specific group), we decided to remove them from the table. However, as we will see, the scores were not part of our analysis, so omitting these values has no effect on our results.

No information was relevant from the previous summary other than the fact that scores are mostly complete within that table, so, in order to validate the way we should interpret these scores, we then focused on the types of assessment the students made. With this in mind, we brought the `assessmentType` column from the `assessments` table and found that, even though the final exams are quite important (since the documentation states that "exams are treated separately and have the weight equal to 100%; the sum of all other assessments is also 100%"), most of the assessments recorded in the database did not actually correspond to exams:

	Number of assessments recorded
CMA	70527
Exam	4959
TMA	98253

Number of assessments recorded

According to the dataset creators, "results of the final exam are usually missing (since they are scored and used for the final marking immediately at the end of the module)". This would represent a serious issue if we wanted to do a regression task. We could not perform regression with respect to a numerical continuous version of the final course marks nor the result of the final exams because we only have a categorical variable that accounts for the final result of the course, while many of the final exam scores are missing; therefore, we cannot use one of these values to correctly compute the other. Furthermore, only 6 module-presentations had recorded results for the final exam (there are 22 course-presentations in total):

code_module	code_presentation	assessment_type	number_of_assessments
CCC	2014B	Exam	747
CCC	2014J	Exam	1168
DDD	2013B	Exam	602
DDD	2013J	Exam	968
DDD	2014B	Exam	524
DDD	2014J	Exam	950

For these reasons, since just a small portion of the dataset contains information regarding the individual scores for the exams (the most important assessment), we decided that **it is reasonable to stick with the classification problem** and discard any sort of separate regression problem regarding the final exam. Also, just as a side note, since the other assessments are just a component of the final result of the course, it would not make sense to include them in the analysis. They are just part of the target variable, so predicting the `final_result` with respect to some of the grades that led into that result would not add any value to the data.

2.3 Our 'objective' table for the statistical analysis

According to what we have seen so far, we can we can delineate the scope of the project as follows:

- Classification with respect to the (updated) `final_result` variable for all of the students' course-presentations: We can model how the explanatory variables are related to the `final_result` groups: *Successful* and *Unsuccessful*. This way, we can use the whole dataset and not only the cases in which the final exam was recorded.

A few steps more were required to finally obtain our 'objective' table. With this, we mean the table that is going to be used for the rest of the analysis and that is organized so that we can easily pull the predictors and the response variable for each observation. For this purpose, we just needed to bring information from two columns: `sum_click` from the `studentVLE` table and `date_registration` from the `studentRegistration` table.

The `sum_click` column gives us the amount of times a student interacts with the course materials in a day (and there is a column that tells us the date of the students' interactions measured as the number of days since the start of the module presentation). Therefore, we can group these values by `code_module`, `code_presentation` and `id_student`, so that we can obtain the total number of clicks (`total_sum_click`) for each student and each course-presentation. For the students' group presentations that did not have interactions, we filled the values with 0 in this new column and conducted further exploration later on.

we checked if there were any characteristics that stood out from the observations with no interactions and found that it happens almost exclusively with unsuccessful cases:

```r
groupedInteractions = (studentVle %>%
                         group_by(code_module, code_presentation, id_student) %>%
                         summarise(total_sum_click = sum(sum_click),
                                   .groups = 'keep'))

studentInfo = merge(studentInfo,
                    groupedInteractions,
                    by = c('code_module','code_presentation','id_student'),
                    all.x = TRUE)

studentInfo$total_sum_click = coalesce(studentInfo$total_sum_click, 0)
summary(studentInfo[studentInfo$total_sum_click==0,])
```

```
##  code_module code_presentation   id_student   gender
##  AAA:  13    2013B: 423        387276 :   4   F:1776
##  BBB:1287    2013J: 872        271702 :   3   M:1589
##  CCC: 451    2014B: 953        339353 :   3
##  DDD: 527    2014J:1117        466430 :   3
##  EEE: 249                      565275 :   3
##  FFF: 670                      577352 :   3
##  GGG: 168                      (Other):3346
##                   region                highest_education    imd_band
##  London Region       : 402   Lower Than A Level   :1678    0-10%  : 482
##  North Western Region: 384   A Level or Equivalent:1284    10-20  : 478
##  East Anglian Region : 352   HE Qualification     : 403    20-30% : 449
##  West Midlands Region: 317                                 30-40% : 346
##  East Midlands Region: 291                                 40-50% : 341
##  South Region        : 289                                 (Other):1212
##  (Other)             :1330                                 NA's   :  57
##  age_band    num_of_prev_attempts studied_credits  disability
##  0-35:2473   Min.   :0.0000       Min.   : 30.00   N:3019
##  35<=: 892   1st Qu.:0.0000       1st Qu.: 60.00   Y: 346
##              Median :0.0000       Median : 90.00
##              Mean   :0.2042       Mean   : 95.15
##              3rd Qu.:0.0000       3rd Qu.:120.00
##              Max.   :4.0000       Max.   :655.00
##
##       final_result  imd_band_middle  total_sum_click
##  Successful  :   3   Min.   :0.0500   Min.   :0
##  Unsuccessful:3362   1st Qu.:0.1500   1st Qu.:0
##                      Median :0.3500   Median :0
##                      Mean   :0.4155   Mean   :0
##                      3rd Qu.:0.6500   3rd Qu.:0
##                      Max.   :0.9500   Max.   :0
##                      NA's   :57
```

Although there are 3,365 rows with no interactions, the fact that these cases are concentrated only on the 'Unsuccesful' group is a motivation for us to remove these observations. It refers to a group of students that did not really participate on the courses in any way, yet our goal is to measure what makes a student successful or not when they actually took part of the modules somehow. Therefore, keeping these examples would distort our data with respect to our classification task.

```
studentInfoFiltered = studentInfo[studentInfo$total_sum_click !=0,]
```

On the other hand, `date_registration` column gives us the date of student's registration on the module presentation, measured as the number of days with respect to the start of the module presentation. Since the course resources are available from the VLE system a few weeks before the start of the presentation, there could exist some negative values. This may allow us to see if registering to the modules in advance (or late) can affect the students' performance. It is worth mentioning that there were 7 observations (0.02% of the data) without a registration date. Since it was a very small percentage, we imputed this value as the rounded mean.

```
studentInfoFiltered = merge(studentInfoFiltered,
                            studentRegistration[,c('code_module','code_presentation',
                                                   'id_student','date_registration')],
                            by=c('code_module','code_presentation','id_student'))

studentInfoFiltered$date_registration = coalesce(studentInfoFiltered$date_registration,
                                                 round(mean(studentInfoFiltered$date_registration,
                                                            na.rm = TRUE)))
```

Finally, we obtained our 'objective' table for the classification problem. Just for the sake of clarity, from now on we are going to call it `courseResults`. In this new table we merged the `code_module`, `code_presentation` and `id_student` columns in a unique identifier called `id_module_presentation_student`, just in case we need it afterwards. That way we can drop the `id_student` column since it is not useful as a predictor. In contrast, we keep the `code_module` and `code_presentation` just to check if there is any difference in the performance of the students among modules and the periods when the courses were taken. **The resulting table has 29.228 observations and the following structure**:

```
##   id_module_presentation_student code_module code_presentation gender
## 1              AAA_2013J_100893         AAA             2013J      M
## 2              AAA_2013J_101781         AAA             2013J      M
## 3              AAA_2013J_102806         AAA             2013J      M
## 4              AAA_2013J_102952         AAA             2013J      M
## 5             AAA_2013J_1035023         AAA             2013J      F
## 6              AAA_2013J_104476         AAA             2013J      M
##                 region     highest_education imd_band imd_band_middle age_band
## 1     Yorkshire Region A Level or Equivalent   20-30%            0.25     0-35
## 2         South Region    Lower Than A Level   80-90%            0.85     35<=
## 3 North Western Region A Level or Equivalent   30-40%            0.35     0-35
## 4        London Region      HE Qualification   60-70%            0.65     0-35
## 5        London Region    Lower Than A Level   30-40%            0.35     35<=
## 6              Ireland      HE Qualification     <NA>              NA     35<=
##   num_of_prev_attempts studied_credits disability total_sum_click
## 1                    0              60          N             744
## 2                    0              60          N            4104
## 3                    0              60          N            1944
## 4                    0              60          N            1150
## 5                    0              60          N            1896
## 6                    0              60          N            4777
##   date_registration final_result
## 1               -62   Successful
## 2              -103   Successful
## 3              -107   Successful
## 4              -100   Successful
```

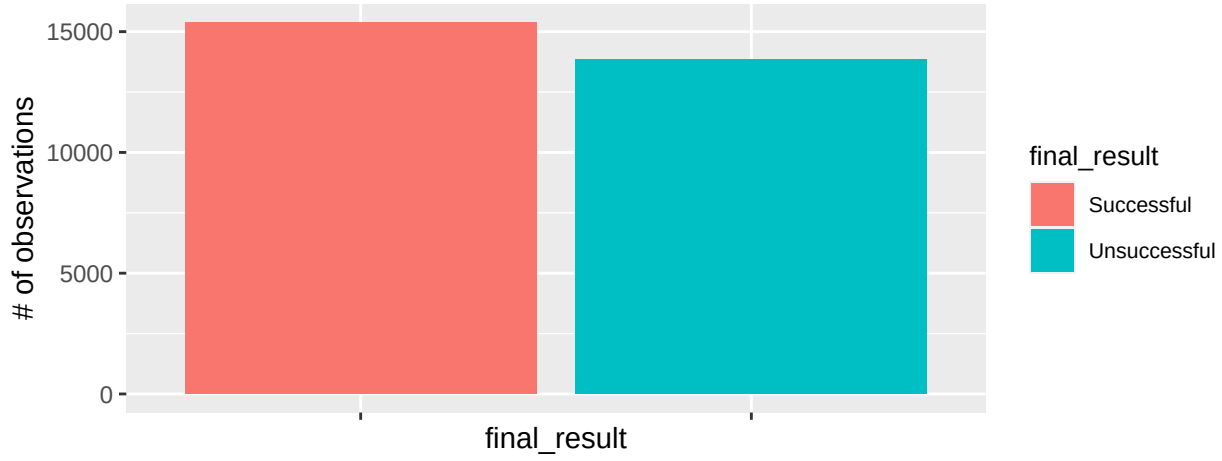
```
## 5               -87   Successful
## 6              -109   Successful
```

This `courseResults` table included the `imd_band` just for plotting during the visualizations, but was then removed for the statistical models (as this column was replaced by the `imd_band_middle`).

3. Exploratory data analysis and further processing

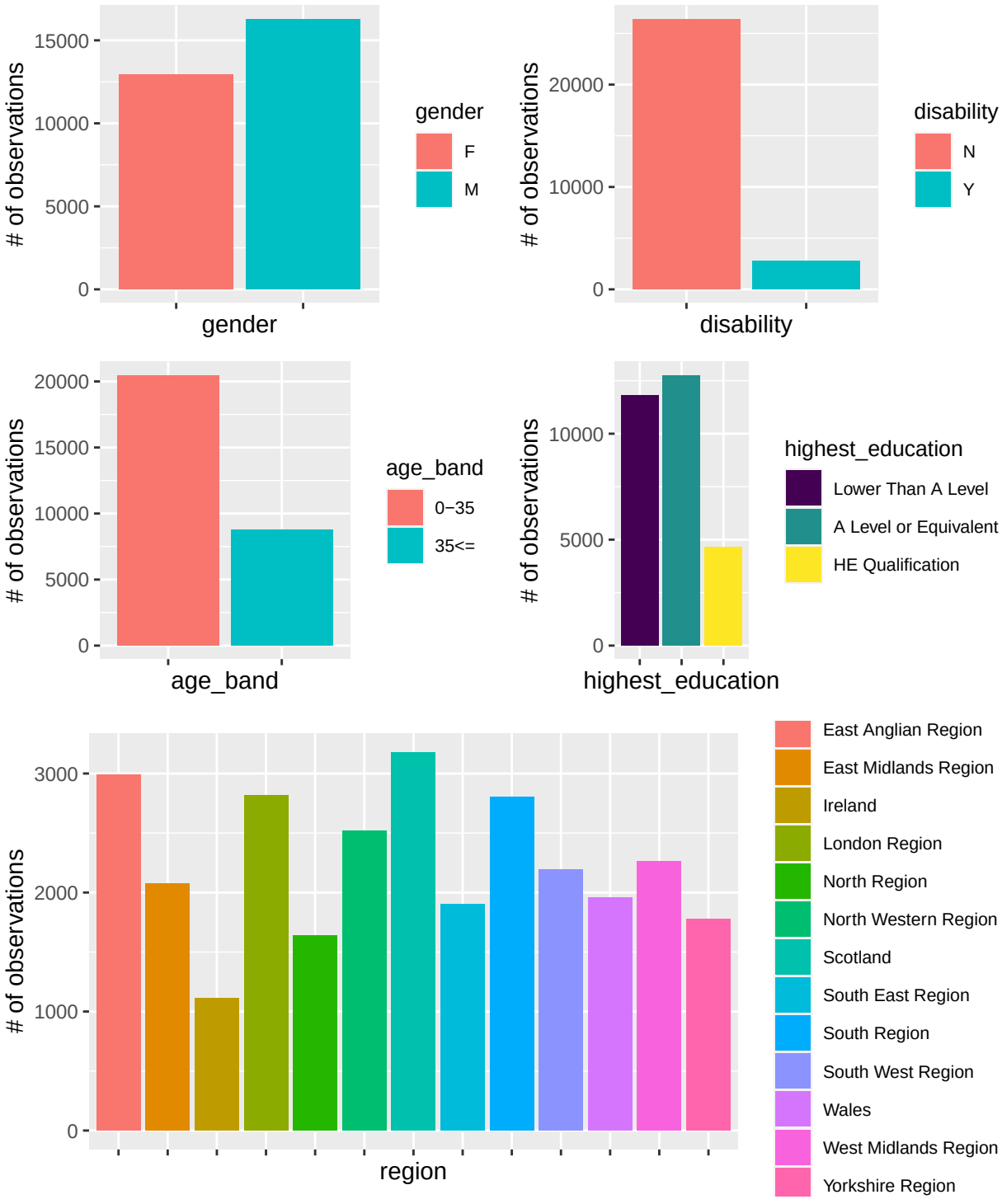
Once the objective table was organized, we wanted to make some visualizations and more profound exploration in order to see if there were any relevant trends or patterns that could guide us in the analysis. We split this section into two: one for the features on their own and another one for the relation between the features and the target variable.

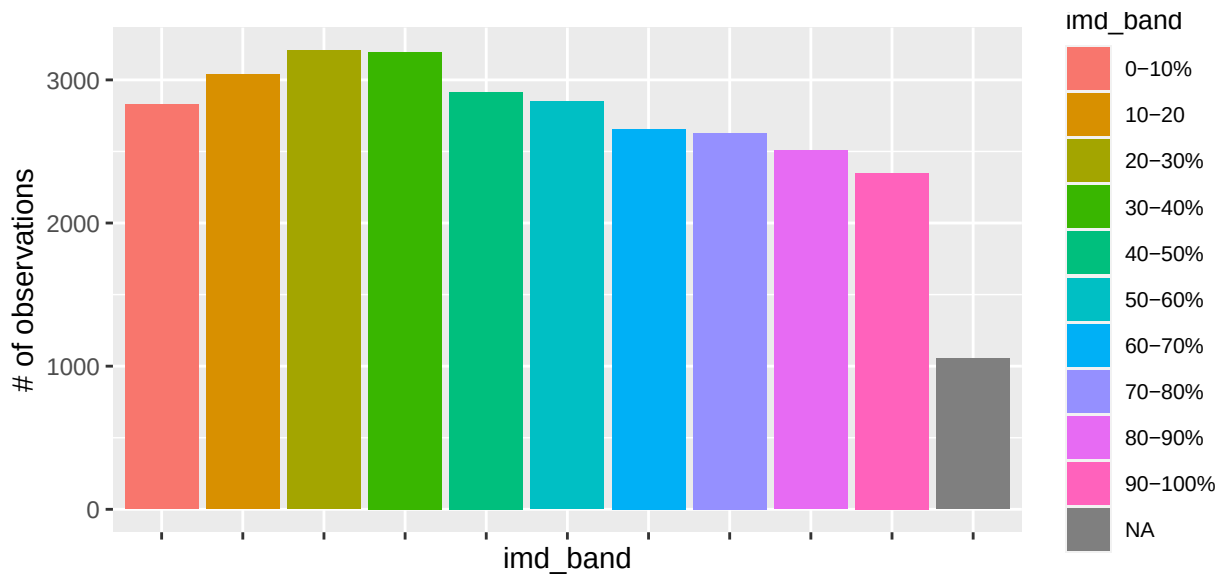
First, before exploring our features in detail, it is important to be fully aware of the balance of the target variable: `final_result`. In our case, after all the procedures we made, the two categories amount almost half of the observations (52.63% cases are 'Successful' and 47.37% cases are 'Unsuccessful'). These are good news in the sense that there is not a strong overrepresentation of a category with respect to the other, so there is not such a high risk of overseeing important relations in our data just because of the size of the two groups:



3.1 Exploration for the features on their own

We were interested in checking the frequency distribution of each of the students' features that are not related with the Virtual Learning Environment, so the `gender`, `disability`, `age_band`, `highest_education`, `region` and `imd_band` columns, which are all originally categorical variables.

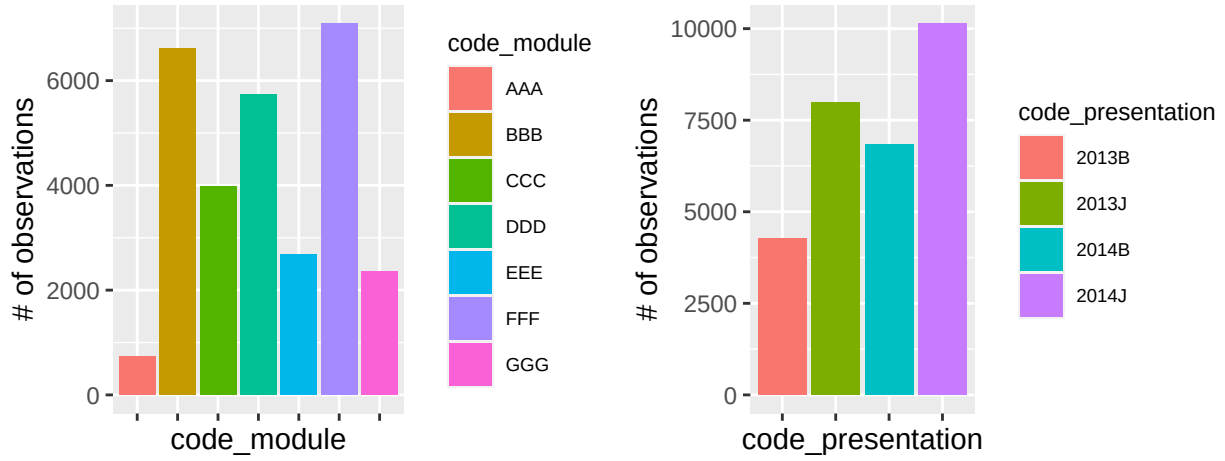

# of observations
15000
10000
5000
0
gender
F
M
gender
# of observations
20000
10000
0
disability
N
Y
disability
# of observations
20000
15000
10000
5000
0
age_band
0−35
35<=
age_band
# of observations
10000
5000
0
highest_education
Lower Than A Level
A Level or Equivalent
HE Qualification
highest_education
# of observations
3000
2000
1000
0
region
East Anglian Region
East Midlands Region
Ireland
London Region
North Region
North Western Region
Scotland
South East Region
South Region
South West Region
Wales
West Midlands Region
Yorkshire Region



From the plots above it can be seen that the majority gender is 'male' with 55.72% of the observations. Also, we notice that the '0-35' `age_band` is the most frequent category (it accounts for 70.04% of the table) and, with respect to the `highest_education` variable, the 'Lower Than A Level' band and the 'A Level or Equivalent' are dominant, since, together, they represent 84.12% of the data. On the other hand, the amount of observations from students who declared a disability (9.64%) is much smaller than the rest of instances.

Also, it is apparent the `region` distribution shows no clear patterns (only Ireland stands out as a region with few observations in comparison to the rest) and that the `imd_band` frequencies seem to concentrate around the mode, which is the '20-30%' band, but we will discuss more about this variable later on.

On the other hand, we can take a look at the features that have to do with the Virtual learning environment. First, still on the realm of the categorical variables, we check how many observations correspond to each course (the `code_module` column) and each period of presentation (the `code_presentation` feature):

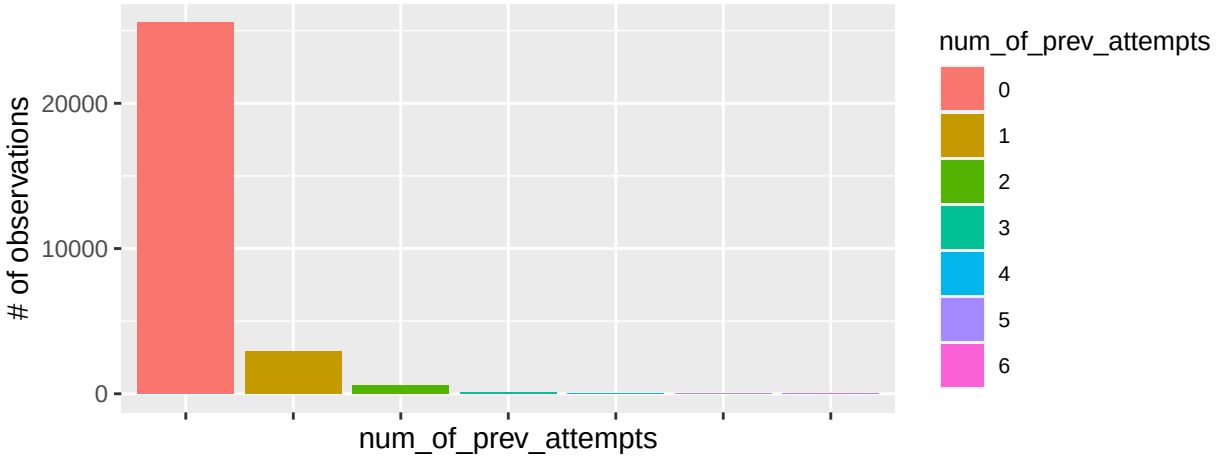


We can see that the module 'AAA' is the one that stands out for having less observations, while in the `code_presentation` column there is not such a phenomenon.

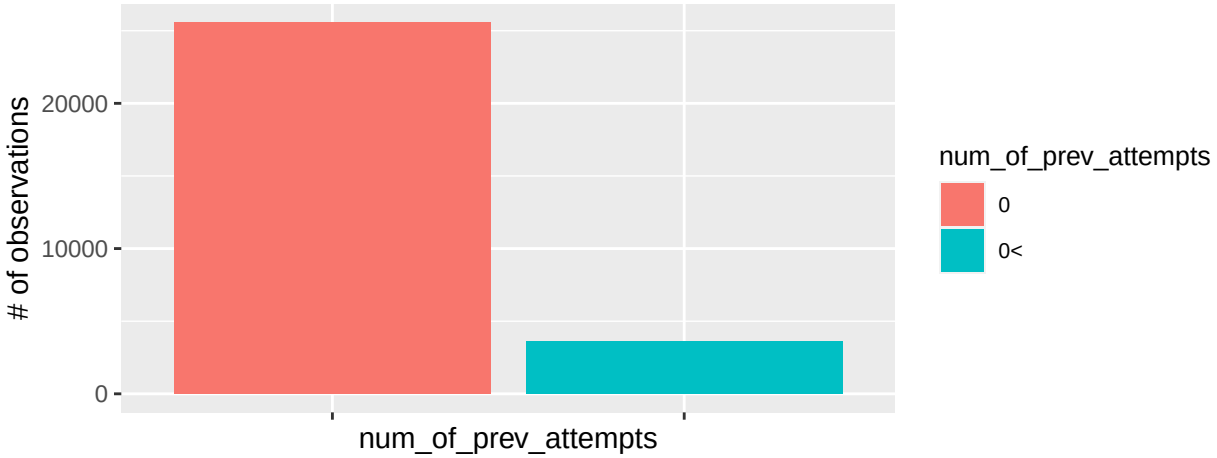
On the other hand, the `studied_credits` and `num_of_prev_attempts` columns originally stored integers, so we were tempted to use them as numeric variables. However, when we tried to visualize these columns as numeric features (with a histogram, a box plot, etc.), the resulting plots were rather uninformative. As a result, we instead considered treating these values as categorical variables and visualizing them as such.

First, let's take a look at the `num_of_prev_attempts` feature. In this case, since the numerical plots were

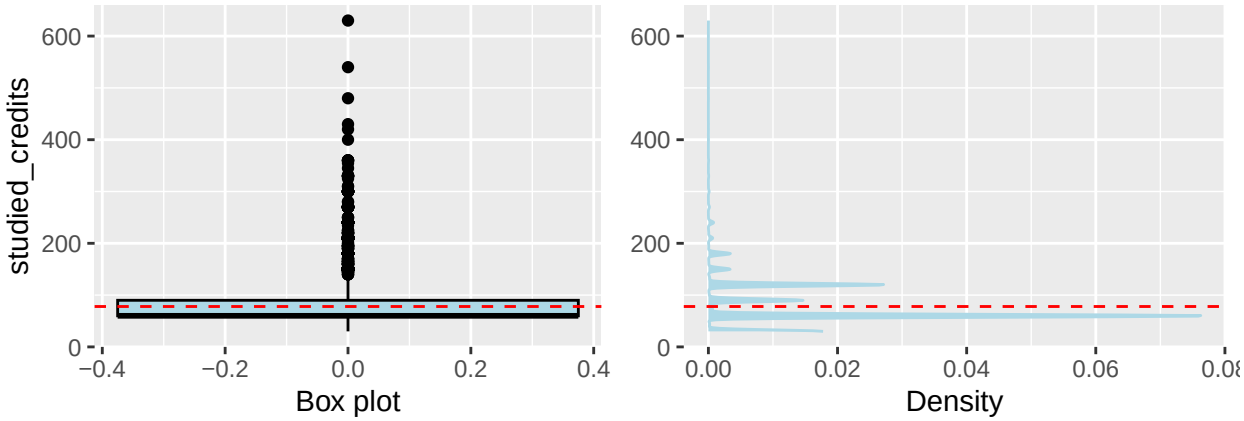
inconclusive, we just converted the feature into a factor column and then plotted it:



This plot allows us to see that most of the values (87.52% of the data) are 0, which corresponds to a first attempt. For this reason, we think that it makes sense to transform the column into two categories: '0' and '0<'. That way, we can assess whether doing more than one attempt has a significant impact on the course results and we end up with a slightly more balanced variable.



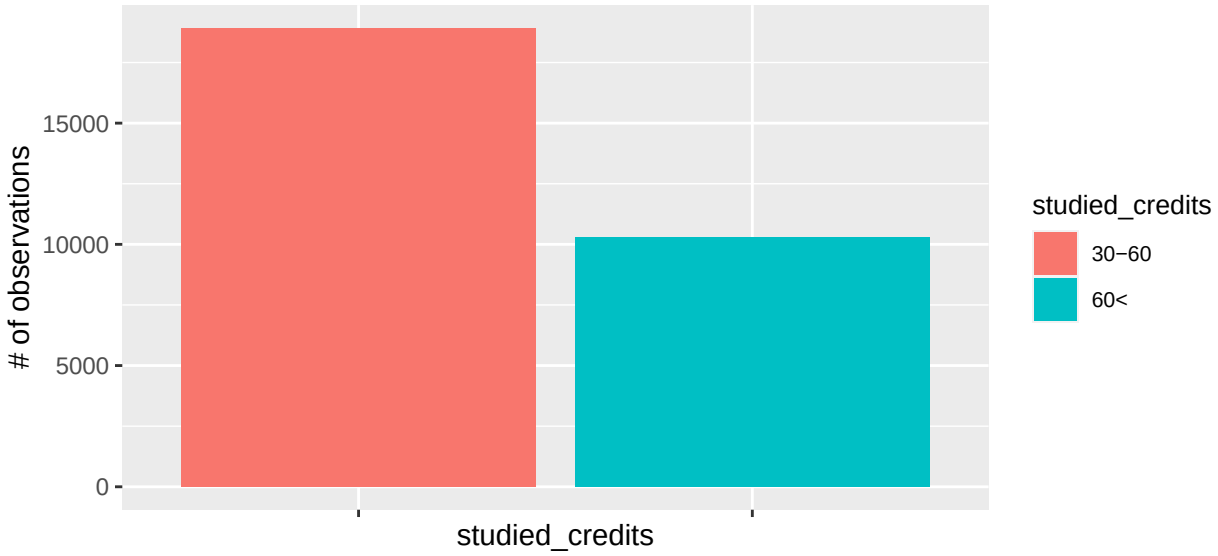
We attempted to do something similar with the the `studied_credits` feature, but in this case we have many more categories than two, so just visualizing them as bar plots may not be as effective. In this scenario, some plots for numerical variables might still help us get some insights:



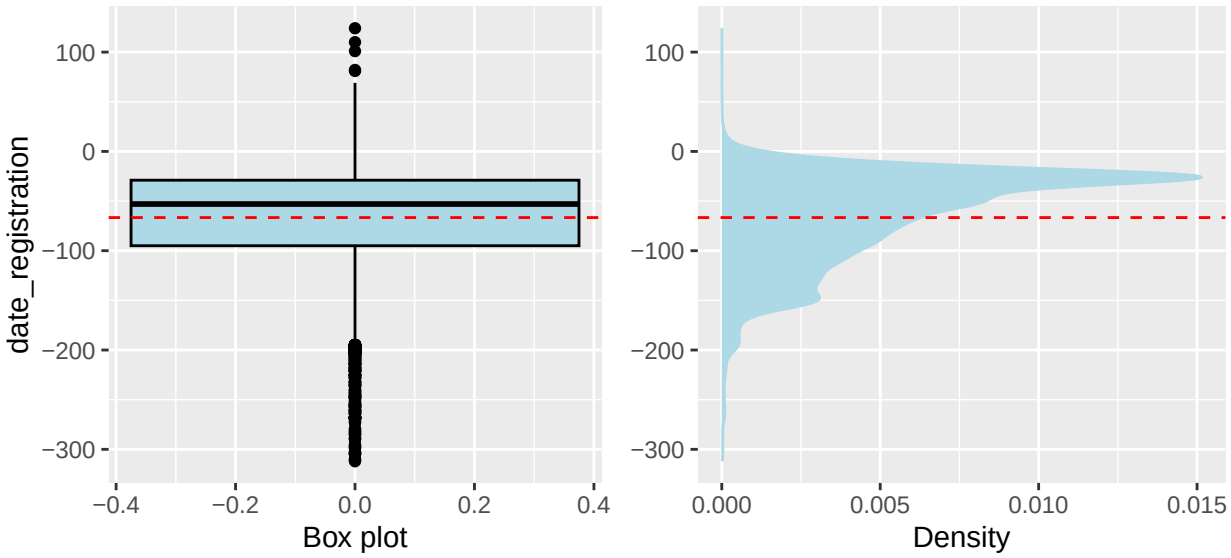
We observe a highly dominant item in the lower part of the distribution. To precisely identify this value, we used a summary of the feature as a factor column (in addition to the summary of the numerical version of the column). With this approach we found that the most frequent value is 60 credits:

studied_credits_fact	studied_credits
60 :15324	Min. : 30.00
120 : 5359	1st Qu.: 60.00
30 : 3539	Median : 60.00
90 : 2849	Mean : 77.99
180 : 661	3rd Qu.: 90.00
150 : 657	Max. :630.00
(Other): 839	NA

These frequencies motivate us to divide the column into two groups: the '30-60' credits band, which accounts for the most frequent category, and the '60<' band, representing the values that deviate from the other group. This grouping allows us to examine if there is a difference in the final course results between taking the usual number of credits or taking a greater amount. Since we only have two groups, we are going to use the feature as an unordered factor column. Again, we end up with a more balanced column:

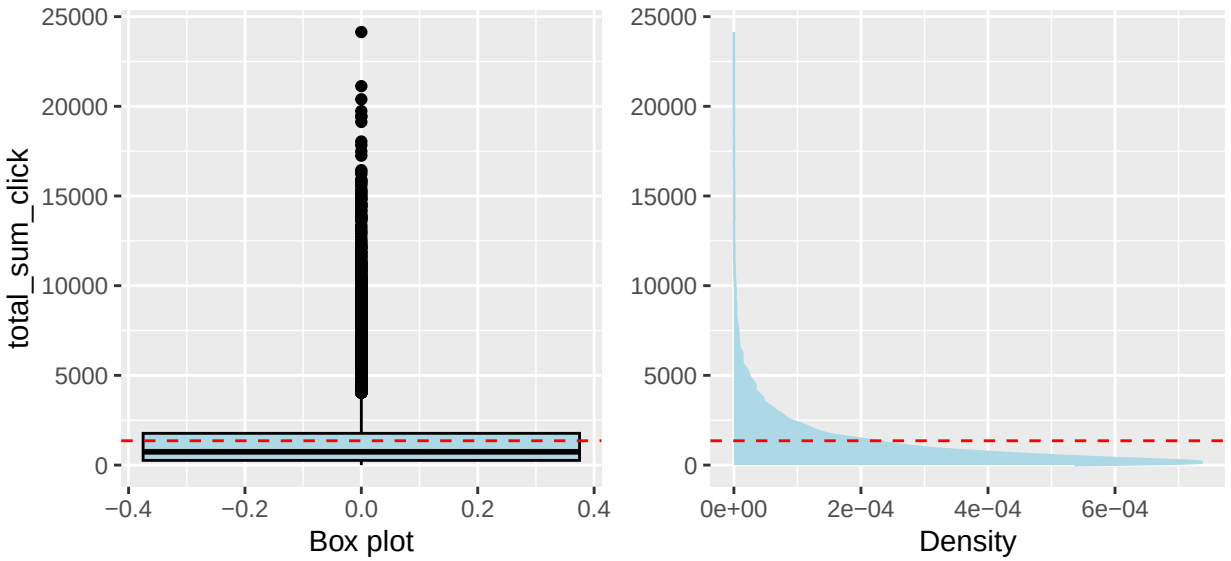


Now, we can explore what the behavior of the "hard" numerical variables is. For this task, it might be useful to visualize a box plot and a density plot for each variable, in order to see if there is any pattern in the distributions. In both graphs, we add a horizontal line to display the value of the mean. First, we look at the `date_registration` variable:

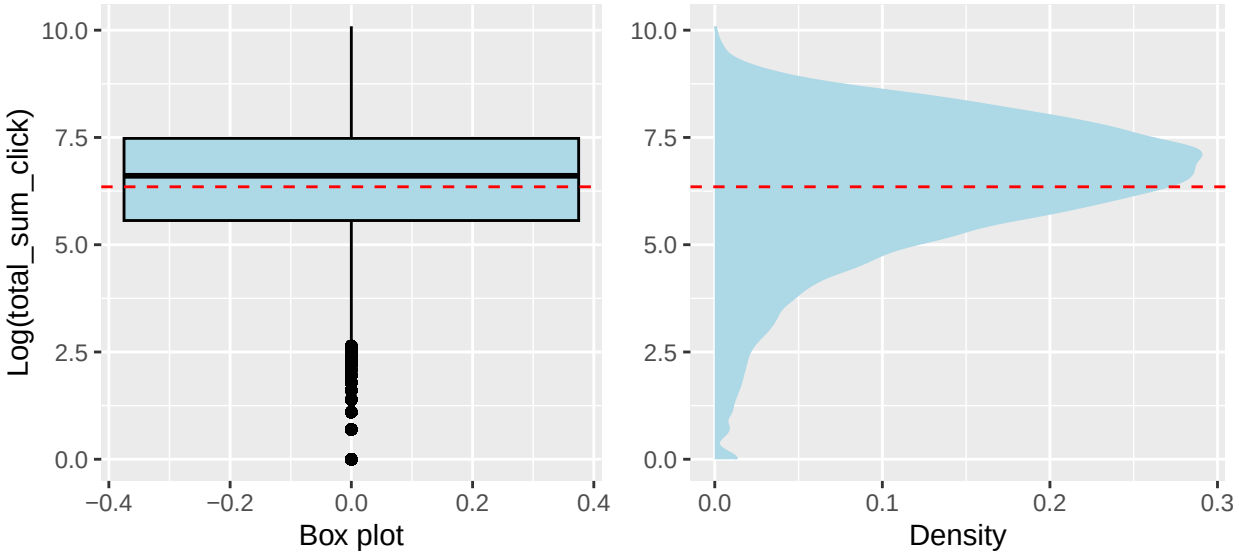


In this case, it seems that most of the students register to the Virtual Learning environment when the date is close to the start of the courses (the 0 value) and very few students register afterwards. Also, we can see that there are some outliers on both sides. We will keep these values because both extreme cases are possible in this context.

On the other hand, as it is shown on the next plots, the `total_sum_click` shows a particular behavior. Firstly, we plotted the variable as it was originally. However, little information can be obtained by looking at these graphs, maybe because of the scale of the variable:



For this reason, we considered applying a log transformation in order to see if this allows us to visualize the distribution more clearly.



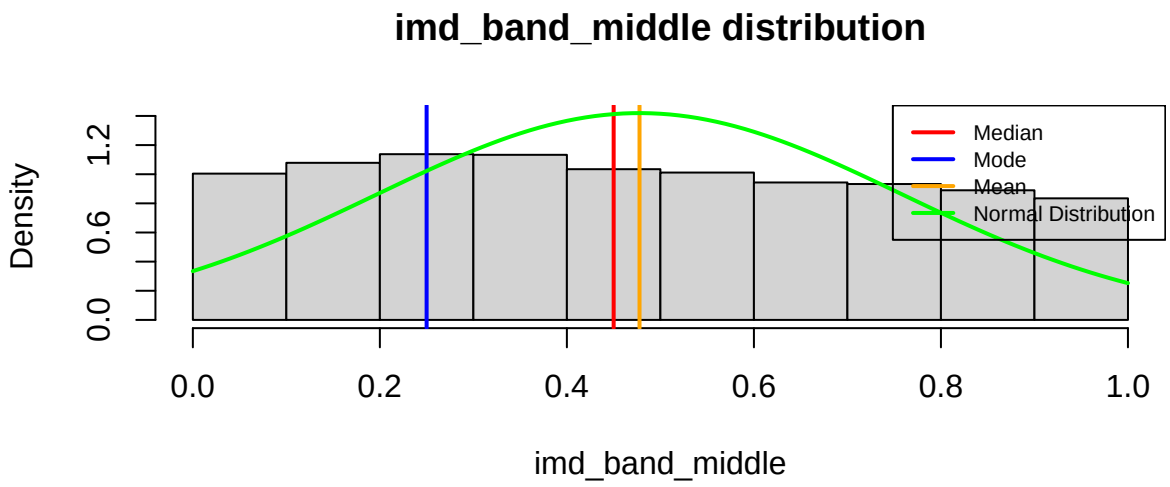
We can see that this approach yields a more 'regular' distribution, as we observe a mean that is close to the distribution's peak and the density plot shows a bell-like shape, though there is still some skewness. In conclusion, applying a log transformation allows us to visualize where the concentration of the interactions are actually happening, and it seems to be close to the mean. This transformation might be useful when we start fitting our models.

At this point, we want to address the relationship among features, to check if there is linear correlation in any case. For this reason, we perform the calculation of the correlation matrix. It is worth mentioning that, since we still have missing values for the `imd_band_middle` variable, we first performed the calculation ignoring the NAs in order to see if there is a linear relationship between this feature and other variables. Also, for these procedures, the values of the `highest_education` column were treated as numbers (keeping the ordering that was previously established and mentioned).

	highest_education	imd_band_middle	total_sum_click	date_registration
highest_education	1.0000000	0.1148029	0.0803362	0.0379516
imd_band_middle	0.1148029	1.0000000	0.0756264	0.0174020
total_sum_click	0.0803362	0.0756264	1.0000000	-0.0424880
date_registration	0.0379516	0.0174020	-0.0424880	1.0000000

From the correlation matrix we can say that there does not appear to be a strong linear relationship among our predictors. Moreover, when considering the `imd_band_middle` feature, which still has missing values, we observe that its highest correlation is with the `highest_education` variable, but the value is only 0.11. Therefore, we will address the missing values situation from a different perspective.

As we saw before, the `imd_band` seems to concentrate around the mode, so now we want to check if its frequencies show some similarity with a normal distribution with a mean equal to the sample mean and a variance equal to the unbiased sample variance:



As we can see, the frequencies do not seem to follow a normal distribution with the mentioned parameters (it is apparent that there is a right-skew). For this reason and given that we already checked if there was any correlation between the `imd_band` column and each of the other variables, we will try a different approach. Since the missing values account only for 3.41% of the observations and that we did not find any specific patterns for the missing data, we are simply going to impute the NAs by randomly selecting one of the bands with a probability equal to the proportion that each category has within the non-missing observations. We do this by selecting all the non-NA observations and randomly picking (with uniform probability) one value for each of the NA observations.

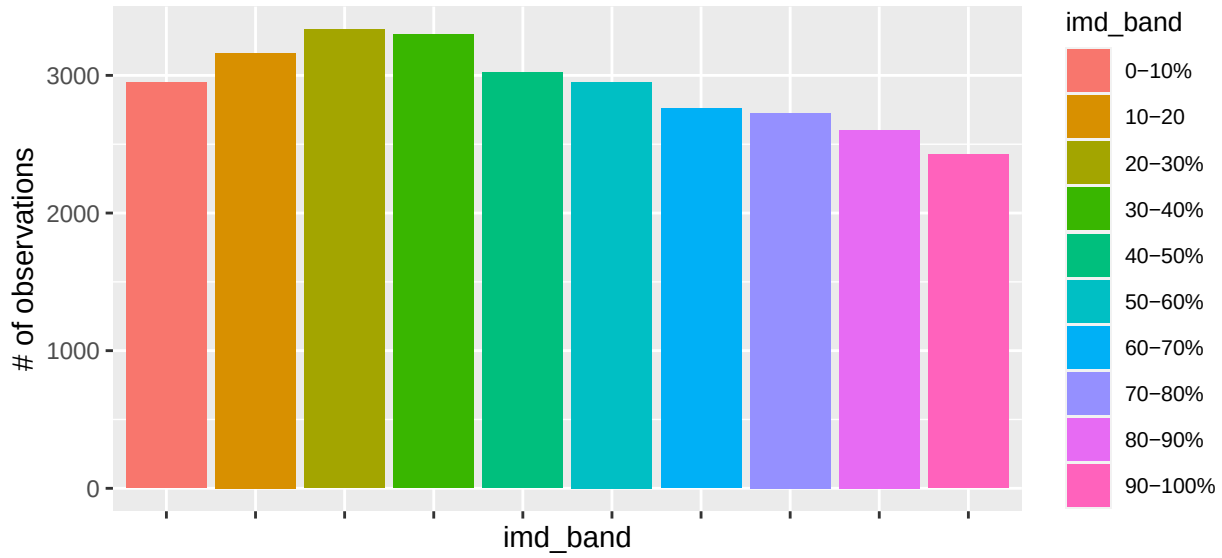
```
# Setting the seed for reproducibility
set.seed(123)

# Creation of a subset of non-missing observations
samples = courseResults[!is.na(courseResults$imd_band), 'imd_band']

# Imputation of missing values
missing_indices = which(is.na(courseResults$imd_band))
imputed_values = sample(samples, length(missing_indices), replace = TRUE)

# Replacement of the missing values with the imputed values
courseResults$imd_band[missing_indices] = imputed_values
```

Then, the `imd_band_middle` was updated with the new imputed values (computing the middle point of the band, as before). We now check how the old distribution did not suffer any distortions:



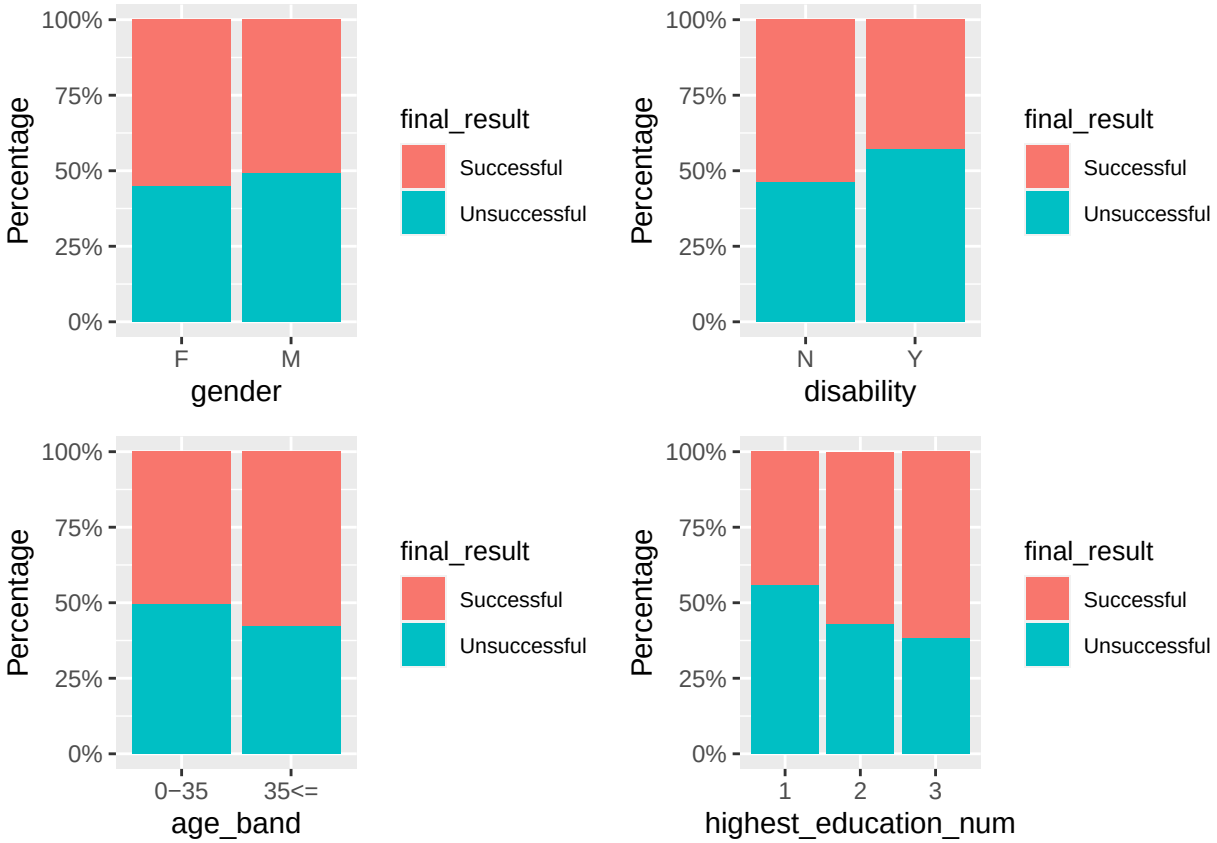
Finally, using the complete table (including the imputed values) we update the correlation matrix, which barely changed:

	highest_education	imd_band_middle	total_sum_click	date_registration
highest_education	1.0000000	0.1093778	0.0805466	0.0410464
imd_band_middle	0.1093778	1.0000000	0.0691021	0.0154988
total_sum_click	0.0805466	0.0691021	1.0000000	-0.0361575
date_registration	0.0410464	0.0154988	-0.0361575	1.0000000

3.2 Exploration of the features with respect to the response variable

In this section, we want to visualize the relationship between our target variable `course_results` and each of the features. These graphical aid might be helpful for the modelling stage.

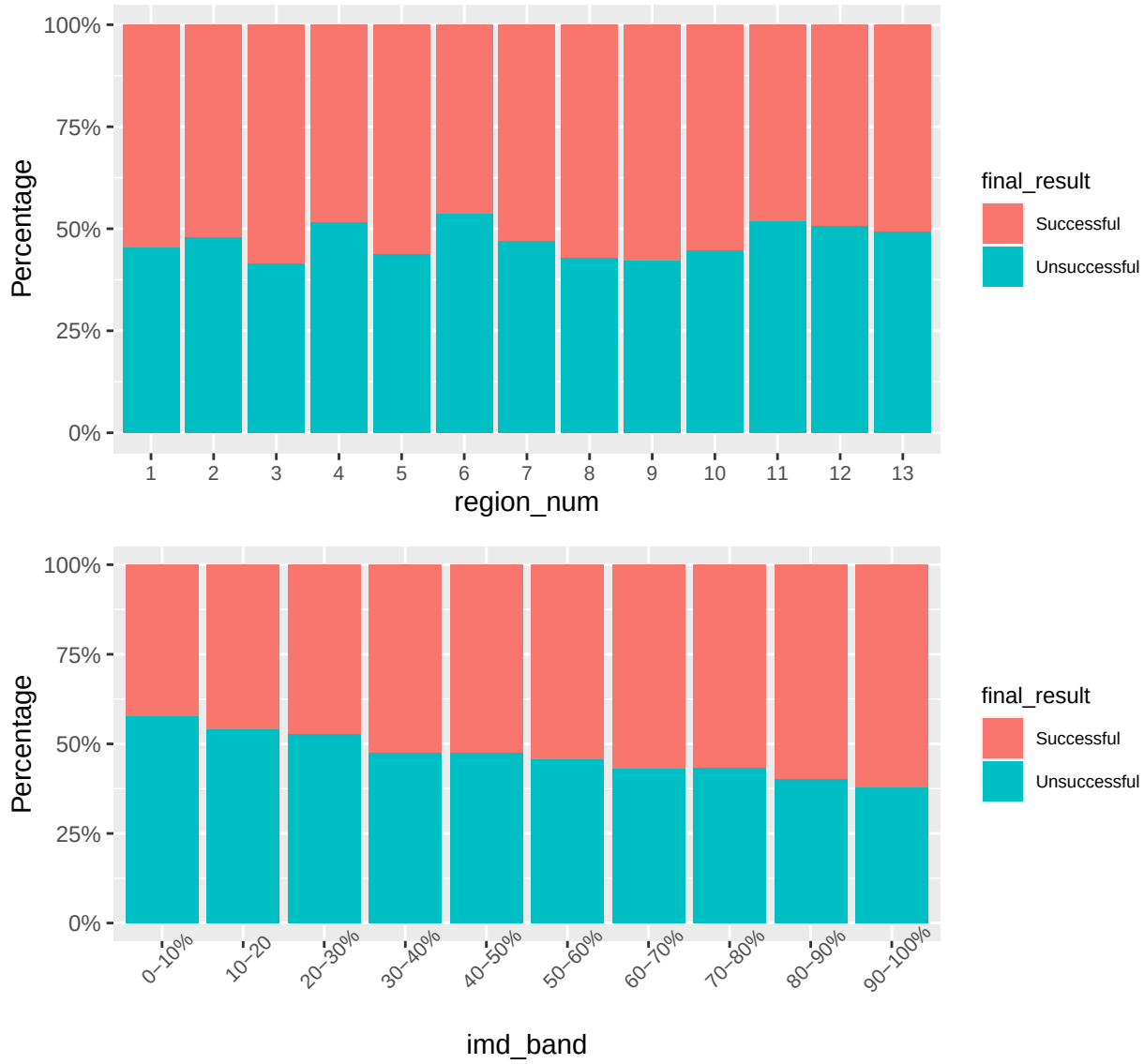
First we will tackle the categorical variables. In this case we want to check if within each category there are differences in the proportion of successful examples. We try to keep the same order of the previous section, so we start with the `gender`, `disability`, `age_band` and `highest_education` columns. It is worth mentioning that, since the labels of the `highest_education` feature are too long to be displayed in the axis, for the plot we used a temporal column `highest_education_num` in which '1','2' and '3' refer to 'Lower Than A Level', 'A Level or Equivalent' and 'HE Qualification', respectively:



We observe that, among the features that only have two categories, the one that shows greater disparity in proportions is `disability`, followed by `age_band`. The group that reported disabilities has a success rate that is 11% lower compared to the other group, while for the age_band feature, the difference is approximately 7%. In contrast, there appears to be little variation between the genders. On the other hand, there seems to be a increasing relationship between the level of education and the rate of success.

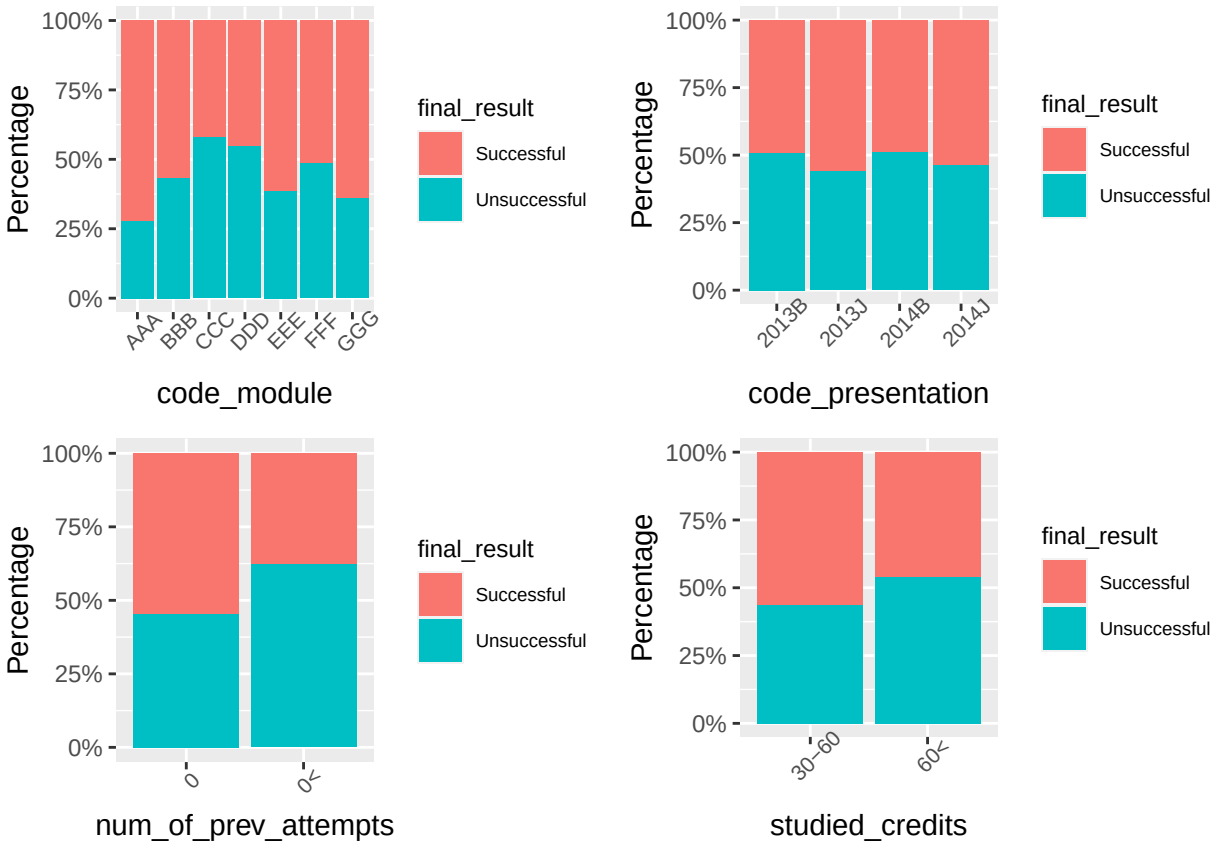
Also, in line with our approach for the highest_education column, we also assigned a number to each region in order to create the plot. The table displaying this mapping, along with the plots for region and imd_band, is presented below:

region	region_num
East Anglian Region	1
East Midlands Region	2
Ireland	3
London Region	4
North Region	5
North Western Region	6
Scotland	7
South East Region	8
South Region	9
South West Region	10
Wales	11
West Midlands Region	12
Yorkshire Region	13



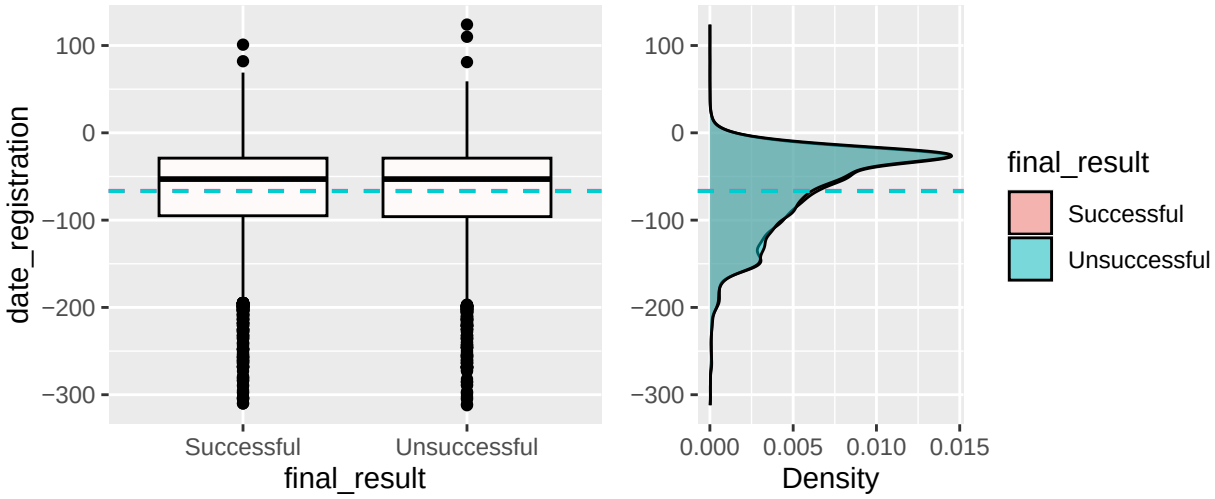
For the `region` feature we do not observe any clear pattern, while for the `imd_band` there seems to be an increase of the success rate as the value of the IMD band increases. We found this interesting, so took a look at the article in which the creators of the dataset describe the information and they refer to an official publication related to the Index of Multiple Deprivation band. According to some charts and descriptions in that document, apparently, the `imd_band` should be interpreted as follows: the '0-10%' band refers to the 10% *most* deprived of areas and the '90-100%' band corresponds to the 10% *least* deprived of areas. According to this, as the deprivation of the area corresponding to each student increases, the rate of success also increases.

Now we look at the last categorical variables (taking into account the updates we have made): `code_module` and `code_presentation`, `num_of_prev_attempts` and `studied_credits`:

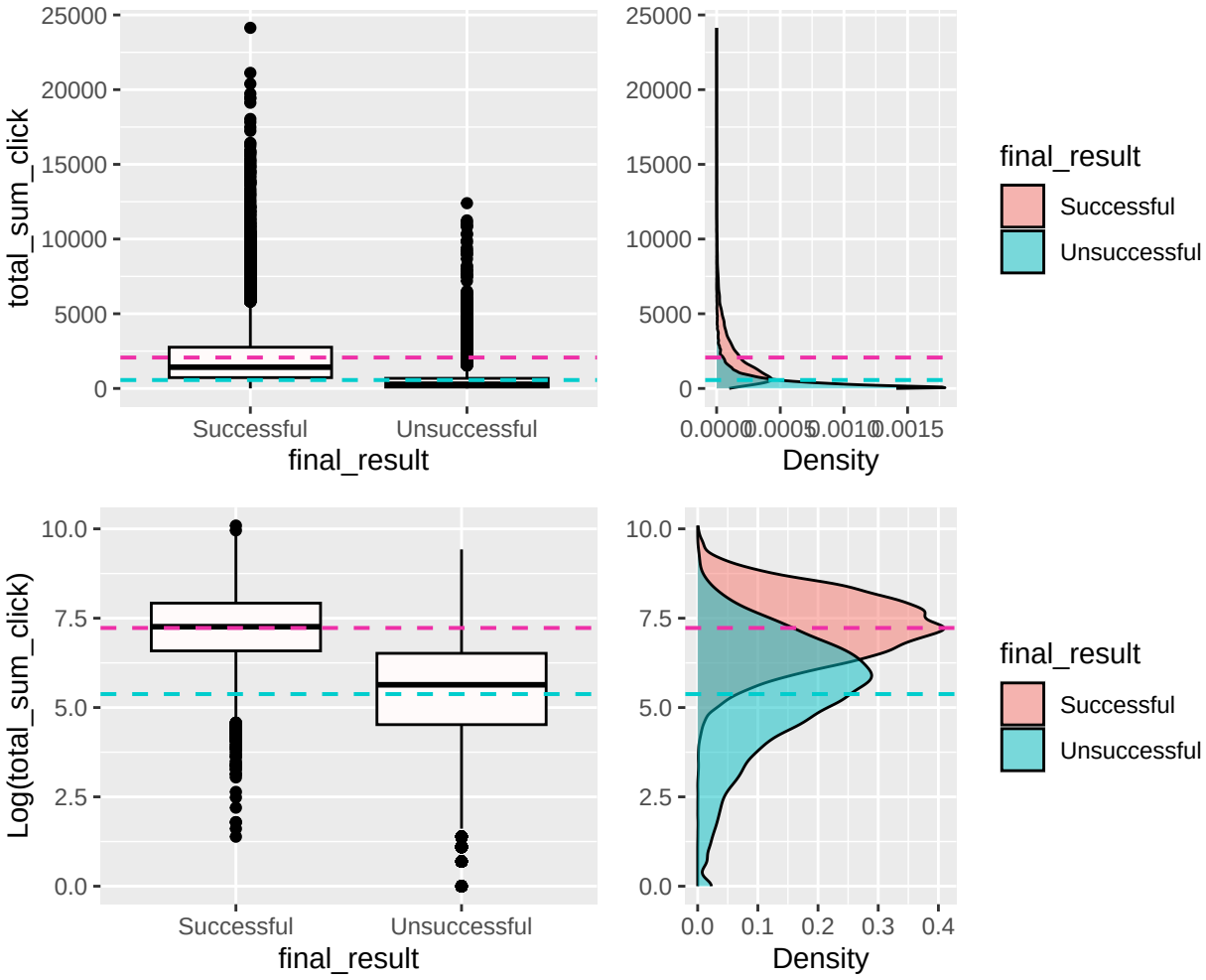


From the first two plots we cannot spot an clear pattern. We just observe that the module 'AAA' is the one that has a higher success rate, which coincides with the category that has less observations, so this phenomenon might have to do with the size of the group, but it must be assessed with care. Also, regarding the period of presentation (`code_presentation`), we observe that the periods marked with a 'J' (i.e. the presentations starting in October) tend to have a better success rate. Additionally, With respect to the columns with two groups, it seems that attempting the courses for the first time and taking the lower - and more usual- amount of credits is related with higher success rates (the differences with respect to the contrasting groups are around 17% and 10% respectively).

Next, we explore the relationship between the `final_result` and the two numerical variables that we have left: `date_registration` and `total_sum_click`. We start with the `date_registration` feature, which does not show a big difference between the registration dates of the two groups ('Successful and 'Unsuccesful'). Our approach is to use a box plot and a density plot, along with the horizontal lines for the mean of each group. As you can see below, the two distributions practically overlay each other.



On the contrary, these two groups appear to have very different behaviors regarding the `total_sum_click` column. Again, we plot the raw values and the log transformation:

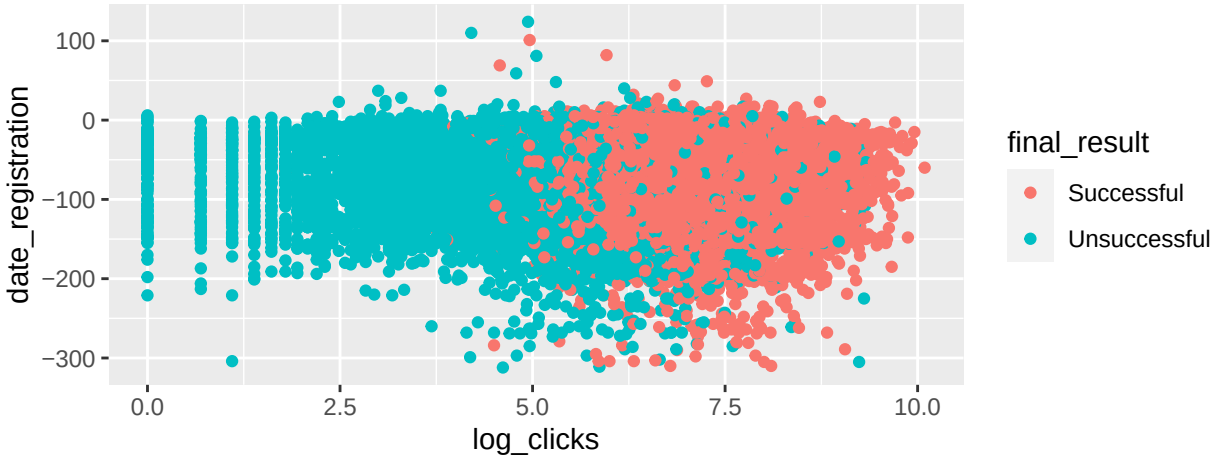


Firstly, it is worth noting that the fact that there are noticeable differences between the two groups suggests that this feature is useful for classifying our data. Also, as it happened when we plotted the `total_sum_click` by its own, the log transformation adds interpretability to the visualization and lets us

better distinguish where the concentration of the data is ocurring. In this case, for example, apart from the fact that the 'Successful' group tends to do more clicks, we notice that it has a less spread distribution in comparison to the 'Unsuccessful' one, so it has less variability. Additionally, the 'Successful' group does not have the skew that the other group and the general distribution (the one without the separation by groups) have.

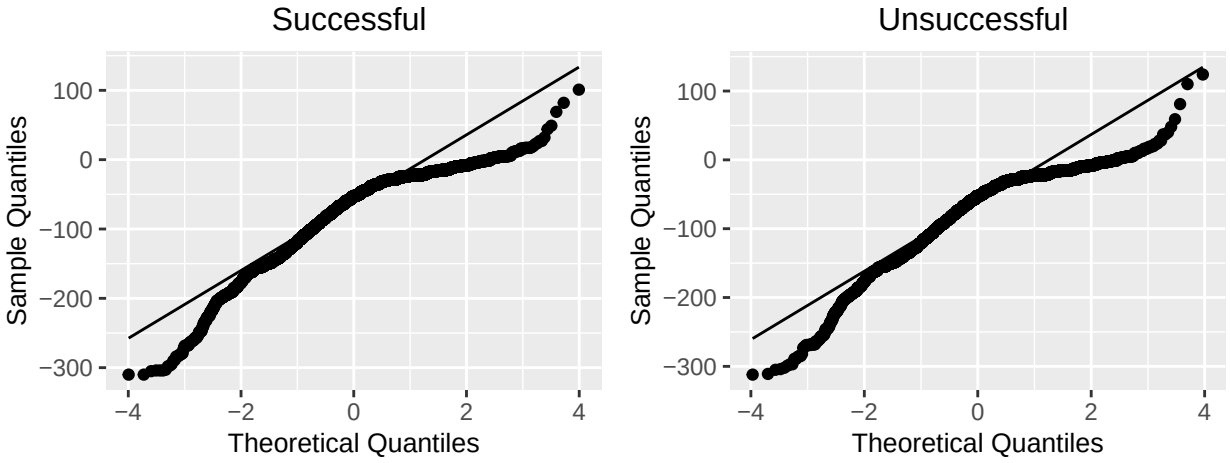
Taking into account this information, when we proceed to fit the statistical models, we might check which version of the `total_sum_click` fits the model best: the raw one or the transformed one.

Just as an exercise to verify if the joint distribution visualization of our 'hard' numerical variables would give us any insight, we also plotted the log transformation of the clicks in the X axis and the registration date in the Y axis with respect to the labels of the two groups. Consistently to what we saw in the previous plots, it is apparent that, out of these two features, only the amount of interactions gives us hints to distinguish between the 'Successful' and 'Unsuccessful' examples.

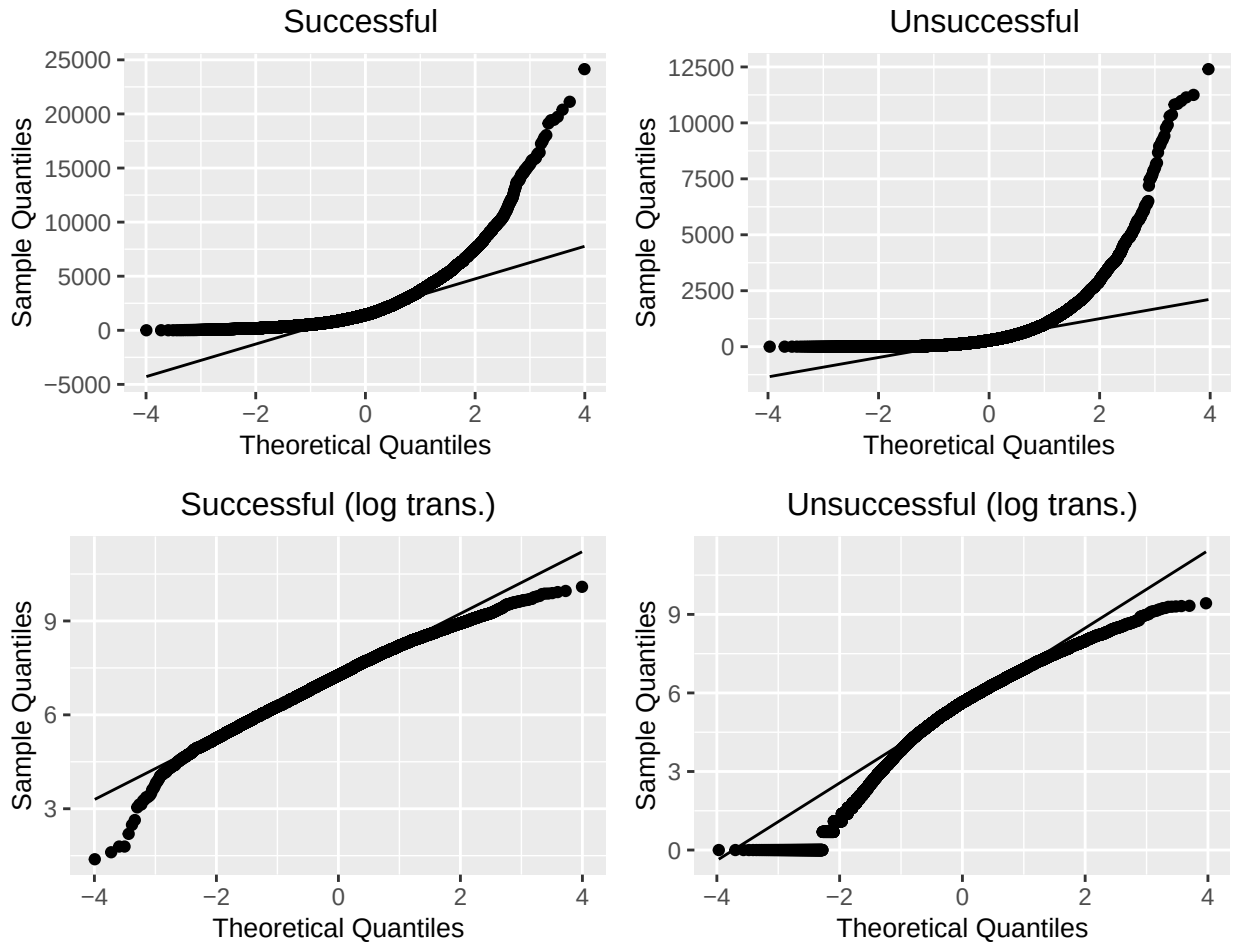


Finally, we just wanted to go deeper on the distributions of the numerical features when we partition them by the `final_result` labels ('Successful' and 'Unsuccessful'). More precisely, we wanted to check if there are some hints of normality. This information might be useful to understand the performance of the models that assume normality of the predictors within each outcome label, such as the Linear Discriminant Analysis and the Quadratic Discriminant analysis.

We drew a QQ normality plot for the `date_registration` feature (partitioning by 'Successful' and 'Unsuccessful', the labels of `final_result`) and we saw that its distribution deviates from the normal distribution in both tails of the two plots, as it can be seen in the graphs below.



In addition, we plotted the corresponding QQ normality plots for the `total_sum_click` variable. As usual, we do it for the raw version and the log transformation:



Once more, we observe that the most distorted plot (for both outcome labels) corresponds to the raw version of the `total_sum_click` feature. Moreover, we evidence that the log transformation smooths the distribution and seems to do a better job at approaching the normal distribution, even though there is still some deviation. Particularly, we see that the 'Unsuccessful' group has a long left tail and seems to deviate more from the normal distribution in comparison to the 'Successful' group. This goes in line with what we observed in the density plots.

As a conclusion to this last part, we think that these deviations from the normal distribution might add noise to the models that assume normality. Also, this exploration gave us an insight as the QQ plots appear to suggest that transforming the `total_sum_click` column with the logarithm might give us a better predictor. Especially for the LDA and QDA models. Also, the different variability of the two groups is something to consider when we interpret the results of the LDA (which assumes a shared variance among the response classes).

4. Statistical models

4.1 General considerations

At this point, we can start fitting our models in order see more in depth the relation between our target variable, `final_result`, and the features we have available. We will do this by different means. More pre-

cisely, we are going use Logistic Regression (and then a Logistic Classifier related to it), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Naive Bayes and K-Nearest Neighbors. In the end, we are going to discuss the outcome of each model.

There are some general procedures that we are going to implement in order to maintain certain uniformity across all the models.

First, in order to interpret the results correctly, we check the current contrasts (the coding R assigned) of the `final_result` column:

	Unsuccessful
Successful	0
Unsuccessful	1

We observe that R had coded 'Successful' with 0 and 'Unsuccessful' with 1. Since it is more common to interpret the probabilities and the outcomes the other way around (with 1 being a success), we can modify this by converting the 'final_result' column into an ordered factor:

```
courseResults$final_result = factor(courseResults$final_result,
                                    levels = c('Unsuccessful','Successful'),
                                    ordered = TRUE)
```

Next, before we start fitting models, we will set aside a test set for assessing the performance of the models obtained by using each method. This test set will remain "unobserved" during the modeling process and will only be used for the final evaluation. For this purpose, we are going to use a 80%/20% split: 20% of the data is randomly selected for the test set and the rest of observations will be used for training.

```
# We remove the columns that are not useful for the models
# (imd_band was replaced by the numeric imd_band_middle)
courseResults = subset(courseResults, select = -c(id_module_presentation_student,imd_band))

# We use a seed for reproducibility and split the data
set.seed(123)

train_indices = createDataPartition(courseResults$final_result, p = 0.8, list = FALSE)
train_courseResults = courseResults[train_indices, ]
test_courseResults = courseResults[-train_indices, ]
```

Next, we we would like to select the "more appropriate version" of two of our features: `total_sum_click` and `highest_education`, but for this, we can start using some statistical tests.

4.2 Logistic regression and logistic classifier

4.2.1 Fitting the logistic regression models

4.2.1.1 Generalities and fitting the stepwise feature selection model for the logistic regression

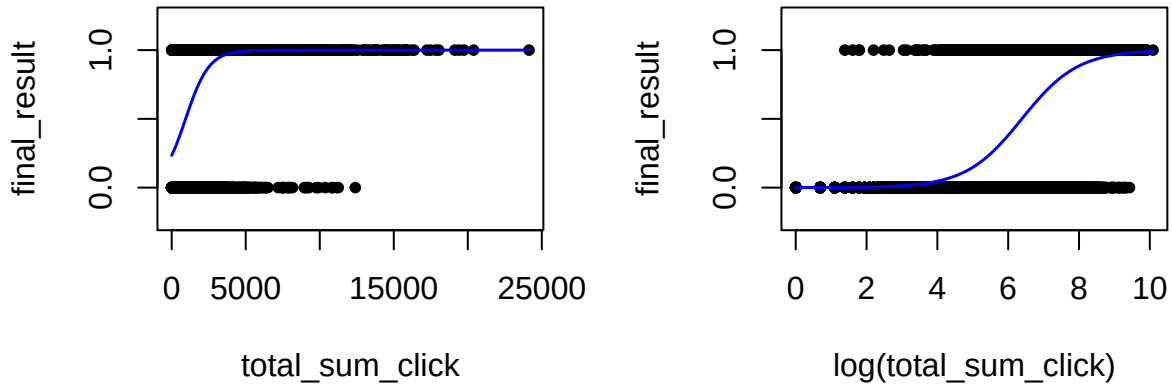
With respect to the `total_sum_click` feature, we have considered two versions of the column: the raw values and the values with a log transformation. For the sake of understanding this feature in isolation, we investigated what happens if we fit a logistic regression with respect to the two versions this predictor only. First, we see how the deviance of the model behaves with each version:

```
lr_rawClicks = glm(final_result ~ total_sum_click, data=train_courseResults,
                   family = binomial)
lr_logClicks = glm(final_result ~ log(total_sum_click), data=train_courseResults,
                   family = binomial)

kable(generate_summary_table(models=list(lr_rawClicks,lr_logClicks),
                        model_names = c('lr_rawClicks','lr_logClicks'),
                        include_b1 = TRUE))
```

Model	Null_deviance	Residual_deviance	AIC	BIC	Beta1	P_value_Beta1
lr_rawClicks	32351.1	25057.03	25061.03	25077.15	0.0012617	0
lr_logClicks	32351.1	22294.44	22298.44	22314.56	1.2646283	0

The deviance of the log transformation is lower than the one of the raw version, so this suggests that the modified version is a better predictor. This goes in line with what we observed during the exploratory data analysis, where for scale reasons the log transformation seemed to represent the feature in a better way. Moreover, we would like to see how this looks graphically:



We observe that using the raw version of the column for a logistic regression model makes it look truncated, while the log transformation plot looks smoother and seems to represent the probabilities across all the range of the x values. For these reasons and in line with what we have seen before, even from the exploratory data analysis, **we will stick with the log transformation of the `total_sum_click` column from now on**.

```
#We also update the column on the test for the evaluations at the end
train_courseResults$total_sum_click = log(train_courseResults$total_sum_click)
test_courseResults$total_sum_click = log(test_courseResults$total_sum_click)

# We rename the column for clarity
colnames(train_courseResults)[colnames(train_courseResults)=='total_sum_click']='log_clicks'
colnames(test_courseResults)[colnames(test_courseResults)=='total_sum_click']='log_clicks'
```

Alternatively, we want to check the most appropriate version of the `highest_education` column. This feature could be interpreted in two ways. Right now, it is an ordered factor column, but this means that

we can still consider it a categorical predictor or, instead, we could use it as a numerical predictor for the regressions.

In the first case, when fitting the models, each of the categories (except one that is taken as a reference) is coded as a new dummy variable with a 1 if the observation belongs to the category or a 0 otherwise. With that setting, the resulting regression coefficients represent the expected change on the response variable with respect to the reference category. In the second case, we would assign an integer to each category (taking into account the order they have) and then treat it normally as a number in the models.

For this, we will do a similar procedure to the one we just presented, so we fit two logistic regression models: one with the column as a categorical feature and another one with the predictor as a number.

```
train_courseResults$highest_education_unordered=factor(train_courseResults$highest_education,
                                                       ordered = FALSE)
train_courseResults$highest_education_unordered=relevel(
                                          train_courseResults$highest_education_unordered,
                                          ref = 'Lower Than A Level')
lr_educationFact=glm(final_result ~ highest_education_unordered, data=train_courseResults,
                     family = binomial)
lr_educationNum=glm(final_result ~ as.numeric(highest_education), data=train_courseResults,
                    family = binomial)
```

In this case, we show the full summary for the regression. We start with the version involving `highest_education` as a categorical column:

```
##
## Call:
## glm(formula = final_result ~ highest_education_unordered, family = binomial,
##     data = train_courseResults)
##
## Coefficients:
##                                                 Estimate Std. Error z value
## (Intercept)                                    -0.23958    0.02075  -11.55
## highest_education_unorderedA Level or Equivalent  0.53289    0.02879   18.51
## highest_education_unorderedHE Qualification       0.71319    0.03968   17.97
##                                                 Pr(>|z|)
## (Intercept)                                       <2e-16 ***
## highest_education_unorderedA Level or Equivalent   <2e-16 ***
## highest_education_unorderedHE Qualification        <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 31862  on 23380  degrees of freedom
## AIC: 31868
##
## Number of Fisher Scoring iterations: 4
```

It is worth recalling that the three categories for the `highest_education` feature are 'Lower Than A Level', 'A Level or Equivalent' and 'HE Qualification', in that order. Since we chose 'Lower Than A Level' as a reference, what the coefficients are showing us is that the other two categories are related to better expected odds of success (the sign is positive) by a factor of $\exp(\beta_j)$. Also, we observe that β_j is higher as the level

of education is also higher, so, indeed, there seems to be an increasing relation between the odds of success and the level of education. This is coherent to what we saw during the data exploration. Next, we check what the logistic regression yields for `highest_education` as a numerical predictor:

```
##
## Call:
## glm(formula = final_result ~ as.numeric(highest_education), family = binomial,
##     data = train_courseResults)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -0.58926    0.03548  -16.61   <2e-16 ***
## as.numeric(highest_education)  0.39733    0.01893   20.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 31901  on 23381  degrees of freedom
## AIC: 31905
##
## Number of Fisher Scoring iterations: 4
```

In this case we also observe the positive correlation between the odds of success and the level of education, captured by the β_1 coefficient. The difference for us is that this kind of model is easier to interpret since it is just one coefficient that accounts for the upward trend that the odds of success show as the level of education increases. Nevertheless, we must mention that both the residual deviance and the Akaike Information Criterion (AIC) are greater in the model with `highest_education` as numbers, so the fit is a little worse. However, since the difference is very small and we are just doing an exploratory evaluation with just one feature, we will disregard this measures for now in order to privilege interpretation and avoid adding unnecessary features to the model (`highest_education` as numbers already provides the information we need and adding the dummy variables is less natural to interpret). For this reason, **we will use `highest_education` as a numerical feature from now on.**

```
#We also update the column on the test for the evaluations at the end
train_courseResults$highest_education = as.numeric(train_courseResults$highest_education)
test_courseResults$highest_education = as.numeric(test_courseResults$highest_education)

# We remove the temporal column we created for the feature as an unordered factor
train_courseResults = subset(train_courseResults, select=-c(highest_education_unordered))
```

At this point, we can start fitting our models with multiple predictors. For computational reasons, because of the amount of predictors that we have and the size of our data, it was unfeasible to implement a best subset selection algorithm. However, taking this into consideration, we will do a first try at running our model with a stepwise feature selection.

For this task, we use the 'step' function with the `direction` parameter set at 'both' and a penalty term of log(n), which corresponds to the Bayesian information criterion (BIC). This way we obtain the best model function taking into account the backward and the forward elimination procedures. It is worth mentioning that we set the selection criterion to the BIC because we still have some categorical features with many labels, so, for the sake of interpretability, we want to know which categories are actually significant for our classification task. A more restrictive feature selection criterion, like the one that BIC provides us, works better in this scenario:

```r
n = dim(train_courseResults)[1]
lr_fullModel = glm(final_result ~ ., data=train_courseResults, family = binomial)
lr_step_bic = step(lr_fullModel, direction="both", k=log(n), trace=0, steps=1000)
summary(lr_step_bic)
```

```
##
## Call:
## glm(formula = final_result ~ code_module + code_presentation +
##     gender + highest_education + imd_band_middle + age_band +
##     disability + log_clicks, family = binomial, data = train_courseResults)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -13.48031    0.23696 -56.888  < 2e-16 ***
## code_moduleBBB           1.21607    0.13178   9.228  < 2e-16 ***
## code_moduleCCC          -1.17246    0.13338  -8.790  < 2e-16 ***
## code_moduleDDD          -0.54534    0.12846  -4.245 2.19e-05 ***
## code_moduleEEE          -0.74549    0.13732  -5.429 5.67e-08 ***
## code_moduleFFF          -2.12848    0.13148 -16.189  < 2e-16 ***
## code_moduleGGG           1.78777    0.14266  12.532  < 2e-16 ***
## code_presentation2013J   0.57130    0.05964   9.580  < 2e-16 ***
## code_presentation2014B   0.56865    0.06300   9.027  < 2e-16 ***
## code_presentation2014J   0.67456    0.05985  11.270  < 2e-16 ***
## genderM                  0.27247    0.04611   5.909 3.45e-09 ***
## highest_education        0.42838    0.02733  15.674  < 2e-16 ***
## imd_band_middle          0.60996    0.06683   9.127  < 2e-16 ***
## age_band35<=            -0.52618    0.04205 -12.512  < 2e-16 ***
## disabilityY             -0.28701    0.06268  -4.579 4.68e-06 ***
## log_clicks               1.90649    0.02654  71.834  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 18395  on 23367  degrees of freedom
## AIC: 18427
##
## Number of Fisher Scoring iterations: 6
```

As we can see, many of the predictors seem to have an effect on the odds of being successful at the courses, but we would like to be more precise. In particular, it catches our attention that the categories of the `code_module` and the `code_presentation` features are all significant, but their coefficients are not coherent with what we observed in the exploratory data analysis.

In the first place, recalling that the four possible categories of the `code_presentation` are '2013B', '2013J', '2014B' and '2014J', what we saw in the visualizations was that both groups marked with a 'J' (the courses starting in October) had greater success rates in comparison to the groups marked with a 'B' (the modules starting in February). However, there was not a clear pattern among groups starting in the same month. Unfortunately, the coefficients of the regression we just fitted are not really showing this trend, since all of the categories seem to have an increase in the odds of 'Success', but it is not possible to spot the difference between the October and February groups.

Moreover, regarding the `code_module` column, we did not spot any clear trends in the rates of success among the different courses, only that it was noticeable that the 'AAA' module had a greater success rate than the

other groups. This is also not appreciated when looking at the regression coefficients. In some cases, the coefficients even suggest that the odds of success are better in other groups.

For these reasons we believe that there might be a confounding effect by adding all the categories along with the rest of predictors when running the regression. Because of this, again, we would like to fit two models with the isolated features in order to see if we can get some predictors that are easier to interpret. First, we do it with the `code_presentation` feature:

```
##
## Call:
## glm(formula = final_result ~ code_presentation, family = binomial,
##     data = train_courseResults)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -0.03229    0.03427  -0.942    0.346
## code_presentation2013J  0.28078    0.04255   6.598 4.16e-11 ***
## code_presentation2014B -0.01135    0.04361  -0.260    0.795
## code_presentation2014J  0.18457    0.04088   4.514 6.35e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 32268  on 23379  degrees of freedom
## AIC: 32276
##
## Number of Fisher Scoring iterations: 3
```

In this case, the isolated model shows the expected increase in the success odds of a presentation starting in October with respect to a presentation starting in February in much clearer way. This result motivates us to regroup the categories by creating two new labels: 'February' would refer to the '2013B' and '2014B' periods, while 'October' would refer to '2013J' and '2014J'.

We will see how this relabeling behaves later on. Before that, we will also take a look at a model with the `code_module` column in isolation:

```
lr_modules = glm(final_result ~ code_module, data=train_courseResults, family = binomial)
summary(lr_modules)
```

```
##
## Call:
## glm(formula = final_result ~ code_module, family = binomial,
##     data = train_courseResults)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.97927    0.09272  10.561  < 2e-16 ***
## code_moduleBBB -0.71650    0.09677  -7.404 1.32e-13 ***
## code_moduleCCC -1.28366    0.09928 -12.930  < 2e-16 ***
## code_moduleDDD -1.16239    0.09736 -11.939  < 2e-16 ***
## code_moduleEEE -0.50532    0.10281  -4.915 8.86e-07 ***
## code_moduleFFF -0.92275    0.09646  -9.566  < 2e-16 ***
```

```
## code_moduleGGG -0.40287    0.10459  -3.852 0.000117 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 31818  on 23376  degrees of freedom
## AIC: 31832
##
## Number of Fisher Scoring iterations: 4
```

This case is particular because what the exploratory data analysis showed is that the group that shows the highest success rates ('AAA') is also the one with less observations. Although we observe that this model with the predictor in isolation shows clearly that 'AAA' module is the one with greater rates of success, regrouping the categories by leaving 'AAA' versus the rest would lead into a more imbalanced feature, which is something that can be avoided. Instead, we are going to take the four courses with the highest success rates ('AAA', 'BBB', 'EEE' and 'GGG') in one group ('ABEG') and the three modules with lower success rate in another category ('CDF') and evaluate how the models perform. With this in mind we do the updates on both columns and rerun the stepwise logistic regression model selection. Afterwards, we compare the two models: the original `lr_step_bic` and the updated `lr_step_bic_update`.

```
# We first do the updates on a temporal dataframe
train_courseResults2 = train_courseResults
train_courseResults2$code_presentation=as.character(train_courseResults2$code_presentation)
train_courseResults2$code_presentation=fct_collapse(train_courseResults2$code_presentation,
                                                    'February' = c('2013B','2014B'))
train_courseResults2$code_presentation=fct_collapse(train_courseResults2$code_presentation,
                                                    'October' = c('2013J','2014J'))
train_courseResults2$code_presentation=as.factor(train_courseResults2$code_presentation)


train_courseResults2$code_module=as.character(train_courseResults2$code_module)
train_courseResults2$code_module=fct_collapse(train_courseResults2$code_module,
                                              'ABEG' = c('AAA','BBB','EEE','GGG'))
train_courseResults2$code_module=fct_collapse(train_courseResults2$code_module,
                                              'CDF' = c('CCC','DDD','FFF'))
train_courseResults2$code_module=as.factor(train_courseResults2$code_module)


lr_fullModel_update = glm(final_result ~ ., data=train_courseResults2, family = binomial)
lr_step_bic_update = step(lr_fullModel_update, direction='both', k=log(n),
                          trace=0, steps=1000)
summary(lr_step_bic_update)
```

```
##
## Call:
## glm(formula = final_result ~ code_module + code_presentation +
##     gender + highest_education + imd_band_middle + age_band +
##     num_of_prev_attempts + studied_credits + disability + log_clicks,
##     family = binomial, data = train_courseResults2)
##
## Coefficients:
```

```
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -9.71915    0.14622 -66.470  < 2e-16 ***
## code_moduleCDF             -1.59015    0.04432 -35.877  < 2e-16 ***
## code_presentationOctober    0.20919    0.03631   5.762 8.32e-09 ***
## genderM                    -0.33803    0.03997  -8.456  < 2e-16 ***
## highest_education           0.39014    0.02596  15.028  < 2e-16 ***
## imd_band_middle             0.62912    0.06422   9.797  < 2e-16 ***
## age_band35<=               -0.35601    0.03986  -8.932  < 2e-16 ***
## num_of_prev_attempts0<     -0.20434    0.05541  -3.688 0.000226 ***
## studied_credits60<         -0.21738    0.03834  -5.670 1.43e-08 ***
## disabilityY                -0.21534    0.06002  -3.588 0.000334 ***
## log_clicks                  1.54353    0.02137  72.214  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 32351  on 23382  degrees of freedom
## Residual deviance: 19681  on 23372  degrees of freedom
## AIC: 19703
##
## Number of Fisher Scoring iterations: 6
```

Model	Null_deviance	Residual_deviance	AIC	BIC
lr_step_bic	32351.1	18395.42	18427.42	18556.38
lr_step_bic_update	32351.1	19681.47	19703.47	19792.13

As a result, we see that the updated version of the two columns shows results that are coherent with the exploration of the data.

Now, the only case in which the coefficient has a sign that does not coincide with what we expected from our visualizations is the one corresponding to the `age_band` feature. In this case we had already grouped the categories in the column in order to obtain a more balanced feature with only two categories, but the imbalance persisted since the '0-35' band had much more observations than the '35<=' band. In order to validate our results, we wanted to check if we have issues regarding collinearity: a situation in which two or more predictors are closely related to one another and that can affect the accuracy of the coefficients. With this in mind, we computed the Variance Inflation Factor (VIF), a metric that can help us assess the situation with respect to collinearity: its smallest possible value is 1 (which indicates absence of collinearity) and, as a rule of thumb, should not exceed 5:

	VIF
code_module	1.520917
code_presentation	1.006015
gender	1.260649
highest_education	1.043821
imd_band_middle	1.020160
age_band	1.045878
num_of_prev_attempts	1.030442
studied_credits	1.058563
disability	1.016019
log_clicks	1.310789

From this validation we notice that all the VIF values are close to 1 and that `age_band` does not stand out for having a high value in comparison with the other predictors. For this reason and since it would be hard to further transform the `age_band` feature, we are going to stick with the methodology and consider that the sign of the coefficient might have to do with the unbalance of the feature or some other confounding effect added by the interaction of the variables. Given that the feature was considered significant in the best model obtained by the stepwise feature selection algorithm, we are going to keep the variable for doing the predictions, but we are going to handle the interpretation of this variable with care. Perhaps the shrinkage models will help us deal with this situation.

Other than that. we obtained a model that is easier to interpret and has more precise meaning for the coefficients than the previous version. On the downside, we observe a drawback regarding the BIC metric. **However, as our goal is to obtain a model that gives us insights about our data, we choose the updated model: `lr_step_bic_update`**. Taking this into account, we commit to our updates:

```
# We update the train test to keep as reference and
# delete the previous models to avoid confusions
train_courseResults = train_courseResults2
rm(lr_fullModel,lr_step_bic,train_courseResults2)

# We also update the test set to maintain coherence when assessing
test_courseResults$code_presentation=as.character(test_courseResults$code_presentation)
test_courseResults$code_presentation=fct_collapse(test_courseResults$code_presentation,
                                                  'February' = c('2013B','2014B'))
test_courseResults$code_presentation=fct_collapse(test_courseResults$code_presentation,
                                                  'October' = c('2013J','2014J'))
test_courseResults$code_presentation=as.factor(test_courseResults$code_presentation)


test_courseResults$code_module=as.character(test_courseResults$code_module)
test_courseResults$code_module=fct_collapse(test_courseResults$code_module,
                                            'ABEG' = c('AAA','BBB','EEE','GGG'))
test_courseResults$code_module=fct_collapse(test_courseResults$code_module,
                                            'CDF' = c('CCC','DDD','FFF'))
test_courseResults$code_module=as.factor(test_courseResults$code_module)
```

4.2.1.2 Fitting the Ridge regression for the logistic model

At this point we can fit our regularized regressions. It is worth mentioning that we are going to use the updated features because the reasons behind the interpretability of the models also apply to this situation.

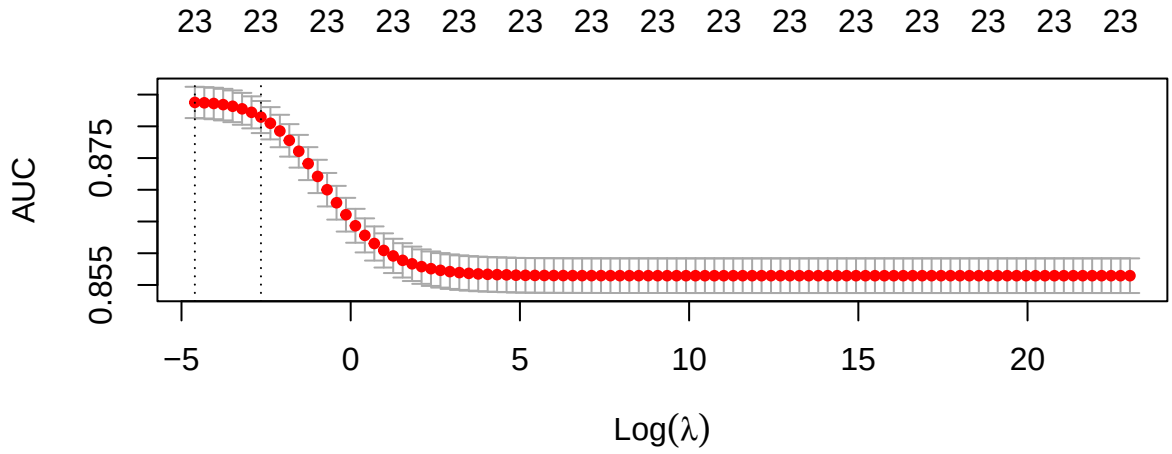
The first instance of this approach will be the Ridge Regression, which is a regression that minimizes a loss function that includes this term: $\lambda \sum_{j=1}^{p} \beta_j^2$, where p is the number of predictors and β_j refers to the coefficients of the regression. In that term, the lambda factor is a hyperparameter that penalizes the complexity of the model. In other words, the lambda factor attempts to shrink the coefficients of the regression that are less relevant. In this case, we are going to use a K-fold cross-validation approach in order to select the best lambda factor. What this means in our case is that, among a possible set of lambda values, for each lambda, a model is trained on a portion of the training set and then is tested (or validated) against the portion of the training set that was left out during the training. This is done K times. As mentioned before, the test set will be used only for the final assessments.

To implement this, we are going to use the `cv.glmnet` function from the `glmnet` package, leaving the `nfolds` parameter by default. Thus, K is set to 10. Also, by default, the function is automatically applied to standardized variables. Additionally, we used a grid of 100 possible lambda values with a wide range: it can go from 0.01 to 10^{10}. In addition, we set the alpha parameter as 0. The purpose of this is to

instruct `cv.glmnet` to apply the Ridge Regression (alpha = 1 corresponds to a Lasso Regression, which will be considered later on). Another thing to consider is that we used the Area Under the Curve (AUC) as the metric to estimate the error on each of the validations. In a nutshell, the AUC is a measure of the performance of a binary classifier that incorporates both the true positive and true false positive rates. For this metric, the greater the value, the better the classifier is.

```
# First we prepare the data to be read by the function. (We need to split X and y)
# The first column of X refers to the intercept and should be removed
X = model.matrix(final_result~., data=train_courseResults)
X = X[,-1]
y = train_courseResults$final_result

# We set a seed for reproducibility and create the lambda grid
set.seed(123)
grid <- 10^seq(10, -2, length=100)
lr_ridge = cv.glmnet(X, y, family='binomial',
                     alpha = 0, nfold=10, type.measure='auc',lambda = grid)
plot(lr_ridge)
```



We observe that there is a decreasing relation between the performance of the regression and the lambda applied. In fact, the lambda that is related with the best cross-validation performance is the minimal one possible (0.01). This means that the Ridge Regression "wants" to behave almost as a non-regularized regression with all the predictors.

As shown below, this model even assigns a value (although it is very small) to the `date_registration` column. This feature, as we saw during the exploratory data analysis, had an almost identical distribution for the 'Successful' and 'Unsuccessful' groups. In the case of the `region` feature (which was not significant in the stepwise feature selection), the coefficients are much larger. This phenomenon results in a model that is harder to interpret, but perhaps is better at doing predicting, unless it is suffering from overfitting. This will be assessed during the final evaluation.

On the other hand, we observe that, again, we obtained an unexpected sign only for the `age_band` variable, although we used a different methodology that obtains an optimum with certain restrictions. Since it is a consistent phenomenon, we will do the same thing we did for the model obtained by stepwise feature selection: keep the variable in the model for predictions but interpret its coefficient with caution.

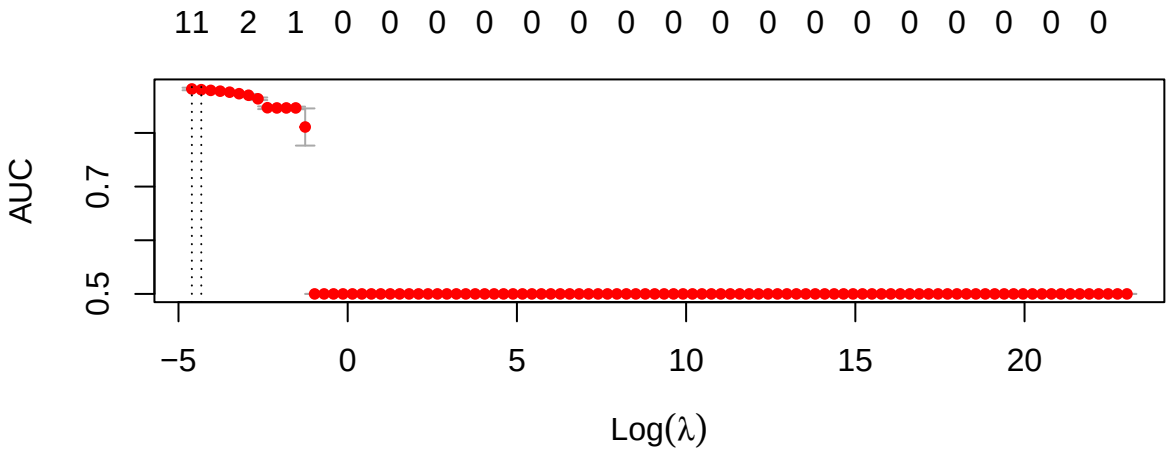
	Beta
(Intercept)	-7.7623221
code_moduleCDF	-1.2479049
code_presentationOctober	0.1962083
genderM	-0.2825783
regionEast Midlands Region	-0.0381125
regionIreland	0.0250638
regionLondon Region	-0.1071226
regionNorth Region	0.0384003
regionNorth Western Region	-0.1100272
regionScotland	-0.3155540
regionSouth East Region	0.1474016
regionSouth Region	0.0797408
regionSouth West Region	0.1189862
regionWales	-0.1703486
regionWest Midlands Region	-0.0049444
regionYorkshire Region	-0.0637530
highest_education	0.3633159
imd_band_middle	0.5093818
age_band35<=	-0.2292900
num_of_prev_attempts0<	-0.2284033
studied_credits60<	-0.2069756
disabilityY	-0.2076304
log_clicks	1.2332536
date_registration	0.0003274

4.2.1.3 Fitting the Lasso Regression for the logistic model

The Lasso Regression is similar to the Ridge Regression. The main difference is that the term used to penalize the complexity of the models changes: it becomes $\lambda \sum_{j=1}^{p} |\beta_j|$. In this case, the $|\beta_j|$ coefficient is not squared. The consequence of this adjustment is that the Lasso Regression can yield coefficients equal to 0, unlike the Ridge counterpart. This expected sparsity in the final model goes inline with our goal to end up with interpretable results. Apart from the change in the penalty term, the procedure is almost identical to the previous one (we also run a K-fold Cross Validation to select the lambda):

```
# We set a seed for reproducibility
# (the other variables are the same as in the Ridge Regression)
set.seed(123)

# We set alpha = 1 for a Lasso regression
lr_lasso = cv.glmnet(X, y, family='binomial',
                     alpha = 1, nfold=10, type.measure='auc',lambda = grid)
plot(lr_lasso)
```



Although the plot looks different, we also see that the performance of the model is worse as lambda becomes larger. Again, the best lambda is 0.01, but the regression coefficients look different when considering the Lasso penalty term:

	Beta
(Intercept)	-8.3183564
code_moduleCDF	-1.3328284
code_presentationOctober	0.0730671
genderM	-0.1631837
regionScotland	-0.0800404
highest_education	0.2717386
imd_band_middle	0.3752330
age_band35<=	-0.1032700
num_of_prev_attempts0<	-0.0757507
studied_credits60<	-0.0918554
disabilityY	-0.0207019
log_clicks	1.3347029

In this case, we notice that the predictors with non-zero coefficients are almost the same that we obtained with the stepwise subset selection. The only difference in that sense is that this version of the regression suggests that the 'Scotland' region is expected to have less success odds than the reference category ('East Anglian Region'). However, it is the only region that shows some kind of difference. Apart from that, the two models look very similar.

Once again, the sign of the `age_band` coefficient for the '35<=' band is negative even though we have used three different methods that attempt to select the best possible combination of features. For this reason and the argumentation we used for the previous models, we are going to be consistent with our decision, so we keep the predictor in the model to classify the new examples, while being very cautious regarding the interpretation.

4.2.2 Testing the logistic regression models

4.2.2.1 Testing the stepwise feature selection model for logistic regression

Now that we have fitted the models we can test their performance against the test set. At the end of the document we compare the performance of all the classifiers, but in the section corresponding to each model we show the individual results. We are going to start with the model obtained by stepwise feature selection.

In order to test our results, first we need to turn our logistic regressions into classifiers. For this purpose, we need to establish a threshold for the probability in order to determine whether an observation is classified as 'Successful' or 'Unsuccessful'. In our case, after obtaining the vector of probabilities by plugging the features of the test set in our fitted model, we compute a ROC curve. This curve computes the sensitivity (the proportion of 'Successful' examples that were correctly classified) and the specificity (the proportion of 'Unsuccessful' examples that were correctly classified) for different thresholds. This way we can find the optimal value for the threshold, which is is the one that maximizes the sum of sensitivity and specificity.

After this, by using the optimal threshold, we can obtain the predicted values for the test set and compare them with respect to the true values by using a confusion matrix. At the end of the logistic regression section, we are going to compare the ROC curves for our logistic models. For now, we simply show the resulting confusion matrix of the model obtained by stepwise feature selection:

```
# We predict the probabilities and obtain the roc curve
prob_lr_step_test = predict(lr_step_bic_update, newdata=test_courseResults,
                            type='response')
roc_lr_step_test = roc(test_courseResults$final_result, prob_lr_step_test,
                       levels=c('Unsuccessful', 'Successful'))

# Using the threshold that maximizes the sum of sensitivity and specificity,
# we predict our labels
best_thresh_lr_step = coords(roc_lr_step_test, 'best')$threshold
pred_lr_step_test = factor(ifelse(prob_lr_step_test >= best_thresh_lr_step,
                                  'Successful', 'Unsuccessful'),
                           levels = c('Unsuccessful','Successful'),
                           ordered = TRUE)
cm_lr_step_test = confusionMatrix(data=pred_lr_step_test,
                                  reference=test_courseResults$final_result,
                                  positive='Successful')
print(cm_lr_step_test$table)
```

```
##                Reference
## Prediction     Unsuccessful Successful
##   Unsuccessful         2078        479
##   Successful            691       2597
```

The resulting confusion matrix allows us to observe that most of the predicted values coincide with the reference observations of the test set. Also, we notice that this is something that happens in the two groups. However, in order to characterize our results in a better way, we decided to make a summary that comprises the overall accuracy, the sensitivity and the specificity. This is the format that we are going to use from now on to assess the performance of our models with respect to the test set:

```
accuracy_step = cm_lr_step_test$overall['Accuracy']
sensitivity_step = cm_lr_step_test$byClass['Sensitivity']
specificity_step = cm_lr_step_test$byClass['Specificity']

results_step = data.frame(LR_stepwise = c(accuracy_step,sensitivity_step,specificity_step))
kable(results_step)
```

	LR_stepwise
Accuracy	0.7998289
Sensitivity	0.8442783
Specificity	0.7504514

From this summary, we confirm that most of our examples were correctly classified and that the results are more or less similar between the two classes, although the sensitivity is better than the specificity.

4.2.2.2 Testing the Ridge regression for the logistic model

For the Ridge logistic regression, we use the same procedure we applied for the stepwise feature selection model (computing the probabilities, obtaining the ROC curve, selecting the best threshold and predicting the labels of the test set). It is worth mentioning that we predicted the probabilities with the model corresponding to best lambda obtained for the Ridge regression (0.01). For the sake of clarity, we display our results directly:

	LR_ridge
Accuracy	0.8001711
Sensitivity	0.8293238
Specificity	0.7677862

As we can see, the overall accuracy, the sensitivity and the specificity resemble the results obtained for the logistic regression with stepwise feature selection.

4.2.2.3 Testing the Lasso regression for the logistic model

Since the Ridge and the Lasso logistic regressions are analogous, we follow the same steps as before, also by using the model corresponding to the best lambda obtained for the Lasso regression (which is also 0.01). The summary is the following:

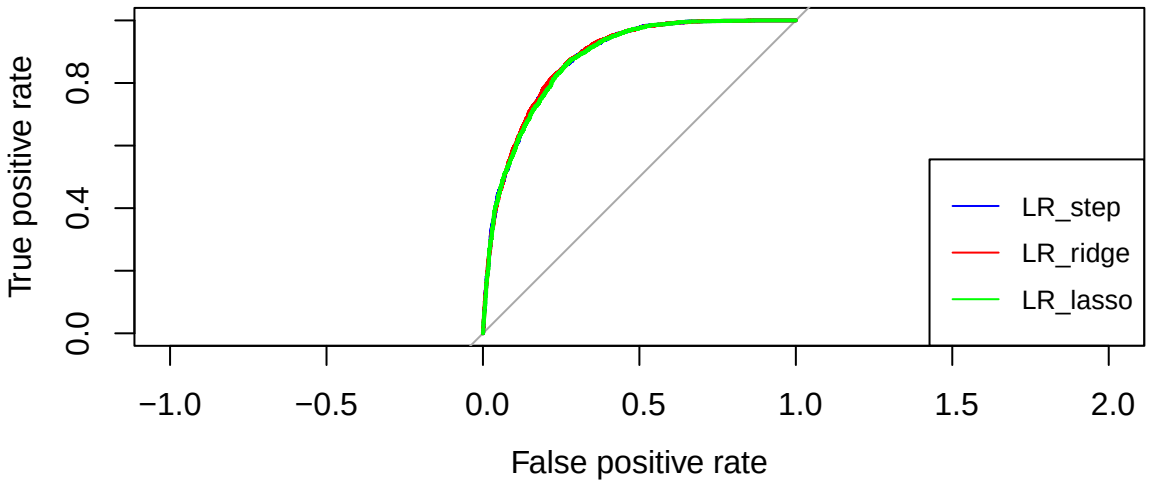
	LR_lasso
Accuracy	0.8001711
Sensitivity	0.8582575
Specificity	0.7356446

The results are very similar to the ones obtained with stepwise feature selection and Ridge Regression. This makes sense as we observed that the predictors were similar in the three cases.

4.2.2.4 Comparing the ROC curves of the logistic classifier

To finalize with the logistic regression models, we can compare the ROC curves obtained for each of the models. In this case, we plot the true positive rate (the sensitivity) in the Y-axis and the false positive rate (which is 1 - specificity) in the X-axis. Since we want to maximize the sensitivity and the specificity, the ideal curve is one that 'hugs' the top left corner, so that its shape forms a square of side 1. Therefore, the area under he curve (AUC) can be used to measure the

overall performance of the classifiers. The higher the value of the AUC, the better. For our classification task, the three logistic classifiers we used show very similar performances, as shown below:



Moreover, we evidence the similarity among the performances by looking at the AUC of the three models. When rounded to two decimals, all models obtained an AUC of 0.88.

4.3 Linear Discriminant Analysis

4.3.1 Fitting the LDA models

Now, we move on to a different methodology for the classification. This methodology, as well as the QDA and the Naive Bayes, use the Bayes Theorem to estimate the probability that an observation belongs to a certain group of the target variable. The probabilities are based on the distribution of the features within each of the groups. However, these three methodologies use different approaches and assumptions in order to do this. The Linear Discriminant analysis, for example, assumes that, within each of the categories of the response variable, the predictors are normally distributed with a different mean for each group, but a shared covariance matrix (or a shared variance in the case of only one predictor). Taking this assumption into account, the model assigns a probability score to each observation, indicating its likelihood of belonging to each group. When analyzing a new example, the model classifies it into the category with the highest score.

What this implies is that, in order to comply with the assumptions of the model, our features should show some normality when partitioned by the target variable labels. Additionally, we can infer that well separated distributions (with distant means) help to differentiate the two groups. As we have seen in the exploratory data analysis, our numeric features do not really appear to satisfy with these conditions.

First, regarding normality, we saw that when treating each of our numeric features separately with respect to the target variable, `date_registration` did not even show a bell-shaped distribution and had deviations with respect to the theoretical normal quantiles in the QQ plot. Alternatively, when using a log transformation, the `total_sum_click` feature did have a density plot more similar shape to the normal distribution, but still did not seem to show normality in the QQ plot. On the other hand, the other two numerical features that we have are somewhat "artificial" in the sense that they were categorical columns converted into numbers due to their nature, so the normality condition does not really apply to them.

Second, about the differentiation of the distributions, the `date_registration` column showed almost identical density plots and almost identical means when partitioned by 'Successful' and 'Unsuccessful' observations. In contrast, the log transformation of `total_sum_click` did show a clear difference between the

density plots (and the means) of the two groups. As for the other two columns (`highest_education` and `imd_band_middle`), we observed that there seemed to be an increasing relation between their values and the success rates, so, when split into the two categories of the `final_result`, they might have separated means.

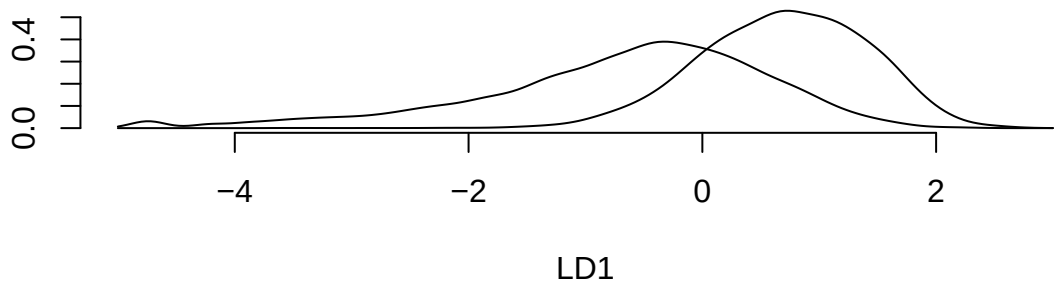
Although all this information is useful, we wanted to go a little deeper. More precisely, as a measure of group separation, we wanted to assess if the means between each `final_reusult` category are different. With this purpose, although our features do not seem to show normality, just in order to get an informal intuition, we conducted an ANOVA test for each of our numerical predictors. In this test, the null hypothesis is that the group means are equal, so a small p-value indicates strong evidence against this hypothesis (thus suggests that the means are different). Also, it is worth mentioning that the ANOVA model assumes homogeneity of the variances across the groups. To test this assumption, we ran a Bartlett's test. In this case, to summarize, the null hypothesis is that the variances are equal.

Feature	Bartlett_pvalue	ANOVA_pvalue
log_clicks	0.000	0.000
date_registration	0.165	0.375
highest_education	0.075	0.000
imd_band_middle	0.755	0.000

As we can see, in most of the cases (the only exception is the log transformation of `total_sum_click`), if the assumptions of each test were complied, there would not be enough evidence to reject the hypothesis that the variances are equal if we set a significance level of 5%. On the contrary, under the same assumptions, the results would suggest that, at least for the `highest_education` and `imd_band_middle` features, there is a difference between the means. However, we must not forget the fact that our predictors do not appear to be normal and these tests were just to get an intuition.

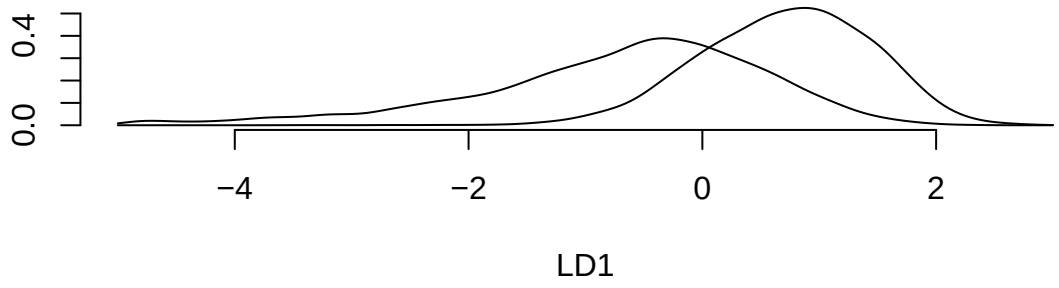
Taking the information from these informal tests, but specially from the exploratory data analysis, we are driven to fit two models and check how well they perform: one model with just the `log_clicks` column (which is the one that shows less deviation from the normal distribution and also displays a difference in the distributions of the two groups) and one model with `log_clicks`, `highest_education` and `imd_band_middle`. In both cases we drop the `date_registration`, which actually coincides with the results from the logistic models, that showed that this feature is not significant. First, we fit the `log_clicks` model and plot the separation of the groups according to the scores the model has assigned.

```
lda_fit_logClicks = lda(final_result~log_clicks,
                        data=train_courseResults)
plot(lda_fit_logClicks, type = 'density')
```



Next, we fit the model with the other variables and notice that there does not seem to be much difference in the group separation. We will have a better idea of the performance of the models during the evaluation stage.

```
lda_fit_fuller = lda(final_result~log_clicks+imd_band_middle+highest_education,
                     data=train_courseResults)
plot(lda_fit_fuller, type = 'density')
```



4.3.2 Testing the LDA models

We can test both LDA models: the one that only takes into account the log transformation of the clicks and the "fuller" version that also includes `imd_band_middle` and `highest_education`. In order to do this, we just predict the labels using the features of the test set and compare the predictions with the reference values of this set. Each predicted label is assigned to the class that obtained a greater probability according to our model. As we did with the logistic regression models, we are going to use three measures to assess the performance of the model: overall accuracy, sensitivity and specificity:

```
# First we make the predictions
lda_pred_logClicks = predict(lda_fit_logClicks, test_courseResults)
lda_pred_fuller = predict(lda_fit_fuller, test_courseResults)
lda_class_logClicks = lda_pred_logClicks$class
lda_class_fuller = lda_pred_fuller$class

# Now we obtain the confusion matrix and the values
# for sensitivity and specificity
cm_lda_logClicks = confusionMatrix(data=lda_class_logClicks,
                                   reference=test_courseResults$final_result,
                                   positive='Successful')
cm_lda_fuller = confusionMatrix(data=lda_class_fuller,
                                   reference=test_courseResults$final_result,
                                   positive='Successful')

accuracy_lda_logClicks = cm_lda_logClicks$overall['Accuracy']
sensitivity_lda_logClicks = cm_lda_logClicks$byClass['Sensitivity']
specificity_lda_logClicks = cm_lda_logClicks$byClass['Specificity']

accuracy_lda_fuller = cm_lda_fuller$overall['Accuracy']
```

```
sensitivity_lda_fuller = cm_lda_fuller$byClass['Sensitivity']
specificity_lda_fuller = cm_lda_fuller$byClass['Specificity']

results_lda = data.frame(LDA_logClicks=c(accuracy_lda_logClicks,
                                         sensitivity_lda_logClicks,
                                         specificity_lda_logClicks),
                         LDA_fuller=c(accuracy_lda_fuller,
                                      sensitivity_lda_fuller,
                                      specificity_lda_fuller))
kable(results_lda)
```

	LDA_logClicks	LDA_fuller
Accuracy	0.7582549	0.7563730
Sensitivity	0.8442783	0.8364759
Specificity	0.6626941	0.6673889

The comparison that takes into account all classifiers is at the end of the document, but for now, we can notice that the LDA preformed worse than the logistic classifiers with respect to the overall accuracy. Something that might be crucial in this aspect is that, as we saw during the fitting stage, the assumptions of the LDA were not fulfilled by our data. Additionally, we can observe that this model seems to be better at predicting 'Successful' examples than 'Unsuccessful' observations, since the sensitivity is above 0.8 and the specificity is below 0.7 in both cases.

4.4 Quadratic Discriminant Analysis

4.4.1 Fitting the QDA models

Since this method has almost the same assumptions as the LDA, the approach we are going to take is quite similar. The difference in this case is that the QDA removes the supposition that the covariance matrix (or the variance, in the one-dimensional case) is shared across groups and estimates a variance matrix for each class separately. Since this is a relaxation on the premises, the new approach does not contradict our previous reasoning. Moreover, from the density plot of the log transformation of `total_sum_click`, we observed that the dispersion of the two classes looked different. Therefore, we are going to fit the same models (one with only `log_clicks` and one that also adds `highest_education` and `imd_band_middle`) but in the QDA version:

```
qda_fit_logClicks = qda(final_result~log_clicks,data=train_courseResults)
qda_fit_fuller = qda(final_result~log_clicks+imd_band_middle+highest_education,
                     data=train_courseResults)
```

4.4.2 Testing the QDA models

As we did for the LDA models, now we are going to test the two QDA classifiers (the one with just the log transformation of the clicks and the fuller model) on the test set. The procedure for predicting the labels of the test set and comparing them with respect to the true values is also equivalent to the one applied for LDA. Also, as we have done for LDA and the logistic classifiers, we will take into account the overall accuracy, the sensitivity and the specificity. Applying these procedures yields the following results for QDA:

	QDA_logClicks	QDA_fuller
Accuracy	0.7544910	0.7544910
Sensitivity	0.8699610	0.8660598
Specificity	0.6262189	0.6305525

The QDA obtained results with the same patterns as those of the LDA, especially regarding the overall accuracy. However, we observe that the difference between sensitivity and specificity is larger in the case of the QDA. Once again, a possible cause of the decrease in the overall performance, compared to the logistic classifiers, is that our data does not meet the assumptions of the QDA model, even if in this case the condition that the variance is the same for the two groups was relaxed.

4.5 Naive Bayes

4.5.1 Fitting the Naive Bayes model

In a different direction, what the Naive Bayes model does is that assumes that, within each class, the predictors are independent. This way, the probability that a series of features occur jointly inside each group, are more easily estimated as the multiplication of the probability that each feature shows within that group. This is a strong assumption, but it is mainly made for convenience. Also, we want to highlight that this model can handle both numerical and categorical features. For the numerical columns, for example, it can assume that they behave normally, with the additional supposition that the covariance matrix is diagonal (because of the independence). For the categorical features, it can count the proportion of observations that each category has inside the groups of the target variable.

Since there are many possibilities to arrange the predictors, we would like to provide the model the most relevant information possible in order to avoid overfitting and to obtain models that we understand. For this reason, we are going to use the same features that resulted from the stepwise feature selection procedure for the Logistic Regression, which was a concise model with interpretable coefficients.

```
nb_fit = naiveBayes(final_result ~ . - region - date_registration,
                    data = train_courseResults)
```

4.5.2 Testing the Naive Bayes model

As we have done with the previous models, for us, the testing process consists in obtaining the overall accuracy, the sensitivity and the specificity by comparing the predicted labels that result from our fitted model against the true values in the test set. The only difference with LDA and QDA is the model used for predicting the labels, but in all cases the label is assigned to the category with highest probability. According to this, the naive Bayes results are the following:

	Naive_Bayes
Accuracy	0.7774166
Sensitivity	0.8741873
Specificity	0.6699169

We see some improvement in the overall accuracy with respect to the LDA and QDA models. Although the independence of the features is a strong assumption, it seems that the increase in the number of predictor does help to slightly improve the overall performance of the model. However, we still observe a considerable gap between sensitivity and specificity.

4.6 K-Nearest Neighbors

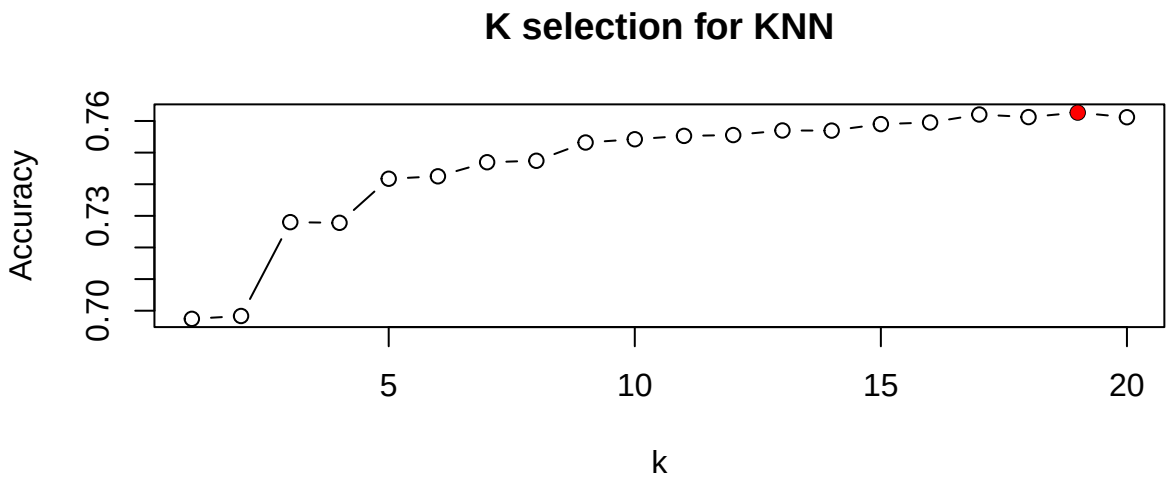
4.6.1 Selecting K

The K-Nearest Neighbors approach is much different from the ones we have considered in the sense that it is a non-parametric approach. What the algorithm does is that, for assigning the class of a new observation, it locates the K closest training examples from the metric space of the training features and selects the label of the majority class of these examples. Therefore, there are not parameters to estimate in order to do the prediction.

What we can do is to select a K value that gives us good results. For this purpose, still using only the training set, we will do a 5-Fold Cross-Validation estimation for the overall accuracy (i.e. the number of correctly classified examples) with different K's. Then, we select the K that showed the best performance in the training set. Finally, we use this value to perform the final evaluation. We are going to check for values of K between 1 and 20.

In addition, it is worth mentioning that the features that are going to constitute the metric space should be scaled. The reason for this is that if we omit this step, one predictor could be overshadowing the others (a big change in a feature with small range will not affect the overall distance as much as a small variation in a feature with great range).

Lastly, as we do not want to add variables that do not offer valuable information, we are going to stick with the numerical features that have been relevant in our analysis: `log_clicks`, `imd_band_middle` and `highest_education`.



Although there are some steps in which there is no improvement, the accuracy seems to show a an increasing tendency as the k value increases. For our procedure and the random seed we used in order to achieve reproducibility, the best K was 19. It is worth noting that after a dramatic improvement with K = 3, the accuracy increases at a slower rate as K is larger, suggesting an stabilization. For this reason, even if our K is close to the limit we imposed, we are going to stick with the results of our cross validation approach and use this K for testing the KNN classifier.

4.6.2 Testing the K-Nearest Neighbor classifier

In this case, testing is a little different. We just run the KNN algorithm on our test set examples using the metric space created by the features of our training set and the best K we obtained (19). This way we obtain

the predicted labels. From this point, we compare the predicted labels with respect to the reference values and compute the overall accuracy, the sensitivity and the specificity as we did for all the other classifiers:

	KNN
Accuracy	0.7478186
Sensitivity	0.8016905
Specificity	0.6879740

We see that, in our case, this approach increases the specificity with respect to LDA, QDA and Naive Bayes, although the overall accuracy is worse, which implies that the sensitivity is also worse.

5. Conclusions

5.1 Model comparison

Now that we have fitted and tested all our models, we can compare them by using the metrics we established with respect to the test set: overall accuracy, sensitivity and specificity:

	Accuracy	Sensitivity	Specificity
LR_stepwise	0.7998289	0.8442783	0.7504514
LR_ridge	0.8001711	0.8293238	0.7677862
LR_lasso	0.8001711	0.8582575	0.7356446
LDA_logClicks	0.7582549	0.8442783	0.6626941
LDA_fuller	0.7563730	0.8364759	0.6673889
QDA_logClicks	0.7544910	0.8699610	0.6262189
QDA_fuller	0.7544910	0.8660598	0.6305525
Naive_Bayes	0.7774166	0.8741873	0.6699169
KNN	0.7478186	0.8016905	0.6879740

As a result, we observe that the most accurate models were the ones that involved the logistic regression. In our case, this makes sense since the data we had did not really satisfy the assumptions made by the other models' formulations, for example, regarding the normality of our predictors. Also, it is worth noting that we had many categorical features and not many numerical columns, so the models that did not incorporate categorical data had less information to make the predictions. For instance, we can see that the KNN, a model that only used numerical variables and did not have such strong suppositions, was the worst performing model overall. This is also consistent with the fact that, after the logistic classifiers, the methodology that obtained the best overall accuracy was the other one that used categorical features: the naive Bayes model.

On the other hand, it is evident that all the classifiers were better at identifying 'Succesful' cases than 'Unsuccessful' cases, given that in each case the sensitivity was higher than the specificity. Yet, again, the models that showed a more balanced performance were the logistic classifiers.

Now, regarding the three logistic regressions we used, there is not a huge difference in the results. However, as we saw when we were fitting the models, the stepwise feature selection and the Lasso regression yield sparser models that are easier to interpret. Also, these two approaches resulted in very similar models with respect to which features are significant to predict a 'Successful' or 'Unsuccessful' final result of the course. For this reason, picking one model or the other is almost indifferent, although there is very small difference in the metrics obtained (the Lasso model has slightly better overall accuracy and slightly worse specificity).

5.2 Interpretation of the results

Finally, since our goal was to understand how the explanatory variables are related to the `final_result` groups, we can comment a little about the coefficients we obtained with the stepwise feature selection methodology:

	Beta
(Intercept)	-9.7191480
code_moduleCDF	-1.5901463
code_presentationOctober	0.2091947
genderM	-0.3380262
highest_education	0.3901383
imd_band_middle	0.6291216
age_band35<=	-0.3560112
num_of_prev_attempts0<	-0.2043438
studied_credits60<	-0.2173819
disabilityY	-0.2153358
log_clicks	1.5435331

From this, the most obvious conclusion is that when the students interact more with the Virtual Learning Environment they have better chances to succeed in the courses. However, there are other important relationships between the odds of success and some other features. Just as a small experiment, we evaluated the logistic regression that involved only the log transformation of the clicks against the test set. Indeed, the results worsen (especially the specificity) if we exclude the other explanatory variables:

	LR_logClicks
Accuracy	0.7596236
Sensitivity	0.8335501
Specificity	0.6775009

For this reason, it is useful to incorporate and interpret the significant features correctly. From the perspective of someone who wanted to monitor which actionable insights are related with successful results in the courses, one possible conclusion is that the students with worse deprivation levels (recalling that the data in here is presented such that a higher `imd_band_middle` is related to less deprivation) and a lower level of education have less odds to succeed. This also happens with the students who reported a disability and the ones that had already attempted the modules before.

With this information, the people in charge of organizing the courses might consider offering special help to these groups of students. Additionally, since the difference is significant, they could assess what is happening so that the courses that start in October and the 'AAA', 'BBB', 'EEE' and 'GGG' modules have higher success rates. They could try to replicate the dynamics of these modules in their other courses. Likewise, they could try to understand what makes females more successful than males and see if they can obtain insights for the courses from this. On the other hand, since a higher load of credits is related to lower odds of success, it would be recommended that the students take the usual amount of credits, which go from 30 to 60.

In contrast, We would not draw any strong conclusions regarding the `age_band` feature. Although its presence in the models improved the expected performance of the classifiers, even after the transformations, this was the only predictor that did not show consistent results with what we saw in the exploratory data analysis. Recalling that this column was imbalanced (almost 70% of the students belong to the '0-35' band) and that there might be some confounding interaction between the variables, we recommend to avoid making decisions regarding this column without doing further research on it.

Finally, with respect to the amount of interactions, the people in charge of the courses could monitor if the students are interacting less or more with respect to their peers and try to notify them accordingly in order to help them improve their performance. This might be similar to the actual use of the data, since the Open University Learning Analytics information is used to "improve students' learning experience by providing informed guidance and to optimise learning materials", according to the creators of the dataset.