CEIT 418 Data Science Project

As your final data science project for CEIT 418, you will explore an educational dataset, and build a classification machine learning model. As the submission, you should submit in OdtuClass the following items:

1. The url of the Google Colab document, and
2. The PDF version of the complete notebook.

Submissions missing any of the files will not be evaluated.

In the first part, mostly you are expected to explore different tables (possible by using functions such as `head`, `shape`, `info`, and `describe`), deal with duplicate records and missing values, and perform some exploratory tasks.

In the second part, you will build a classification model and report its accuracy.

Important Information

For any action you take on the data, you should **explain your rationale** (e.g., I took into account colmuns X and Y when detecting duplicates because Z). Also, you should **provide an explanation/interpretation for outputs** produced by your code (e.g., based on this result, A and B columns can be dropped since they carry mostly missing values).

About the Dataset

For the final project, you will work on a public educational dataset shared by UK Open University. Although throughout this document you will be provided with sufficient information about this public dataset, you are strongly recommended to refer to https://analyse.kmi.open.ac.uk/open_dataset for more detailed information.

There is also Kaggle page where you can see some analysis performed shared by other publicly. I think they can be also helpful if you want to explore the dataset beyond this assignment. https://www.kaggle.com/datasets/rocki37/open-university-learning-analytics-dataset

1. Exploratory Analysis

1.1. Courses Table

Courses table (`courses.csv`) contains the list of all available modules and their presentations.

The columns are:

- **code_module** – code name of the module, which serves as the identifier.
- **code_presentation** – code name of the presentation. It consists of the year and "B" for the presentation starting in February and "J" for the presentation starting in October.
- **length** - length of the module-presentation in days.

The structure of B and J presentations may differ and therefore it is good practice to analyse the B and J presentations separately. Nevertheless, for some presentations the corresponding previous B/J presentation do not exist and therefore the J presentation must be used to inform the B presentation or vice versa. In the dataset this is the case of CCC, EEE and GGG modules.

In [47]:
```
import numpy as np
import pandas as pd
```

TASK1: Identify and treat duplicate/missing values (if there is any).

In [116…
```
courses = pd.read_csv('dataset/courses.csv')
courses.head(3)
```

Out[116]:

	code_module	code_presentation	module_presentation_length
0	AAA	2013J	268
1	AAA	2014J	269
2	BBB	2013J	268

In [117…
```
courses.shape
```

Out[117]:
```
(22, 3)
```

In [118…
```
courses.isnull().sum()
```

Out[118]:
```
code_module                 0
code_presentation           0
module_presentation_length  0
dtype: int64
```

In [119…
```
courses[courses.duplicated() == True]
```

Out[119]:

code_module	code_presentation	module_presentation_length

TASK2: Find out how many courses started in February vs October, and compare their length. Interpret the results.

In [120…
```
coursesJ = courses[courses.code_presentation.str.contains("J")]
coursesJ.module_presentation_length.describe()
```

Out[120]:
```
count     13.000000
mean     266.384615
std        3.428762
min      261.000000
25%      262.000000
50%      268.000000
75%      269.000000
max      269.000000
Name: module_presentation_length, dtype: float64
```

In [121…
```
coursesB = courses[courses.code_presentation.str.contains("B")]
coursesB.module_presentation_length.describe()
```

Out[121]:
```
count      9.000000
mean     239.888889
std        2.260777
min      234.000000
25%      240.000000
50%      241.000000
75%      241.000000
max      241.000000
Name: module_presentation_length, dtype: float64
```

TASK3: Find the courses that were offered two years consecutively in the same semester (such as, 2013B and 2014B, or 2013J and 2014J), and check if they had the same length or not.

In [122…]:
```
yearCol = courses.apply(lambda row: row["code_presentation"][:4], axis='columns')
semCol =  courses.apply(lambda row: row["code_presentation"][4:], axis='columns')
courses['Year'] = yearCol.astype(int)
courses['Semester'] = semCol
courses.head()
```

Out[122]:

	code_module	code_presentation	module_presentation_length	Year	Semester
0	AAA	2013J	268	2013	J
1	AAA	2014J	269	2014	J
2	BBB	2013J	268	2013	J
3	BBB	2014J	262	2014	J
4	BBB	2013B	240	2013	B

In [148…]:
```
def generateCols(s):
    s['difYear'] = s['Year'] - s['Year2']
    s['isSame'] = s['module_presentation_length'] == s['Duration2']
    s['howMany'] = len(s)
    return s
```

In [149…]:
```
coursesJ = courses[courses["Semester"] == "J"]
coursesJ = pd.concat([coursesJ, coursesJ.Year.shift(1).rename("Year2")], axis=1)
coursesJ = pd.concat([coursesJ, coursesJ.module_presentation_length.shift(1).rename("Duration2")], axis=1)
coursesJ.head(5)
```

Out[149]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2
0	AAA	2013J	268	2013	J	NaN	NaN
1	AAA	2014J	269	2014	J	2013.0	268.0
2	BBB	2013J	268	2013	J	2014.0	269.0
3	BBB	2014J	262	2014	J	2013.0	268.0
6	CCC	2014J	269	2014	J	2014.0	262.0

In [150…]:
```
coursesJUpdated = coursesJ.groupby('code_module').apply(generateCols)
coursesJUpdated[coursesJUpdated.difYear == 1]
```

Out[150]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2	difYear	isSame	howMany
1	AAA	2014J	269	2014	J	2013.0	268.0	1.0	False	2
3	BBB	2014J	262	2014	J	2013.0	268.0	1.0	False	2
9	DDD	2014J	262	2014	J	2013.0	261.0	1.0	False	2
13	EEE	2014J	269	2014	J	2013.0	268.0	1.0	False	2
16	FFF	2014J	269	2014	J	2013.0	268.0	1.0	False	2
20	GGG	2014J	269	2014	J	2013.0	261.0	1.0	False	2

In [151…
```
coursesB = courses[courses["Semester"] == "B"]
coursesB = pd.concat([coursesB, coursesB.Year.shift(1).rename("Year2")], axis=1)
coursesB = pd.concat([coursesB, coursesB.module_presentation_length.shift(1).rename("Duration2")], axis=1)
coursesB.head(5)
```

Out[151]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2
4	BBB	2013B	240	2013	B	NaN	NaN
5	BBB	2014B	234	2014	B	2013.0	240.0
7	CCC	2014B	241	2014	B	2014.0	234.0
10	DDD	2013B	240	2013	B	2014.0	241.0
11	DDD	2014B	241	2014	B	2013.0	240.0

In [152…
```
coursesBUpdated = coursesB.groupby('code_module').apply(generateCols)
coursesBUpdated[coursesBUpdated.difYear == 1]
```

Out[152]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2	difYear	isSame	howMany
5	BBB	2014B	234	2014	B	2013.0	240.0	1.0	False	2
11	DDD	2014B	241	2014	B	2013.0	240.0	1.0	False	2
18	FFF	2014B	241	2014	B	2013.0	240.0	1.0	False	2

TASK4: Find the courses that were offered in two consecutive semesters of the same year (such as, 2013B and 2013J, or 2014B and 2014J), and

check if they had the same length or not.

In [155…

```
coursesSameYear = pd.concat([courses, courses.Year.shift(1).rename("Year2")], axis=1)
coursesSameYear= pd.concat([coursesSameYear, coursesSameYear.module_presentation_length.shift(1).rename("Duration2")], axis=1)
coursesSameYear.head(25)
```

Out[155]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2
0	AAA	2013J	268	2013	J	NaN	NaN
1	AAA	2014J	269	2014	J	2013.0	268.0
2	BBB	2013J	268	2013	J	2014.0	269.0
3	BBB	2014J	262	2014	J	2013.0	268.0
4	BBB	2013B	240	2013	B	2014.0	262.0
5	BBB	2014B	234	2014	B	2013.0	240.0
6	CCC	2014J	269	2014	J	2014.0	234.0
7	CCC	2014B	241	2014	B	2014.0	269.0
8	DDD	2013J	261	2013	J	2014.0	241.0
9	DDD	2014J	262	2014	J	2013.0	261.0
10	DDD	2013B	240	2013	B	2014.0	262.0
11	DDD	2014B	241	2014	B	2013.0	240.0
12	EEE	2013J	268	2013	J	2014.0	241.0
13	EEE	2014J	269	2014	J	2013.0	268.0
14	EEE	2014B	241	2014	B	2014.0	269.0
15	FFF	2013J	268	2013	J	2014.0	241.0
16	FFF	2014J	269	2014	J	2013.0	268.0
17	FFF	2013B	240	2013	B	2014.0	269.0
18	FFF	2014B	241	2014	B	2013.0	240.0
19	GGG	2013J	261	2013	J	2014.0	241.0
20	GGG	2014J	269	2014	J	2013.0	261.0
21	GGG	2014B	241	2014	B	2014.0	269.0

In [154…

```
coursesSameYear = coursesSameYear.groupby('code_module').apply(generateCols)
coursesSameYear[(coursesSameYear.difYear == 0) & (coursesSameYear.howMany > 1)]
```

Out[154]:

	code_module	code_presentation	module_presentation_length	Year	Semester	Year2	Duration2	difYear	isSame	howMany
6	CCC	2014J	269	2014	J	2014.0	234.0	0.0	False	2
7	CCC	2014B	241	2014	B	2014.0	269.0	0.0	False	2
14	EEE	2014B	241	2014	B	2014.0	269.0	0.0	False	3
21	GGG	2014B	241	2014	B	2014.0	269.0	0.0	False	3

1.2. Student Info Table

StudentInfo (`studentInfo.csv`) file contains **demographic** information about the students together with their final result. File contains the following columns:

- **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.
- **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 Depravation band of the place where the student lived during the module-presentation.
- **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.

TASK1: Identify and treat duplicate/missing values (if there is any)

In [183…

```
students = pd.read_csv('dataset/studentInfo.csv')
students.shape
```

Out[183]:

```
(32593, 12)
```

In [184…

```
students.head(2)
```

Out[184]:

	code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_prev_attempts	studied_credits	disab
0	AAA	2013J	11391	M	East Anglian Region	HE Qualification	90-100%	55<=	0	240	
1	AAA	2013J	28400	F	Scotland	HE Qualification	20-30%	35-55	0	60	

In [185…
```
students.drop_duplicates(subset=['id_student', 'code_module', 'code_presentation'], inplace=True)
students.shape
```

Out[185]:
```
(32593, 12)
```

In [186…
```
students.isnull().sum()
```

Out[186]:
```
code_module               0
code_presentation         0
id_student                0
gender                    0
region                    0
highest_education         0
imd_band               1111
age_band                  0
num_of_prev_attempts      0
studied_credits           0
disability                0
final_result              0
dtype: int64
```

In [187…
```
students.dropna(subset=['imd_band'], inplace=True)
```

TASK2: Treating categorical variables.

For this table, besides fixing any potential issue about duplicate/missing values, you are expected to explore the categorical variables: such as `highest_education`, `imd_band`, and `age_band`.

In particular, you may want to check if some categories of `highest_education`, `imd_band`, `age_band` variables (e.g., *Post Graduate Qualification* in `highest_education`) contain few instances. In such cases, you may need to merge the minority categories with a major

category and even dedice to create a new set of (fewer) categories based on the existing ones. In some cases, you may even want to decide the reduce the number of categories (if you think they are many).

As long as you provide the rationale, you can decide such details by yourself.

In [189…
```
pd.value_counts(students['highest_education'])
```

Out[189]:
```
A Level or Equivalent          13762
Lower Than A Level             12762
HE Qualification                4444
No Formal quals                  328
Post Graduate Qualification      186
Name: highest_education, dtype: int64
```

Merge HE Qualification and Post Graduate Qualification into HE and Beyond

In [190…
```
students.loc[students['highest_education'] == 'Post Graduate Qualification',
             'highest_education'] =  'HE and Beyond'
students.loc[students['highest_education'] == 'HE Qualification',
             'highest_education'] =  'HE and Beyond'
```

Merge No Formal quals and Lower Than A Level into Lower Than A Level

In [191…
```
students.loc[students['highest_education'] == 'No Formal quals',
             'highest_education'] =  'Lower Than A Level'
```

In [192…
```
pd.value_counts(students['imd_band'])
```

Out[192]:
```
20-30%     3654
30-40%     3539
10-20      3516
0-10%      3311
40-50%     3256
50-60%     3124
60-70%     2905
70-80%     2879
80-90%     2762
90-100%    2536
Name: imd_band, dtype: int64
```

In [193…
```
pd.value_counts(students['age_band'])
```

Out[193]:
```
0-35     22272
35-55     9009
55<=       201
Name: age_band, dtype: int64
```

In [194…
```
students.loc[students['age_band'] == '55<=',
             'age_band'] =  '35-55'
```

TASK3: Demographic Information and Performance Levels

More importantly for this table you are expected to explore various relationships between `final_result` and **at least three** categorical variable (e.g., did students with HE qualification perform better, did students with low `imd_band` withdraw more often, etc.). For this purpose you can visualize data and compute some basic statistics.

In [206…
```
students.groupby(["age_band", 'final_result']).count()['gender'].unstack()
```

Out[206]:

final_result	Distinction	Fail	Pass	Withdrawn
age_band				
0-35	1752	5135	8130	7255
35-55	1073	1772	3700	2665

In [207…
```
students.groupby(["imd_band", 'final_result']).count()['gender'].unstack()
```

Out[207]:

final_result imd_band	Distinction	Fail	Pass	Withdrawn
0-10%	168	916	996	1231
10-20	191	912	1167	1246
20-30%	263	844	1226	1321
30-40%	315	784	1345	1095
40-50%	293	697	1224	1042
50-60%	297	701	1227	899
60-70%	299	538	1209	859
70-80%	313	599	1170	797
80-90%	328	495	1165	774
90-100%	358	421	1101	656

In [208…

```
students.groupby(["highest_education", 'final_result']).count()['gender'].unstack()
```

Out[208]:

final_result highest_education	Distinction	Fail	Pass	Withdrawn
A Level or Equivalent	1441	2674	5658	3989
HE and Beyond	700	780	1883	1267
Lower Than A Level	684	3453	4289	4664

1.3. Registration Table

Registration table (`studentRegistration.csv`) contains information about the time when the student registered for the module presentation. For students who unregistered the date of unregistration is also recorded. File contains five columns:

- **code_module** – an identification code for a module.
- **code_presentation** - the identification code of the presentation.

- **id_student** – a unique identification number for the student.
- **date_registration** – the date of student's registration on the module presentation, this is the number of days measured relative to the start of the module-presentation (e.g. the negative value -30 means that the student registered to module presentation 30 days before it started).
- **date_unregistration** – date of student unregistration from the module presentation, this is the number of days measured relative to the start of the module-presentation. Students, who completed the course have this field empty. Students who unregistered have *Withdrawal* as the value of the `final_result` column in the `studentInfo.csv` file.

TASK1: As the first task, you need to ensure that there are no conflicts between `studentRegistration.csv` and `studentInfo.csv` dataset in terms of **Withdrawal** status of *unregistered* students. For example, if a student unregistered from a course at some point (which can be found in "studentRegistration.csv"), his/her `final_result` should be **Withdrawal**. Otherwise, it should be changed to Withdrawal.

In [261…
```
registrations = pd.read_csv('dataset/studentRegistration.csv')
registrations.shape
```

Out[261]:
```
(32593, 5)
```

In [262…
```
registrations.head(5)
```

Out[262]:

	code_module	code_presentation	id_student	date_registration	date_unregistration
0	AAA	2013J	11391	-159.0	NaN
1	AAA	2013J	28400	-53.0	NaN
2	AAA	2013J	30268	-92.0	12.0
3	AAA	2013J	31604	-52.0	NaN
4	AAA	2013J	32885	-176.0	NaN

In [263…
```
registrations = registrations.merge(students[['final_result', 'id_student']],
                                    on='id_student',
                                    how='left')
```

In [264…
```
registrations[(registrations['final_result'] != "Withdrawn") & (registrations['date_unregistration'].isnull() == False)]
```

Out[264]:

	code_module	code_presentation	id_student	date_registration	date_unregistration	final_result
16	AAA	2013J	65002	-180.0	96.0	Fail
24	AAA	2013J	94961	-170.0	72.0	Pass
50	AAA	2013J	135400	-32.0	144.0	Pass
54	AAA	2013J	141377	-110.0	129.0	Pass
63	AAA	2013J	148993	-174.0	158.0	Pass
...	...	...	...	...	...	...
38623	GGG	2013J	600320	-37.0	167.0	Pass
38680	GGG	2013J	602655	-60.0	5.0	Distinction
39415	GGG	2014B	624767	-24.0	212.0	Distinction
39490	GGG	2014B	627812	-23.0	119.0	Distinction
39587	GGG	2014B	631446	-24.0	219.0	Pass

1878 rows × 6 columns

In [265...
```
condition = (registrations['final_result'] != "Withdrawn") & (registrations['date_unregistration'].isnull() == False)
registrations.loc[condition, 'final_result'] = 'Withdrawn'
```

In [266...
```
registrations[condition]
```

Out[266]:

	code_module	code_presentation	id_student	date_registration	date_unregistration	final_result
16	AAA	2013J	65002	-180.0	96.0	Withdrawn
24	AAA	2013J	94961	-170.0	72.0	Withdrawn
50	AAA	2013J	135400	-32.0	144.0	Withdrawn
54	AAA	2013J	141377	-110.0	129.0	Withdrawn
63	AAA	2013J	148993	-174.0	158.0	Withdrawn
...	...	...	...	...	...	...
38623	GGG	2013J	600320	-37.0	167.0	Withdrawn
38680	GGG	2013J	602655	-60.0	5.0	Withdrawn
39415	GGG	2014B	624767	-24.0	212.0	Withdrawn
39490	GGG	2014B	627812	-23.0	119.0	Withdrawn
39587	GGG	2014B	631446	-24.0	219.0	Withdrawn

1878 rows × 6 columns

TASK2: Categorize students based on the day they registered for a course. In other words, you need to **bin** the registration data based on the `date_registration` column. Just to illustrate this idea, you can group students into categories such as "Very early birds", "early birds", "in-time", and "late-comers". You can use the categories given in this example or create your own categories.

In [267…
```
cats_registration = pd.cut(registrations['date_registration'],
                           bins=[registrations['date_registration'].min(), -15, 0, 7, registrations['date_registration'].max()],
                           labels=["Very early birds", "early birds", "in-time", "late-comers"])
```

In [268…
```
pd.value_counts(cats_registration)
```

Out[268]:
```
Very early birds    38653
early birds          1506
in-time               158
late-comers           129
Name: date_registration, dtype: int64
```

In [269…
```
registrations = pd.concat([registrations, cats_registration.rename('reg_type')], axis=1)
registrations.head(3)
```

Out[269]:

	code_module	code_presentation	id_student	date_registration	date_unregistration	final_result	reg_type
0	AAA	2013J	11391	-159.0	NaN	Pass	Very early birds
1	AAA	2013J	28400	-53.0	NaN	Pass	Very early birds
2	AAA	2013J	30268	-92.0	12.0	Withdrawn	Very early birds

TASK3: Categorize students based on the day they *unregistered* a course. In other words, you need to **bin** registration date based on the `date_unregistration` column. You are free to determine the number and the name of the categories (as in Task1).

In [286…
```
cats_unregistration = pd.cut(registrations['date_unregistration'],
                             bins=[registrations['date_unregistration'].min(), 3, 15, 90, registrations['date_unregistration'].max
                             labels=["Very quick leavers", "quick leavers", "late goers", "very late goers"])
```

In [287…
```
pd.value_counts(cats_unregistration)
```

Out[287]:
```
very late goers       4370
Very quick leavers    4210
late goers            3547
quick leavers         1681
Name: date_unregistration, dtype: int64
```

In [288…
```
registrations = pd.concat([registrations, cats_unregistration.rename('unreg_type')], axis=1)
registrations.head(3)
```

Out[288]:

	code_module	code_presentation	id_student	date_registration	date_unregistration	final_result	reg_type	unreg_type
0	AAA	2013J	11391	-159.0	NaN	Pass	Very early birds	NaN
1	AAA	2013J	28400	-53.0	NaN	Pass	Very early birds	NaN
2	AAA	2013J	30268	-92.0	12.0	Withdrawn	Very early birds	quick leavers

In [294…
```
registrations['unreg_type'] = registrations['unreg_type'].astype('string')
registrations['unreg_type'].fillna('Stay', inplace =True)
registrations['unreg_type'] = registrations['unreg_type'].astype('category')
registrations.head(5)
```

Out[294]:

	code_module	code_presentation	id_student	date_registration	date_unregistration	final_result	reg_type	unreg_type
0	AAA	2013J	11391	-159.0	NaN	Pass	Very early birds	Stay
1	AAA	2013J	28400	-53.0	NaN	Pass	Very early birds	Stay
2	AAA	2013J	30268	-92.0	12.0	Withdrawn	Very early birds	quick leavers
3	AAA	2013J	31604	-52.0	NaN	Pass	Very early birds	Stay
4	AAA	2013J	32885	-176.0	NaN	Pass	Very early birds	Stay

TASK4: Choose three variables from demographic data (`studentInfo.csv`), and explore if there is some relationship between students' registration/unregistration behaviour and the chosen demographic variables (e.g., did students from HE registered early? did male students unregistered sooner than female students?). You are free in exploring the data to answer similar questions that you determine. If you find no relationship, this is totally fine. Just remember that your analysis should be accompanied with meaningful interpretations.

In [296…
```
stud_reg = registrations.merge(students, on='id_student', how='inner')
```

In [301…
```
pd.crosstab(stud_reg['highest_education'], stud_reg['reg_type'])
```

Out[301]:

reg_type	Very early birds	early birds	in-time	late-comers
highest_education				
A Level or Equivalent	24415	786	67	104
HE and Beyond	8703	300	29	25
Lower Than A Level	21570	852	94	92

In [300…
```
pd.crosstab(stud_reg['highest_education'], stud_reg['unreg_type'])
```

Out[300]:

unreg_type highest_education	Stay	Very quick leavers	late goers	quick leavers	very late goers
A Level or Equivalent	16177	2723	2346	1039	3129
HE and Beyond	5895	956	768	262	1195
Lower Than A Level	12767	2840	2698	1316	3033

In [302…
```
pd.crosstab(stud_reg['age_band'], stud_reg['unreg_type'])
```

Out[302]:

unreg_type age_band	Stay	Very quick leavers	late goers	quick leavers	very late goers
0-35	24999	4863	4278	1980	5216
35-55	9840	1656	1534	637	2141

In [303…
```
pd.crosstab(stud_reg['imd_band'], stud_reg['unreg_type'])
```

Out[303]:

unreg_type imd_band	Stay	Very quick leavers	late goers	quick leavers	very late goers
0-10%	3246	741	608	372	834
10-20	3504	730	672	366	752
20-30%	3807	818	857	386	984
30-40%	3973	687	584	354	779
40-50%	3509	794	612	294	669
50-60%	3509	638	528	173	658
60-70%	3359	524	571	213	710
70-80%	3452	515	473	166	781
80-90%	3358	618	471	175	630
90-100%	3122	454	436	118	560

1.4. Course Components Table

Course components table (`moodle.csv`) contains information about the available materials in the Moodle LMS. Typically these are html pages, pdf files, etc. Students have access to these materials online and their interactions with the materials are recorded. The `moodle.csv` file contains the following columns:

- **id_site** – an identification number of the material.
- **code_module** – an identification code for module.
- **code_presentation** - the identification code of presentation.
- **activity_type** – the role associated with the module material.
- **week_from** – the week from which the material is planned to be used.
- **week_to** – week until which the material is planned to be used.

TASK1: In this dataset, some columns contain mainly missing values. Detect them and drop them to save space in the memory.

In [305…
```
components = pd.read_csv('dataset/moodle.csv')
components.shape
```

Out[305]:
```
(6364, 6)
```

In [306…
```
components.head(3)
```

Out[306]:

	id_site	code_module	code_presentation	activity_type	week_from	week_to
0	546943	AAA	2013J	resource	NaN	NaN
1	546712	AAA	2013J	oucontent	NaN	NaN
2	546998	AAA	2013J	resource	NaN	NaN

In [307…
```
components.isnull().sum()
```

Out[307]:
```
id_site                0
code_module            0
code_presentation      0
activity_type          0
week_from           5243
week_to             5243
dtype: int64
```

In [309…
```
components.drop(['week_from', 'week_to'], axis=1, inplace=True)
```

TASK2: First identify the top 5 popular course component (`activity_type`) across all courses. Then, create a new table that displays how many times each of these popular components were included in each offering (`code_presentation`) of each course (`code_module`). Briefly interpret this table.

In [319…
```
top_comp = components.groupby('activity_type').count().sort_values('id_site', ascending=False).reset_index().head(5)['activity_ty
top_comp
```

Out[319]:
```
0     resource
1      subpage
2    oucontent
3          url
4      forumng
Name: activity_type, dtype: object
```

In [322…
```
components_bytop = components[components['activity_type'].isin(top_comp)]
components_bytop
```

Out[322]:

	id_site	code_module	code_presentation	activity_type
0	546943	AAA	2013J	resource
1	546712	AAA	2013J	oucontent
2	546998	AAA	2013J	resource
3	546888	AAA	2013J	url
4	547035	AAA	2013J	resource
...	...	...	...	...
6359	897063	GGG	2014J	resource
6360	897109	GGG	2014J	resource
6361	896965	GGG	2014J	oucontent
6362	897060	GGG	2014J	resource
6363	897100	GGG	2014J	resource

5791 rows × 4 columns

In [331…

```
components_bytop = components_bytop.groupby(['code_module', 'code_presentation', 'activity_type']).count()
components_bytop = components_bytop.rename(columns={'id_site':'count'})
components_bytop
```

Out[331]:

			count
code_module	code_presentation	activity_type	
AAA	2013J	forumng	15
		oucontent	68
		resource	95
		subpage	6
		url	18
...	...	...	...
GGG	2014B	subpage	5
	2014J	forumng	2
		oucontent	26
		resource	63
		subpage	5

107 rows × 1 columns

In [332…

```
components_bytop.pivot_table(index=['code_module', 'code_presentation'], columns='activity_type', values='count')
```

Out[332]:

code_module	code_presentation	activity_type	forumng	oucontent	resource	subpage	url
AAA	2013J		15.0	68.0	95.0	6.0	18.0
	2014J		6.0	68.0	93.0	6.0	20.0
BBB	2013B		17.0	1.0	236.0	37.0	15.0
	2013J		19.0	3.0	236.0	38.0	15.0
	2014B		17.0	3.0	231.0	37.0	14.0
	2014J		3.0	70.0	104.0	10.0	6.0
CCC	2014B		9.0	47.0	78.0	28.0	13.0
	2014J		9.0	58.0	85.0	31.0	19.0
DDD	2013B		11.0	6.0	182.0	114.0	92.0
	2013J		16.0	13.0	178.0	194.0	44.0
	2014B		13.0	13.0	177.0	193.0	41.0
	2014J		13.0	13.0	169.0	110.0	46.0
EEE	2013J		5.0	48.0	34.0	7.0	7.0
	2014B		5.0	46.0	32.0	7.0	7.0
	2014J		5.0	45.0	37.0	7.0	7.0
FFF	2013B		5.0	108.0	129.0	55.0	125.0
	2013J		7.0	107.0	136.0	55.0	132.0
	2014B		6.0	101.0	109.0	53.0	133.0
	2014J		7.0	103.0	80.0	52.0	132.0
GGG	2013J		2.0	24.0	95.0	5.0	NaN
	2014B		2.0	25.0	81.0	5.0	NaN
	2014J		2.0	26.0	63.0	5.0	NaN

1.5. Student Activity Data

Student activity data (`studentMoodleInteract.csv`) contains information about each student's interactions with the materials in the VLE. This file contains the following columns:

- **code_module** – an identification code for a module.
- **code_presentation** - the identification code of the module presentation.
- **id_student** – a unique identification number for the student.
- **id_site** - an identification number for the course material/component.
- **date** – the date of student's interaction with the material measured as the number of days since the start of the module-presentation.
- **sum_click** – the number of times a student interacts with the material in that day.

In [333…
```
activity = pd.read_csv('dataset/studentMoodleInteract.csv')
activity.head(5)
```

Out[333]:

	code_module	code_presentation	id_student	id_site	date	sum_click
0	AAA	2013J	28400	546652	-10	4
1	AAA	2013J	28400	546652	-10	1
2	AAA	2013J	28400	546652	-10	1
3	AAA	2013J	28400	546614	-10	11
4	AAA	2013J	28400	546714	-10	1

TASK1: Display the total number of clicks for each course per each semester delivered. Besides a textual output, some visualizations might be very helpful for interpreting the data.

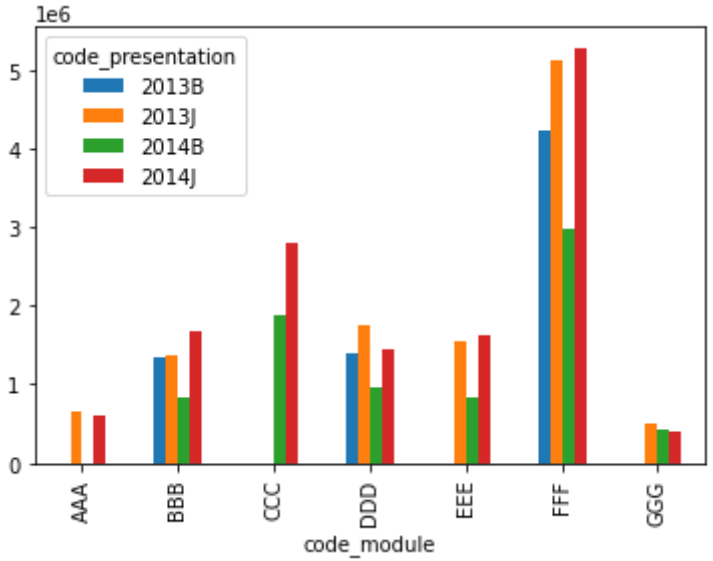
In [342…
```
activityPVT = activity.pivot_table(index='code_module', columns='code_presentation', values='sum_click', aggfunc='sum').fillna(0
activityPVT.head(10)
```

Out[342]:

code_module	2013B	2013J	2014B	2014J
AAA	0.0	648494.0	0.0	598158.0
BBB	1347911.0	1378656.0	833865.0	1673744.0
CCC	0.0	0.0	1889177.0	2792972.0
DDD	1387467.0	1757214.0	947657.0	1437751.0
EEE	0.0	1535953.0	832038.0	1616236.0
FFF	4220055.0	5116439.0	2975619.0	5281809.0
GGG	0.0	509091.0	425165.0	399628.0

In [343…
```
activityPVT.plot(kind='bar')
```

Out[343]:
```
<AxesSubplot:xlabel='code_module'>
```



TASK2: As a follow up to the first task, identify the courses in which the total number of clicks is higher in 2014 than 2013. If the course was taught two times in the same year (such as, 2013B and 2013J) use the average of both semesters (`(2013B+2013J)/2`) to compare with the other year.

In [364…]:
```
activityPVT = activityPVT.reset_index()
activityPVT.head()
```

Out[364]:

code_presentation	code_module	2013B	2013J	2014B	2014J
0	AAA	0.0	648494.0	0.0	598158.0
1	BBB	1347911.0	1378656.0	833865.0	1673744.0
2	CCC	0.0	0.0	1889177.0	2792972.0
3	DDD	1387467.0	1757214.0	947657.0	1437751.0
4	EEE	0.0	1535953.0	832038.0	1616236.0

In [370…]:
```
activityPVT = activityPVT.reset_index().assign(AVG_2013=(activityPVT['2013B']+ activityPVT['2013J'])/2)
activityPVT = activityPVT.reset_index().assign(AVG_2014=(activityPVT['2014B']+ activityPVT['2014J'])/2)
activityPVT
```

Out[370]:

code_presentation	level_0	index	code_module	2013B	2013J	2014B	2014J	AVG_2013	AVG_2014
0	0	0	AAA	0.0	648494.0	0.0	598158.0	324247.0	299079.0
1	1	1	BBB	1347911.0	1378656.0	833865.0	1673744.0	1363283.5	1253804.5
2	2	2	CCC	0.0	0.0	1889177.0	2792972.0	0.0	2341074.5
3	3	3	DDD	1387467.0	1757214.0	947657.0	1437751.0	1572340.5	1192704.0
4	4	4	EEE	0.0	1535953.0	832038.0	1616236.0	767976.5	1224137.0
5	5	5	FFF	4220055.0	5116439.0	2975619.0	5281809.0	4668247.0	4128714.0
6	6	6	GGG	0.0	509091.0	425165.0	399628.0	254545.5	412396.5

In [371…]:
```
activityPVT[activityPVT['AVG_2014']>activityPVT['AVG_2013']]
```

Out[371]:

code_presentation	level_0	index	code_module	2013B	2013J	2014B	2014J	AVG_2013	AVG_2014
2	2	2	CCC	0.0	0.0	1889177.0	2792972.0	0.0	2341074.5
4	4	4	EEE	0.0	1535953.0	832038.0	1616236.0	767976.5	1224137.0
6	6	6	GGG	0.0	509091.0	425165.0	399628.0	254545.5	412396.5

TASK3: Which type of resources were mostly clicked by the students? Do you observe a common pattern accross courses (e.g., in almost all courses, clicks on `resource` is higher than `quiz`)?

In [377…

```
resourcePVT = activity.pivot_table(index='id_site', columns='code_module', values='sum_click', aggfunc='sum').reset_index().fill
resourcePVT.head()
```

Out[377]:

code_module	id_site	AAA	BBB	CCC	DDD	EEE	FFF	GGG
0	526721	0.0	0.0	0.0	0.0	0.0	586632.0	0.0
1	526735	0.0	0.0	0.0	0.0	0.0	27693.0	0.0
2	526737	0.0	0.0	0.0	0.0	0.0	152414.0	0.0
3	526738	0.0	0.0	0.0	0.0	0.0	366998.0	0.0
4	526739	0.0	0.0	0.0	0.0	0.0	21315.0	0.0

In [380…

```
components.columns
```

Out[380]:

```
Index(['id_site', 'code_module', 'code_presentation', 'activity_type'], dtype='object')
```

In [381…

```
resourcePVT.columns
```

Out[381]:

```
Index(['id_site', 'AAA', 'BBB', 'CCC', 'DDD', 'EEE', 'FFF', 'GGG'], dtype='object', name='code_module')
```

In [383…

```
resourcePVT = resourcePVT.merge(components[['id_site', 'activity_type']], on='id_site', how='left')
resourcePVT.head(5)
```

Out[383]:

	id_site	AAA	BBB	CCC	DDD	EEE	FFF	GGG	activity_type
0	526721	0.0	0.0	0.0	0.0	0.0	586632.0	0.0	homepage
1	526735	0.0	0.0	0.0	0.0	0.0	27693.0	0.0	forumng
2	526737	0.0	0.0	0.0	0.0	0.0	152414.0	0.0	forumng
3	526738	0.0	0.0	0.0	0.0	0.0	366998.0	0.0	forumng
4	526739	0.0	0.0	0.0	0.0	0.0	21315.0	0.0	forumng

In [399…

```
resourcePVT = resourcePVT.groupby('activity_type').sum().drop('id_site',axis=1)
resourcePVT
```

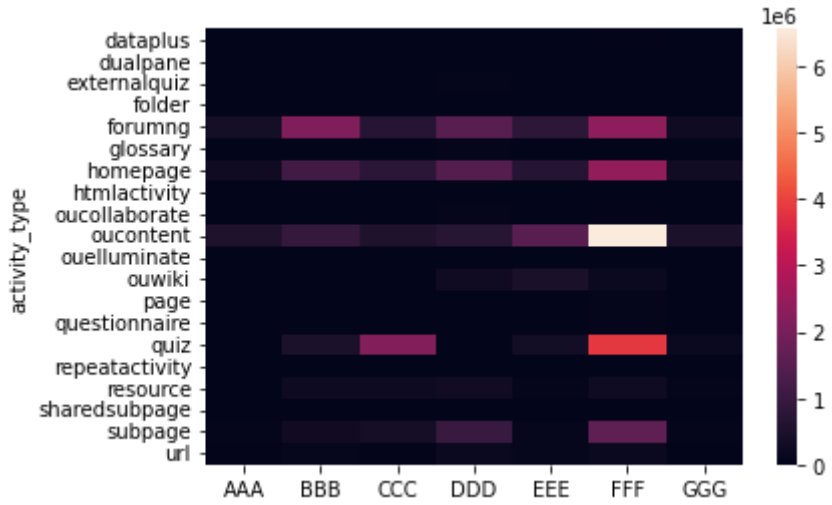
Out[399]:

activity_type	AAA	BBB	CCC	DDD	EEE	FFF	GGG
dataplus	3398.0	0.0	0.0	0.0	0.0	44070.0	0.0
dualpane	0.0	0.0	0.0	0.0	4871.0	15845.0	0.0
externalquiz	0.0	0.0	0.0	64292.0	0.0	0.0	0.0
folder	0.0	0.0	0.0	0.0	0.0	5420.0	0.0
forumng	337103.0	2109720.0	643094.0	1493375.0	792656.0	2358160.0	239282.0
glossary	656.0	9820.0	0.0	63336.0	0.0	6142.0	8008.0
homepage	266720.0	1136792.0	756032.0	1433206.0	654525.0	2418335.0	283454.0
htmlactivity	0.0	0.0	0.0	0.0	0.0	9239.0	0.0
oucollaborate	498.0	12556.0	9822.0	53390.0	6920.0	25788.0	0.0
oucontent	523394.0	921260.0	516502.0	686331.0	1514727.0	6567216.0	477373.0
ouelluminate	0.0	1433.0	0.0	13565.0	0.0	24030.0	0.0
ouwiki	0.0	0.0	0.0	275867.0	457939.0	160706.0	0.0
page	0.0	0.0	4188.0	1536.0	879.0	57028.0	0.0
questionnaire	0.0	6730.0	0.0	0.0	0.0	58034.0	0.0
quiz	0.0	462995.0	2164110.0	0.0	322772.0	3831026.0	200337.0
repeatactivity	0.0	0.0	0.0	0.0	0.0	9.0	0.0
resource	14924.0	213605.0	217846.0	301644.0	52423.0	237075.0	72615.0
sharedsubpage	0.0	171.0	0.0	0.0	0.0	0.0	0.0
subpage	71582.0	273600.0	342574.0	979982.0	83270.0	1607759.0	52815.0
url	28377.0	85494.0	27981.0	163565.0	93245.0	168040.0	0.0

In [401…

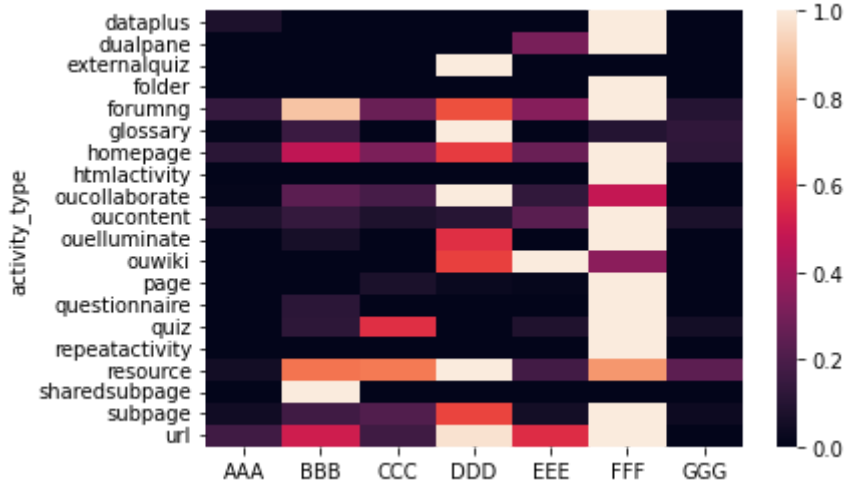
```
import seaborn as sns
sns.heatmap(resourcePVT)
```

Out[401]: <AxesSubplot:ylabel='activity_type'>



In [405…
```
sns.heatmap(resourcePVT.div(resourcePVT.max(axis=1), axis=0))
```

Out[405]: <AxesSubplot:ylabel='activity_type'>



TASK4: For each student, compute the total number of clicks per each course component type (`activity_type` column in `moodle.csv`) separately for each course and semester. A simple representation of the expected table is provided below with fake data (note that in the given example columns and rows are incomplete).

Student Id	code_module	code_presentation	PDF	Assignment
1234	AAA	2013J	23	33
1234	BBB	2014B	5	42

Note that, in this task you actually create some features that can be used for predictive modeling.

In [419…
```
activityPVT2 = activity.groupby(['id_student', 'code_module', 'code_presentation', 'id_site'])
activityPVT2 = activityPVT2.sum().rename(columns={'date':'clicks'})[['clicks']].reset_index()
activityPVT2.head()
```

Out[419]:

	id_student	code_module	code_presentation	id_site	clicks
0	6516	AAA	2014J	877011	1262
1	6516	AAA	2014J	877012	11990
2	6516	AAA	2014J	877015	1475
3	6516	AAA	2014J	877023	368
4	6516	AAA	2014J	877025	864

In [420…
```
activityPVT2 = activityPVT2.merge(components[['id_site', 'activity_type']], on='id_site', how='left')
activityPVT2.head(5)
```

Out[420]:

	id_student	code_module	code_presentation	id_site	clicks	activity_type
0	6516	AAA	2014J	877011	1262	forumng
1	6516	AAA	2014J	877012	11990	forumng
2	6516	AAA	2014J	877015	1475	forumng
3	6516	AAA	2014J	877023	368	forumng
4	6516	AAA	2014J	877025	864	forumng

In [440…
```
activityPVT22 = activityPVT2.pivot_table(index=['id_student', 'code_module', 'code_presentation'],
                                         columns='activity_type',
                                         values='clicks').reset_index()
```

```
activityPVT22 = activityPVT22.sort_values(['id_student', 'code_module', 'code_presentation'])
activityPVT22.head()
```

Out[440]:

	activity_type	id_student	code_module	code_presentation	dataplus	dualpane	externalquiz	folder	forumng	glossary	homepage	...	ouellumin
0		6516	AAA	2014J	174.666667	NaN	NaN	NaN	3191.800000	NaN	18707.0	...	N
1		8462	DDD	2013J	NaN	NaN	156.666667	NaN	151.666667	NaN	2513.0	...	N
2		8462	DDD	2014J	NaN	NaN	NaN	NaN	20.000000	NaN	10.0	...	N
3		11391	AAA	2013J	NaN	NaN	NaN	NaN	380.090909	NaN	3872.0	...	N
4		23629	BBB	2013B	NaN	NaN	NaN	NaN	217.250000	NaN	662.0	...	N

5 rows × 23 columns

TASK5: Using proper visualizations and statistical analysis, please explore if there is any relationship between students' course performance (`final_result` column in `studentInfo.csv`) and clicks on different resources.

In [513…

```
reg_clicks = activityPVT22.merge(registrations[['id_student', 'final_result', 'code_module', 'code_presentation']], on=['id_stude
reg_clicks.head()
```

Out[513]:

	id_student	code_module	code_presentation	dataplus	dualpane	externalquiz	folder	forumng	glossary	homepage	...	ouwiki	page	questi
0	6516	AAA	2014J	174.666667	NaN	NaN	NaN	3191.800000	NaN	18707.0	...	NaN	NaN	
1	8462	DDD	2013J	NaN	NaN	156.666667	NaN	151.666667	NaN	2513.0	...	186.0	NaN	
3	8462	DDD	2014J	NaN	NaN	NaN	NaN	20.000000	NaN	10.0	...	NaN	NaN	
5	11391	AAA	2013J	NaN	NaN	NaN	NaN	380.090909	NaN	3872.0	...	NaN	NaN	
6	23629	BBB	2013B	NaN	NaN	NaN	NaN	217.250000	NaN	662.0	...	NaN	NaN	

5 rows × 24 columns

In [514…

```
reg_clicks = reg_clicks.dropna(thresh=reg_clicks.shape[0]*.80, how='all', axis=1).round(2).fillna(0)
reg_clicks.head(10)
```

Out[514]:

	id_student	code_module	code_presentation	forumng	homepage	oucontent	resource	subpage	url	final_result
0	6516	AAA	2014J	3191.80	18707.0	444.42	117.40	1685.80	1107.8	Pass
1	8462	DDD	2013J	151.67	2513.0	84.62	54.56	75.04	37.4	Withdrawn
3	8462	DDD	2014J	20.00	10.0	0.00	0.00	0.00	0.0	Withdrawn
5	11391	AAA	2013J	380.09	3872.0	302.38	163.80	235.50	264.0	Pass
6	23629	BBB	2013B	217.25	662.0	0.00	4.00	4.00	0.0	Fail
7	23698	CCC	2014J	576.00	5432.0	65.67	264.30	539.33	120.0	Pass
8	23798	BBB	2013J	986.67	8730.0	245.50	170.25	273.43	1827.0	Distinction
9	24186	GGG	2014B	460.00	2552.0	156.00	84.43	213.00	0.0	Pass
10	24213	DDD	2014B	3630.00	13076.0	622.89	103.82	529.30	301.8	Withdrawn
11	24213	DDD	2014B	3630.00	13076.0	622.89	103.82	529.30	301.8	Pass

In [515…

```
reg_clicks2 = reg_clicks.drop('id_student', axis=1).groupby('final_result').mean().round(2).reset_index()
reg_clicks2.head(5)
```

Out[515]:

	final_result	forumng	homepage	oucontent	resource	subpage	url
0	0	2930.21	10664.84	366.35	157.53	491.65	303.16
1	Distinction	3828.39	11591.07	379.27	178.41	584.43	368.79
2	Fail	617.60	2743.61	165.70	78.29	199.65	107.17
3	Pass	2330.22	8896.98	367.28	172.49	510.87	317.63
4	Withdrawn	557.44	1980.80	116.01	52.97	132.82	91.87

In [516…

```
from scipy.stats import ttest_ind

cat1 = reg_clicks[reg_clicks['final_result']=='Pass']
cat2 = reg_clicks[reg_clicks['final_result']=='Fail']

ttest_ind(cat1['forumng'], cat2['forumng'])
```

Out[516]:

```
Ttest_indResult(statistic=44.37580420071086, pvalue=0.0)
```

2. Predictive Modeling

In this section, you will build a machine learning model to predict students' final course outcome (`final_result` column in `studentInfo.csv`). That is, whether student is 'Pass', 'Withdrawn', 'Fail', or 'Distinction'. If you consider the number of students in some of these categories are too few, you can combine them into a new category.

2.1. Generate Features from Demographic Information

In Section 1.2, you explored demographic data about students and tuned some categorical variables. From these categorical variables, please generate *at least* **10** *dummy* variables to be used for predictors in the machine learning model.

In [480…
```
students.head()
```

Out[480]:

	code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_prev_attempts	studied_credits	disal
0	AAA	2013J	11391	M	East Anglian Region	HE and Beyond	90-100%	35-55	0	240	
1	AAA	2013J	28400	F	Scotland	HE and Beyond	20-30%	35-55	0	60	
2	AAA	2013J	30268	F	North Western Region	A Level or Equivalent	30-40%	35-55	0	60	
3	AAA	2013J	31604	F	South East Region	A Level or Equivalent	50-60%	35-55	0	60	
4	AAA	2013J	32885	F	West Midlands Region	Lower Than A Level	50-60%	0-35	0	60	

In [495…
```
features_demog = pd.get_dummies(students.drop(['code_module', 'code_presentation', 'imd_band',
                                               'num_of_prev_attempts', 'studied_credits', 'final_result'],
                                              axis=1),
```

```
                                    columns=['age_band', 'highest_education', 'region', 'gender', 'disability'], drop_first=True)
features_demog.head()
```

Out[495]:

	id_student	age_band_35-55	highest_education_HE and Beyond	highest_education_Lower Than A Level	region_East Midlands Region	region_Ireland	region_London Region	region_North Region	region_North Western Region
0	11391	1	1	0	0	0	0	0	0
1	28400	1	1	0	0	0	0	0	0
2	30268	1	0	0	0	0	0	0	1
3	31604	1	0	0	0	0	0	0	0
4	32885	0	0	1	0	0	0	0	0

2.2. Generate/Select Features from Click Data

In Section 1.5, you have already created some features from students' click behaviour. You can use all of them here as additional predictors.

Additionally, you should create *at least* **3** features indicating the engagement level of students at different course components. Some example features are provided below :

- a dummy variable that indicates if students clicked at least three types of course components or not,
- each student's average number of clicks across all components per a single course and semester,
- a dummy variable indicating if students clicked all types of course components.

There is no limit in the type and number of additional feature you can generate from the click data.

In [517…

```
reg_clicks.columns
```

Out[517]:

```
Index(['id_student', 'code_module', 'code_presentation', 'forumng', 'homepage',
       'oucontent', 'resource', 'subpage', 'url', 'final_result'],
      dtype='object')
```

In [518…

```
mean_click_comp = reg_clicks.iloc[:, 3:9].mean(axis=0)
mean_click_comp
```

Out[518]:
```
forumng       1637.607775
homepage      5990.113264
oucontent      256.355488
resource       119.708360
subpage        347.106230
url            215.162265
dtype: float64
```

In [521…
```
reg_clicksx = reg_clicks.assign(aboveforumng=(reg_clicks.forumng>mean_click_comp.forumng).astype(int))
reg_clicksx = reg_clicksx.assign(abovehomepage=(reg_clicksx.homepage>mean_click_comp.homepage).astype(int))
reg_clicksx = reg_clicksx.assign(aboveoucontent=(reg_clicksx.oucontent>mean_click_comp.oucontent).astype(int))
reg_clicksx = reg_clicksx.assign(aboveresource=(reg_clicksx.resource>mean_click_comp.resource).astype(int))
reg_clicksx = reg_clicksx.assign(abovesubpage=(reg_clicksx.subpage>mean_click_comp.subpage).astype(int))
reg_clicksx = reg_clicksx.assign(aboveurl=(reg_clicksx.url>mean_click_comp.url).astype(int))
reg_clicksx.head()
```

Out[521]:

	id_student	code_module	code_presentation	forumng	homepage	oucontent	resource	subpage	url	final_result	aboveforumng	abovehomepag
0	6516	AAA	2014J	3191.80	18707.0	444.42	117.40	1685.80	1107.8	Pass	1	
1	8462	DDD	2013J	151.67	2513.0	84.62	54.56	75.04	37.4	Withdrawn	0	
3	8462	DDD	2014J	20.00	10.0	0.00	0.00	0.00	0.0	Withdrawn	0	
5	11391	AAA	2013J	380.09	3872.0	302.38	163.80	235.50	264.0	Pass	0	
6	23629	BBB	2013B	217.25	662.0	0.00	4.00	4.00	0.0	Fail	0	

In [529…
```
features_clicks = reg_clicksx.drop(['code_module', 'code_presentation'], axis=1)
features_clicks.head()
```

Out[529]:

	id_student	forumng	homepage	oucontent	resource	subpage	url	final_result	aboveforumng	abovehomepage	aboveoucontent	aboveresourc
0	6516	3191.80	18707.0	444.42	117.40	1685.80	1107.8	Pass	1	1	1	
1	8462	151.67	2513.0	84.62	54.56	75.04	37.4	Withdrawn	0	0	0	
3	8462	20.00	10.0	0.00	0.00	0.00	0.0	Withdrawn	0	0	0	
5	11391	380.09	3872.0	302.38	163.80	235.50	264.0	Pass	0	0	1	
6	23629	217.25	662.0	0.00	4.00	4.00	0.0	Fail	0	0	0	

2.3. Training and Testing the Model

As the last activity in this project, you are expected to train and test a logistic regression model for predicting students' final course status. You should use 10-fold cross-validation.

Interpret the results based on confusion matrix and AUC scores. In your interpretation, please also mention the features with high predictive power and those with low predictive power.

Please note that the achiving low/high accuracy in the predictions has no importance for your grade.

In [533…
```
df = features_clicks.merge(features_demog, on='id_student', how='inner')
df.columns
```

Out[533]:
```
Index(['id_student', 'forumng', 'homepage', 'oucontent', 'resource', 'subpage',
       'url', 'final_result', 'aboveforumng', 'abovehomepage',
       'aboveoucontent', 'aboveresource', 'abovesubpage', 'aboveurl',
       'age_band_35-55', 'highest_education_HE and Beyond',
       'highest_education_Lower Than A Level', 'region_East Midlands Region',
       'region_Ireland', 'region_London Region', 'region_North Region',
       'region_North Western Region', 'region_Scotland',
       'region_South East Region', 'region_South Region',
       'region_South West Region', 'region_Wales',
       'region_West Midlands Region', 'region_Yorkshire Region', 'gender_M',
       'disability_Y'],
      dtype='object')
```

In [558…
```
pd.value_counts(df.final_result)
```

Out[558]:
```
Pass           14986
Withdrawn      13128
Fail            8420
Distinction     3809
Name: final_result, dtype: int64
```

In [592…
```
df.loc[df['final_result'] == 'Distinction', 'final_result'] = 'Pass'
df.loc[df['final_result'] == 'Withdrawn', 'final_result'] = 'Fail'

df.loc[df['final_result'] == 'Pass', 'final_result'] = 1
df.loc[df['final_result'] == 'Fail', 'final_result'] = 0
df['final_result'] = df['final_result'].astype('int')
```

In [593…
```
pd.value_counts(df.final_result)
```

Out[593]:
```
0    21548
1    18795
Name: final_result, dtype: int64
```

In [594…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [615…
```
x_train, x_test, y_train, y_test = train_test_split(df.drop(columns=['final_result', 'id_student']), df.loc[:,'final_result'],
                                                    shuffle=True, train_size=.8, test_size=.2)
```

In [602…
```
model = LogisticRegression(penalty='l1', max_iter=3000, solver='liblinear', multi_class='auto', tol=0.05)
model.fit(X=x_train, y=y_train)
```

Out[602]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=3000, multi_class='auto',
          n_jobs=None, penalty='l1', random_state=None, solver='liblinear',
          tol=0.05, verbose=0, warm_start=False)
```

In [603…
```
cross_val_score(model, x_test, y_test, cv=10)
```

Out[603]:
```
array([0.83663366, 0.80693069, 0.79579208, 0.81164808, 0.80793061,
       0.78810409, 0.80521092, 0.82009926, 0.79156328, 0.79776675])
```

In [605…
```
y_pred = cross_val_predict(model, x_test, y_test, cv=10)
conf_mat = confusion_matrix(y_test, y_pred)
conf_mat
```

Out[605]:
```
array([[3404,  819],
       [ 750, 3096]], dtype=int64)
```

In [606…
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, y_pred)
```

Out[606]:
```
0.8055271204454619
```

In [614…
```
feature_importance = pd.DataFrame()
feature_importance['coef_'] = pd.DataFrame(data={'coef_': model.coef_[0]})[['coef_']]
```

```
feature_importance['feature'] = df.drop(columns=['final_result', 'id_student']).columns
feature_importance.sort_values('coef_')
```

Out[614]:

	coef_	feature
27	-0.668152	gender_M
17	-0.530082	region_London Region
28	-0.497568	disability_Y
19	-0.435419	region_North Western Region
18	-0.429398	region_North Region
20	-0.412385	region_Scotland
14	-0.380922	highest_education_Lower Than A Level
25	-0.366393	region_West Midlands Region
15	-0.258286	region_East Midlands Region
26	-0.256548	region_Yorkshire Region
24	-0.195459	region_Wales
16	-0.125925	region_Ireland
21	-0.123437	region_South East Region
12	-0.106732	age_band_35-55
22	-0.096683	region_South Region
2	-0.000626	oucontent
5	-0.000183	url
0	-0.000076	forumng
1	0.000177	homepage
4	0.000560	subpage
3	0.001787	resource
23	0.016133	region_South West Region
13	0.026489	highest_education_HE and Beyond
11	0.091405	aboveurl

	coef_	feature
6	0.160063	aboveforumng
7	0.259045	abovehomepage
10	0.436259	abovesubpage
9	0.609836	aboveresource
8	0.651557	aboveoucontent

In [ ]:

In [ ]:

In [ ]: