INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE TOULOUSE

Internship Report

Analytics and NLP on Student Engagement Dashboard data

- June 07 to August 31, 2022 -

ADAPT RESEARCH CENTRE

Dublin City University, Collins Ave Ext, Whitehall, Dublin 9
Dublin, IRELAND

Lila Roig

Specialty:
Applied Mathematics, Year 4

Supervisor:
Dr Annalina Caputo

Submission date:
October 30, 2022

Contents

I Introduction . . . 5
II Brief description of ADAPT Centre . . . 6
III Context and purpose of the internship . . . 6
IV Dataset description . . . 7
V Step 1: Analysis of non-textual data . . . 8
V.1 Dataset cleaning . . . 8
V.2 Distribution of students in the different modules . . . 8
V.3 Distribution of marks . . . 9
V.4 Distribution and influence of the progress on the mark . . . 10
V.5 Distribution and influence of the meetings' duration on the mark . . . 11
V.6 Understanding supervisors behaviour . . . 11
V.7 Implementation of interactive panels to facilitate visualisation . . . 12
VI Step 2: Sentiment analysis of the comments using NLP algorithms . . . 14
VI.1 A first naive approach with unsupervised NLP algorithms . . . 15
VI.1.1 Sentiment analysis with TextBlob . . . 15
VI.1.2 Sentiment analysis with VADER . . . 16
VI.2 A second approach based on supervised state-of-the-art algorithm BERT . . . 16
VI.2.1 Overview of the model . . . 16
VI.2.2 Text preprocessing . . . 18
VI.2.3 Formatting training and validation datasets . . . 19
VI.2.4 Fine-tuning the model . . . 19
VI.2.5 Predictions on the validation set . . . 20
VI.2.6 Training the model on the entire dataset . . . 21
VI.2.7 Two-phase fine-tuning and labeled datasets . . . 21
VI.3 Evaluation Results . . . 22
VI.3.1 TextBlob and VADER results . . . 22
VI.3.2 BERT results . . . 23
VII Step 3: Prediction of students' marks . . . 24
VII.1 Creation of a new dataset . . . 24
VII.2 Machine learning algorithm for prediction . . . 25
VII.2.1 Multinomial logistic regression . . . 25
VII.2.2 Feedforward neural network . . . 26
VII.3 Mark Classification Results . . . 27

VIII Critical review of the results obtained and areas for improvement 29
IX Conclusion 30

A Appendices **31**
I First descriptive analysis on non-textual data 31
II Prediction of students' marks 31

Acknowledgements

I would like to thank all the people who contributed to the success of my internship.

First of all, I express my thanks and gratitude to my supervisor Dr Annalina Caputo, Assistant Professor in the School of Computing at DCU, for allowing me to realise this internship by welcoming me and including me in her team. I thank her for guiding me with accuracy and kindness throughout my internship and for making me feel at home at DCU.

I would also like to thank Dr Andrew McCarren, Assistant Professor in the School of Computing at DCU, for his advice on the methods and models to be used.

I thank DCU and ADAPT Centre for welcoming me in their facilities and for organising social events to exchange with other students and researchers, which was a very enriching experience for me. I also express my warmest thanks to all the PhD students, interns and post-doctoral fellows I met at DCU, who created a positive and energetic working environment. I was able to spend wonderful moments with them, allowing me to learn about the culture and the world of research.

Finally, I would like to thank Philippe Villedieu, lecturer in the GMM department and in charge of internships at INSA, for helping me with the administrative procedures for the establishment of the internship agreement. I thank Juliette Chevallier, lecturer in the GMM department, for having accepted to supervise this internship from my home university.

I Introduction

This internship took place from June to August 2022 at the ADAPT Center, a world-leading Science Foundation Ireland Research Centre, for AI-Driven Digital Content Technology. The ADAPT Centre I worked in is located in Dublin City University (DCU), in the north of Dublin in The Republic Of Ireland.

The internship was supervised by Dr Annalina Caputo, Assistant Professor in the School of Computing at Dublin City University and Academic Lead for the MSc in Artificial Intelligence. Dr Caputo is also a funded investigator in the ADAPT 2 and I-Form centres. Her main research interests are in Natural Language Processing, Information Access and Retrieval, and Machine Learning [1].

The internship consisted of working on a dataset extracted from The Student Dashboard, an IT platform maintained by the School of Computing at DCU to supervise their students during the realisation of their final year projects. The internship project was structured in three parts: first, a descriptive analysis of the dashboard data was performed in order to get an insight into the information contained in the dashboard. Secondly, Natural Language Processing (NLP) methods were implemented to determine the sentiment of the texts contained in the dashboard. Finally, the final mark of the students for their final year project was predicted, using machine learning algorithms and features extracted during the first two parts of this project. One of the main challenges of this internship was to apply NLP algorithms to the dashboard text data, which is different from the usual datasets used for NLP.

The internship report will be structured as follows. First, we will present in more detail the missions and objectives of this internship in section II and briefly describe the research centre where the internship took place in section III. Then, after describing the dataset in section IV, we will detail in three parts the realised tasks. We will start by presenting the descriptive analysis of the non-textual data extracted from the dashboard in section V. Then, we will explain in section VI the NLP algorithms used to determine the main sentiment of the texts from the dashboard and present the results obtained in section VI.3. Finally, we will discuss the machine learning methods used to predict students' marks in section VII and present the results in section VII.3. We will have a critical look at the results and suggest ways of improving our model in section VIII and conclude this internship report in section IX.

[1] https://annalina.github.io/

II Brief description of ADAPT Centre

ADAPT Centre is a research centre for AI-Driven Digital Content Technology funded by the Irish Government through Science Foundation Ireland, the industry and the competitive research funding (EU). It is coordinated by Trinity College Dublin and co-hosted by Dublin City University. ADAPT combines the expertise of 300 researchers at 8 Irish higher education institutions with that of more than 140 industry partners in Ireland and beyond, ranging from start-ups to multinational enterprises [1].

ADAPT Centre is pioneering new Human Centric AI techniques and technologies and its research topics include content analytics and knowledge extraction, personalisation, natural language processing, intelligent machine translation and human-computer interaction, dynamic personalised digital media discovery, as well as setting the standards for data governance, quality, privacy, and ethics for digital content [2].

ADAPT conducts interdisciplinary research by integrating experts in the complementary fields of social sciences, communication, financial technologies, ethics, law, health, environment and sustainable development.

Finally, ADAPT's technology and research is applied in fields such as information and communication technology, localisation, financial services, eCommerce, eHealth, media, entertainment and games, life sciences, digital culture and humanities, and education.

III Context and purpose of the internship

At the School of Computing, DCU, Master's and Bachelor's students are required to complete a project over the course of the year. All projects start in October and vary in duration depending on the student's year.
Each student is typically supervised in their project by a supervisor who is normally a professor and occasionally can be postdoctoral researcher. Students have regular meetings with their supervisor to discuss the progress of their projects and are assessed by submitting a report in a paper format and by an oral presentation. The evaluation of the project is not done by the students' supervisors but by examiners who have not been monitoring the progress of the students. Supervisors may, however, ask to review the mark given by the examiners if they consider that it is not representative of the student investment in the project. The success of these projects plays an important part in the student's final mark and may determine whether or not they succeed in their year, especially for students in the last year of their Master.

To help supervisors and students in the conduct of these projects, the School of Computing set up an online Student Dashboard. The dashboard is an IT platform which allows supervisors to enter data that students can access in a simple way. It contains various information such as the date and duration of meetings between supervisors and students or the supervisor's satisfaction with the student's progress. It also contains comments written by the supervisor and addressed to the students about how the meeting went, what needs to be improved or what tasks need to be done for the next meeting. All the information contained in the dashboard is detailed in section IV

At first, the objective of this project is to carry out a descriptive study of all non-textual data extracted from the dashboard. This first analysis will allow us to get an insight into the dispersion of the dashboard data, observe how supervisors mark students and which variables influence the success (i.e. mark) of the students.

In a second phase, the aim is to understand and apply to our case, a state-of-the-art Natural Language Processing (NLP) algorithm to analyse the textual comments given by the supervisors. NLP is one of the most important technologies in use today. It combines several fields such as linguistics, artificial intelligence and computer science to make models which can comprehend details from text or speech. For this project, several NLP algorithms will be used to extract the main sentiment of supervisors' comments to classify them as positive, negative or neutral or simply as positive and negative. We will then be able to observe if the main sentiment of the supervisors' comments and the students' progress influence the final mark.

Finally, from all the information gathered from the first two phases of this project, the aim is to predict whether a student is likely to succeed in the project. This will allow the supervisor to be notified in time and to support students in difficulty.

IV Dataset description

The dataset taken from the dashboard was provided in the form of an Excel table. The extraction, anonymisation and initial formatting of the data was carried out by Dr Andrew McCarren. The Excel table contains 11789 rows and 10 columns. Each row corresponds to an interaction between a supervisor and a student. Columns contain:

• *Student*: Student IDs (anonymised). There are 1002 students and their IDs can take values from 1 to 1076.
• *Supervisor*: Supervisor IDs (anonymised). There are 68 supervisors and their IDs can take values from 1 to 68.
• *Marks*: final mark of students.
• *type*: type of interaction between the supervisor and the student which can take the values:
- *meeting*: regular meeting and discussion between supervisor and student.
- *approval*: approval meeting, where students' projects are assessed by academics different from the supervisors.
- *ethics*: if an etichs approval should be sought by the student.
• *progress*: Supervisor's satisfaction with the student's progress from a meeting or approval. This can take the values *satisfactory*, *moderate* or *unsatisfactory*.
• *comments*: comments from the supervisor to the student (appreciation, improvement points, methods to be tested, description of the topics discussed during the meeting...) concerning the meeting or the approval.
• *deliverables*: text with tasks to be completed for the next meeting. However, one can also find appreciations.
• *duration*: duration of interaction between supervisor and student.

• *module*: Course followed by the student in which the project takes place. This can take the values *mcm* (Master final year project), *ca326*, (3rd year projects module code) *ca400*, *ca472* (4th year projects module code) or *pnu* (Princess Norah bint Abdulrahman University in Riyadh, Saudi Arabia, Master practicum.)
• *year*: academic year of the student.
• *approved*: Indicates whether or not the student's proposed plan for the project was approved by the examiner. This can take the values *True* or *False*. If a plan has not been approved, the student must propose a new plan.

We note the presence of missing values (NaN values) in each column of the dataset. Figure 1 shows an extract of the raw data.

	Index	approved	comments	date	deliverables	duration	module	Year	progress	type	Marks	Student	Supervisor
14	14.0	0.0	NaN	2018-03-01	NaN	NaN	mcm	2018	satisfactory	approval	58	66	27
15	15.0	NaN	31st January - meeting (student was not regist...	2018-02-08	TASKS for next week:\n0. Check timeline for fe...	30	mcm	2018	moderate	meeting	40	79	5
16	16.0	NaN	1st Feb email exchange (student yet not regist...	2018-02-08	Think about these options, find other approach...	15	mcm	2018	moderate	meeting	40	79	5
17	17.0	NaN	Read a very recent paper (2017) using RNN and ...	2018-02-08	1. draft the agreed plan into the proposal doc...	45	mcm	2018	moderate	meeting	40	79	5
18	18.0	NaN	Read through the proposal template that seems ...	2018-02-15	- Finish section on evaluation in proposal tem...	30	mcm	2018	moderate	meeting	40	79	5
19	19.0	NaN	Obtaining the code for the three approaches to...	2018-04-12	Start keeping a simple blog with dates and few...	50	mcm	2018	moderate	meeting	40	79	5
20	20.0	NaN	Average Absolute percentage error is used in p...	2018-05-18	Blog link to be provided.\nRun experiments on ...	20	mcm	2018	moderate	meeting	40	79	5
21	21.0	NaN	No success in implementing RNN form the paper...	2018-05-08	What is the difference between the MSE and the...	30	mcm	2018	unsatisfactory	meeting	40	79	5
22	22.0	1.0	Facial mood recognition using songs and facial...	2017-12-04	NaN	NaN	mcm	2018	satisfactory	approval	35	109	45

Figure 1: Extract of the dataset (pandas DataFrame)

This project has been entirely coded in Python using the computing platform Jupyter Notebook from Anaconda and the free license of Google Colab, in order to benefit from the computing power of the free Google GPUs used for the neural networks involved in the NLP algorithms.

V Step 1: Analysis of non-textual data

The first part of this project consists in performing a descriptive analysis of all non-textual data to get a first insight of the factors influencing the final mark of students. The *comments* and *deliverables* columns are therefore removed from the dataset.

V.1 Dataset cleaning

Although a first formatting of the dataset has been done, it is necessary to clean up the *duration* column as the duration has not always been entered correctly. A function to hamonize the duration (cleaning, transforming unknown characters into NaN values and transforming hours into minutes) is implemented.

Once the data is cleaned and the correct format has been applied, we can perform the descriptive analysis.

V.2 Distribution of students in the different modules

The graph in figure 2 shows the distribution of students in the different modules for each year.

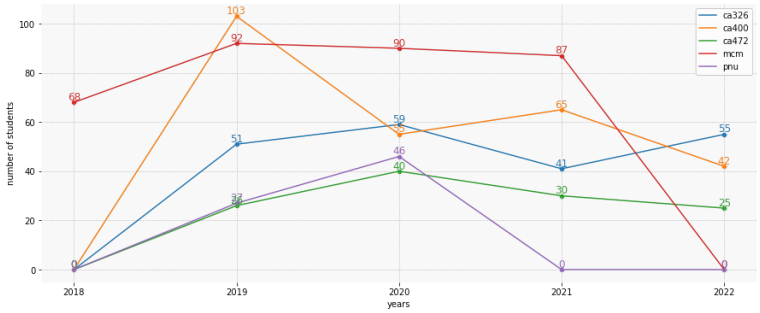


Figure 2: Number of Students per module for each year

The module with the most students is *mcm* with about 70-90 students each year. 3rd and 4th years code modules (*ca326*, *ca400* and *ca472*) have fewer students with about 40-60 students per year, with an exception for the *ca400* module which had 103 students in 2019. Finally, the *pnu* module was only present in 2019 and 2020 with about 35 students.

V.3 Distribution of marks

The figure 3 shows the distribution of students' marks.

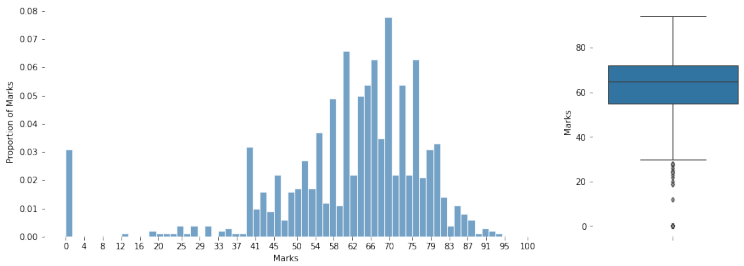


Figure 3: Histogram and boxplot of the Students' marks

The marks are globally good with a median around 65/100. The pass mark is 40/100 and the first quartile being at 55/100, we can say that less than 1/4 of the students fail to pass. The highest score is 94/100 while the lowest is 0/100. After computation, 3% of the marks are at value 0. Similar histograms and boxplots in appendix A.1 have also been plotted for each module.

Figure 4 presents the median mark per module for each year.

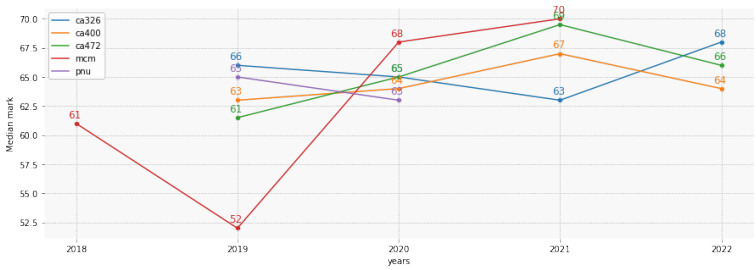


Figure 4: Median mark per module for each year.

We notice that students' marks have globally increased during the Covid period of 2020-2021 for modules *mcm*, *ca400* and *ca472* gaining 3 points on the median value (from about 66/100 to

about 69/100) while module *ca326* lost 2 points on its median (from 65 to 63). Nevertheless, we can see that the variations in the median are small over the years 2019-2022, except for the *mcm* module which has considerably increased from a median of 52/100 in 2019 to 70/100 in 2021.

Finally, Figure 5 shows the boxplots of student marks for each module.

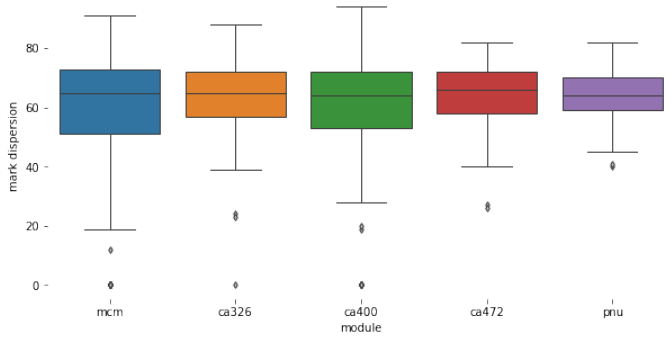


Figure 5: Boxplot of students' marks for each module from 2018 to 2022.

The median of the modules is uniform around 65/100 which is also the overall median of the marks as seen in the boxplot of figure 3. The modules *mcm* and *ca400* have a larger mark dispersion. This can be explained by the fact that these are also the modules with the most students as can be seen in figure 2. For these modules, the minimum mark is around 20/100 for *mcm* and 30/100 for *ca400* and maximum is around 85/100. Modules *ca326*, *ca400* and *pnu* have a higher minimum around 40/100 and a lower maximum around 80/100.

V.4 Distribution and influence of the progress on the mark

Figure 6 represents the boxplot of the student's marks grouped by their progress. If a student has at least one unsatisfactory progress, they fall into the *unsatisfactory box*. If a student has no unsatisfactory progress but at least one moderate progress, they fall into the *moderate box* and if a student has only satisfactory progress they fall into the *satisfactory box*. Thus, a student falls into only one box.

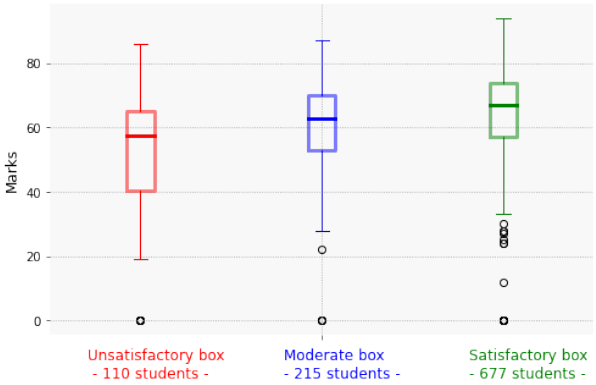


Figure 6: Boxplot of Student's marks grouped by their progress fomr 2018 to 2022.

Figure 6 shows that the poorer the students' progress, the lower the mark: the median mark for students in the *unsatisfactory box* is 57/100, the median mark for students in the *moderate*

box is 63/100 and students in the *satisfactory box* have the highest median at 67/100. It can be assumed that student's progress given by the supervisor (unsatisfactory, moderate or satisfactory) and student's mark are correlated. Thus, the progress given by the supervisor could be a good indicator of whether a student will succeed in the project.

V.5 Distribution and influence of the meetings' duration on the mark

The duration is the amount of time meetings between supervisors and students last in minutes. As mentioned in the section V.1, the dashboard interface contains a box where the supervisor can enter the duration of the meeting. However, the duration was filled for only 36% of the meetings. Figure 7 allows to identify which supervisors filled the duration box.

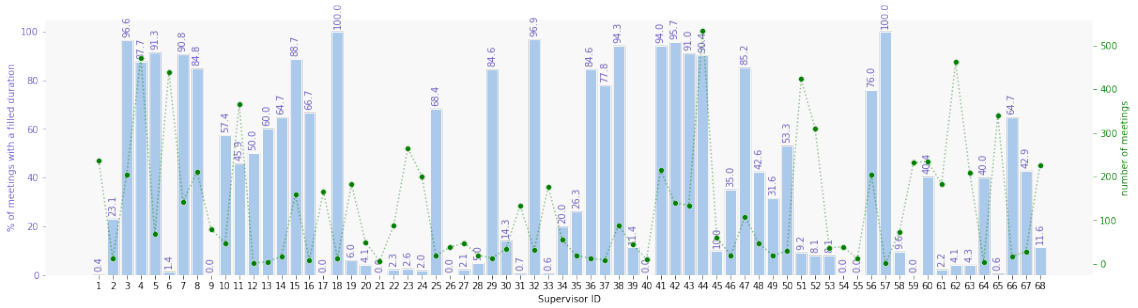


Figure 7: Plot of the percentage of meetings with a filled duration and the number of meetings for each supervisor.

We can see from the figure 8 that the duration of a meeting does not seem to have an influence on the students' mark since all boxplots are aligned. From Figure 9, duration and marks are not correlated as no pattern can be identified in the scatterplot. The duration of meetings is therefore no indication of whether a student will succeed in the project.

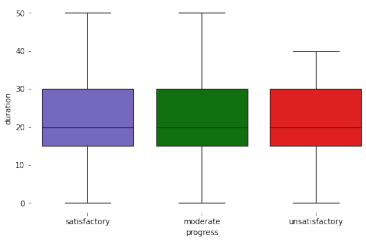


Figure 8: Boxplot of progress and duration of meetings

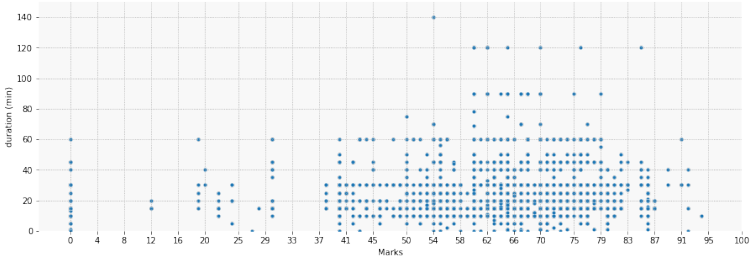


Figure 9: Scatter plot of duration as a function of Marks

V.6 Understanding supervisors behaviour

The figures 10 and 11 below display the proportion of satisfactory, moderate and unsatisfactory progress given, the average mark assigned, the dispersion of marks, the number of students and the number of meetings for each supervisor.

The graph figure 10 shows that a high proportion of unsatisfactory progress for a supervisor does not always represent a low mark. Some supervisors tend to be more severe and use the unsatisfactory progress in order to notify the students of a problem to be rectified, as for example

supervisor n°56 (average mark 62/100) and who has about 10% unsatisfactory progress. On the opposite, supervisors such as supervisor n°25 may tend to give good progress even if the work results in a poor mark (average mark 32/100). However, this needs to be compared with the graph figure 11 by looking at the number of students per supervisor. Supervisor n°25, for example, has less than 10 students, which does not really allow to observe a behaviour.

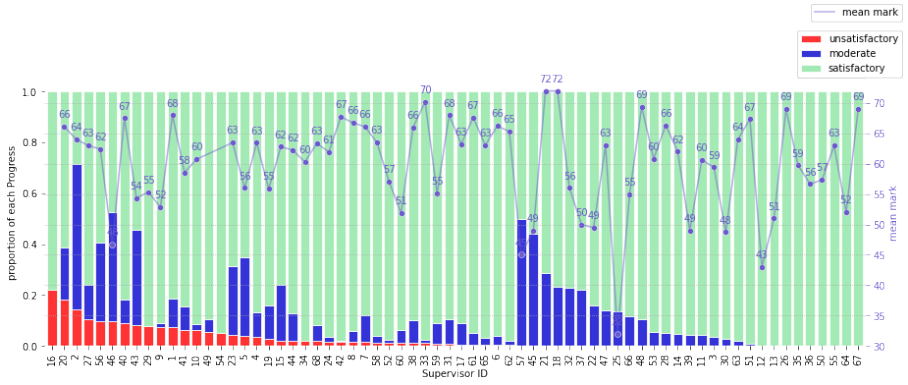


Figure 10: Progress and mean mark for each Supervisor sorted by satisfaction

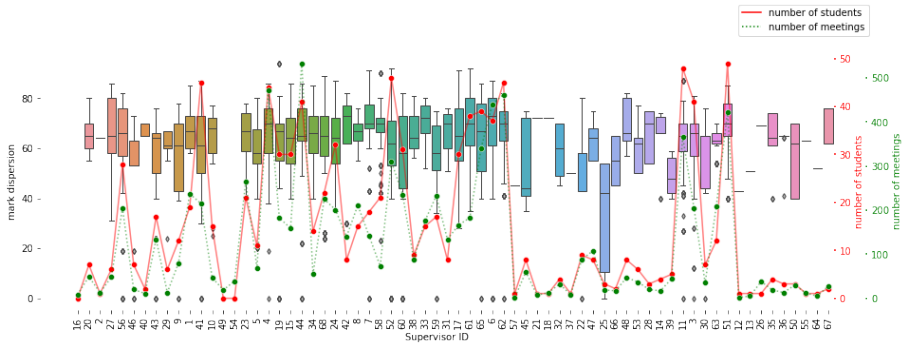


Figure 11: Mark dispersion, number of students and number of meetings for each Supervisor sorted by satisfaction

Graphs similar to those in figure 10 and 11 are constructed for each module and are presented in Appendix A.2.

V.7 Implementation of interactive panels to facilitate visualisation

Due to the large number of students (1002 students), interactive panels are created to select the information to be displayed.

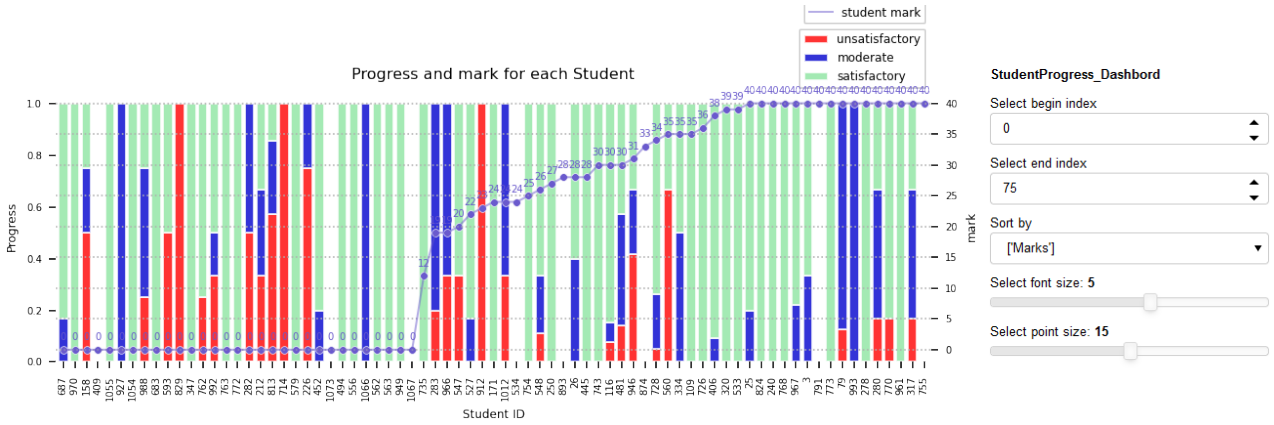


Figure 12: Interactive panel showing students' mark and associated progress in percentage, sorted by mark values.

The panel in figure 12 allows the user to select the range of student IDs to be displayed and to sort them by ascending marks (as shown above), satisfaction or by student IDs.

The panel in figure 13 displays all students IDs on a single graph, sorted by increasing mark value. The proportion of unsatisfactory progress associated with each mark is also displayed. It can be seen that for a student, the lower the mark (on the left of the graph), the closer the proportion of unsatisfactory progress is to 1 i.e. the higher the number of unsatisfactory progresses for the students. This is consistent with the graph shown in figure 6, which shows that students' marks are correlated with their progress. This panel also allows the user to sort students by progress, display the proportion of moderate or satisfactory progress and to hide values such as proportions of 1 or 0 for better visibility.

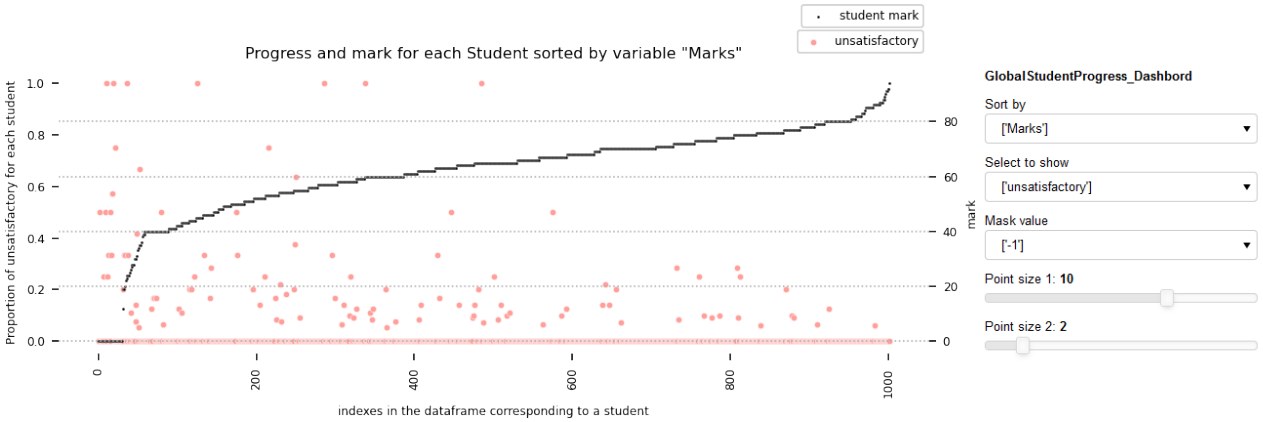


Figure 13: Interactive panel showing students' mark and proportion of unsatisfactory progress, sorted by mark values.

The panel shown in figure 14 is used to select a student ID and display for that student the date, duration and progress of meetings and approvals over the year, as well as the final marked obtained. The panel in figure 15 displays for a supervisor and a given period of time, the number of meetings, approvals and ethics and the associated duration. The information can be grouped by year, day or month.

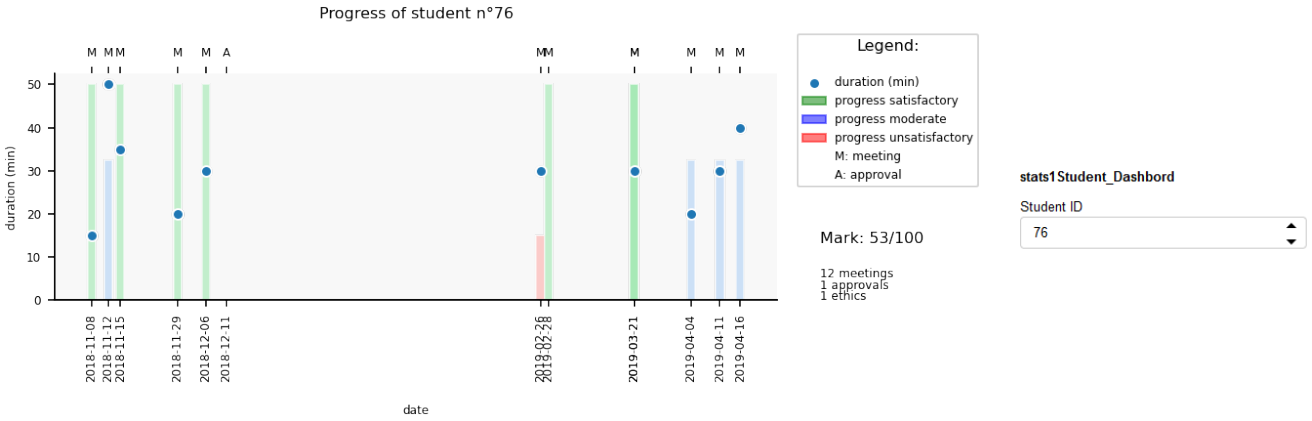


Figure 14: Interactive panel showing information for one student.

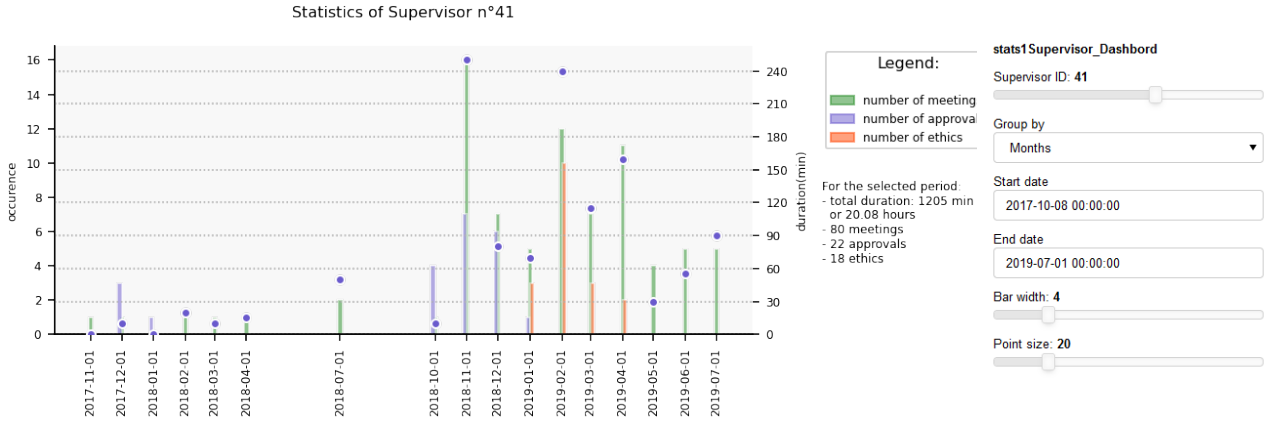


Figure 15: Interactive panel showing information for one supervisor.

Once a first descriptive analysis has been carried out, we now use NLP algorithms to find the main sentiment of the comments.

VI Step 2: Sentiment analysis of the comments using NLP algorithms

Sentiment analysis is the process of identifying someone's opinions expressed in text and categorizing them as positive, negative, or neutral.
State of the art NLP algorithm offering the best performance often require labelled data. However, the dataset provided does not contain the true sentiment of the comments and labelling such a large amount of data by hand would be very time-consuming.
Therefore, we will use as a first simple method unsupervised learning algorithms based on lexicon approaches. Then, following the advice of Dr Annalina Caputo, we will apply the state of the art supervised algorithm BERT [3], for which we need labelled data.

Numerous papers report the application of BERT or other NLP algorithms for sentiment analysis. The majority of the datasets used in these papers contain:
- Reviews from streaming websites like movie reviews with the paper [4] applying BERT to the IMDb dataset or [5] applying BERT to the Stanford Sentiment Treebank dataset.
- Buisiness reviews such as restaurants, shopping, entertainment, activities or services with the

papers [4] and [6] applying the algorithms BERT and XLNet to the Yelp dataset.
- Messages from users of the tweeter platform with the paper [7] using LSTMs + CNNs model on the SemEval dataset.
- Amazon reviews with the paper [8] or financial texts with [9], both using BERT for sentiment analysis.

Therefore, the challenge of this project lies essentially in the application of the supervised algorithm BERT to supervisors' comments, since we did not find such labelled datasets on internet to train our model. Moreover, the text of the supervisors' comments is very different (more nuanced, longer and more descriptive) from the text of the labelled datasets listed above. It is not guaranteed that training BERT with the labelled datasets found on internet will provide good results on the supervisors' comments.

VI.1 A first naive approach with unsupervised NLP algorithms

This section presents the first approach, which consists of implementing unsupervised learning algorithms based on lexicon approaches.

VI.1.1 Sentiment analysis with TextBlob

TextBlob is a Python library for processing textual data providing a simple API for unsupervised NLP tasks such as sentiment analysis. It is based on a lexicon approach and uses Natural Language ToolKit (NLTK) library. [10]

Lexicon-based approach uses a pre-defined dictionary containing the vocabulary of a language and the sense in which each word is used. For each word used in a particular sense, the dictionary gives its polarity (real value between -1 for negative and 1 for positive) and its subjectivity (real value between 0 for objective and 1 for subjective). [11]

The polarity (or sentiment) of a word is calculated by TextBlob by averaging all polarity values given by the different senses of this word. The polarity of a sentence is calculated by averaging polarity values of each word in the sentence. Finally, if TextBlob is applied to a long text, the average polarity of each sentence will be returned. Thus, this method is naive and may be outperformed by supervised NLP algorithms.

As the size of the comments can be long, predicting by the average may not be very accurate. We decide to work at the sentence level by predicting with TextBlob the sentiment of each sentence in the comment and take as the comment's overall sentiment, the most common sentiment among each sentence.

To transform the polarity value returned by TextBlob (which is a real number between -1 and 1) into a rating out of 3, the following rule is considered:
- if polarity of the sentence $\leq t_1$ the sentiment is negative,
- if polarity of the sentence $\geq t_2$ the sentiment is positive,
- otherwise the sentiment is neutral.

t_1 and t_2 are thresholds optimized in order to have the best Accuracy and F1-score values on a validation set. Accuracy and F1-score are performance metrics that evaluates the ability of a

classification model to provide the best results. However, Accuracy can be misleading if used with imbalanced datasets and in this case the F1-score is preferred [12].

Before applying TextBlob, it is necessary to clean up the text by removing urls, unknown characters, newline characters or words containing numbers. The punctuation of the text is kept because it is taken into account into TextBlob sentiment analysis. However, TextBlob is not case sensitive.

VI.1.2 Sentiment analysis with VADER

VADER works on a lexical approach in the same way as TextBlob and therefore does not require labelled data. VADER sentiment analysis works best on short documents and can detect the sentiment of emoticons, acronyms and slang. In addition to the lexical approach, VADER considers five simple heuristics: punctuation ("Great" and "Great!!"), capitalization ("great" and "GREAT"), degree modifiers ("super cute" and "sort of cute"), shift in polarity due to "but" and tri-grams to detect negation. [13]

For each text, the VADER sentiment analyzer returns a dictionary of four elements containing the percentage of negative, neutral and positive sentiment in the text, and the compound score. The compound score is commonly used for sentiment analysis and ranges from -1 to 1. We then use the compound score in the same way as the polarity score returned by TextBlob and use the same method as in section VI.1.1 to transform the compound value into a score out of 3 and to compute the overall sentiment of a comment.

VI.2 A second approach based on supervised state-of-the-art algorithm BERT

We now use a supervised model to calculate the main sentiment of comments and expect better performances than the previous unsupervised methods. The implemented model will allow comments to be classified as positive, negative or neutral (3 labels) or only as positive and negative (2 labels).

VI.2.1 Overview of the model

BERT basis:

To predict the comments' sentiment, we use the BERT algorithm which stands for Bidirectional Encoder Representations from Transformers. It is a state-of-the art supervised Machine Learning model which can be used in more than 11 NLP tasks developed in 2018 by researchers at Google AI Language. [3]

BERT's model is based on a multi-layer bidirectional Transformer encoder architecture. Multi-layer Transformer neural network was originally created to solve the problem of language translation. It is formed by an encoder and a decoder block. This type of architecture is fast (words can be processed simultaneously) and deeply bidirectional by learning information from left to right and from right to left simultaneously, providing a better understanding of language and context [14]. The BERT model consists of several encoder blocks placed one after the other.

BERT training is divided into two steps: **pre-training** and **fine-tuning**.

BERT pre-training:

During pre-training, the model is trained on unlabeled data over two tasks simultaneously:
- Masked Language Modeling (MLM), where a word is hidden in a sentence and the model bidirectionally uses the words on either side of the covered word to predict the masked word.
- Next sentence Prediction (NSP), where the model predicts if a given sentence follows the previous sentence or not. BERT was pre-trained on large informal datasets such as Wikipedia and Google's BooksCorpus which gave BERT deep knowledge on the English language. [3]

BERT input:

To provide BERT with an input that it can understand, the raw text is separated into smaller units called tokens in the process of Tokenization. Tokens can be words, characters, or subwords and are mapped with an integer number called token ID. BERT uses WordPiece algorithm for Tokenization which has a vocabulary of about 30K tokens. BERT also uses the following special tokens:
- [CLS] to add at the beginning of every sentence and [SEP] to add at the end of every sentence for the NSP task.
- [MASK] to hide some of the input tokens for the MLM task.
- [UNK] for unknown words which are not part of BERT vocabulary.

Then, each token is converted into embeddings using pre-trained WordPiece embeddings. For BERT base model, embeddings are vectors of dimension 768 [15] that encapsulates the meaning of the word such that similar words have similar embeddings.

Thus, the input of BERT consists in blocks of two sentences A and B with some tokens being masked in the following format: "*[CLS] sentence A [SEP] sentence B [SEP]*".
In question-answering problems, the token [SEP] is used to tell the model that the text in sentence A (question) has a different role from the text in sentence B (paragraph containing the question's answer). However, as our problem is a classification problem, our input will be of the form: "*[CLS] sentence A [SEP]*".
It should be noted that the term *sentence* used in the BERT paper [3] is not an actual linguistic sentence and may be an arbitrary span of a contiguous text. In our case, a sentence will represent a supervisor's comment.

BERT output:

The BERT encoder outputs as many hidden states as there are input tokens. These output hidden states are used in different NLP tasks. The first hidden state corresponding to the first token [CLS] is a vector of size 768 in BERT base model and is used in the NSP task to predict if sentence B actually follows sentence A. However, this first hidden state also contains the understanding of BERT at the sentence level and will be used as an input in our classification tasks in the fine-tuning step, as shown in figure 16.

BERT fine-tuning:

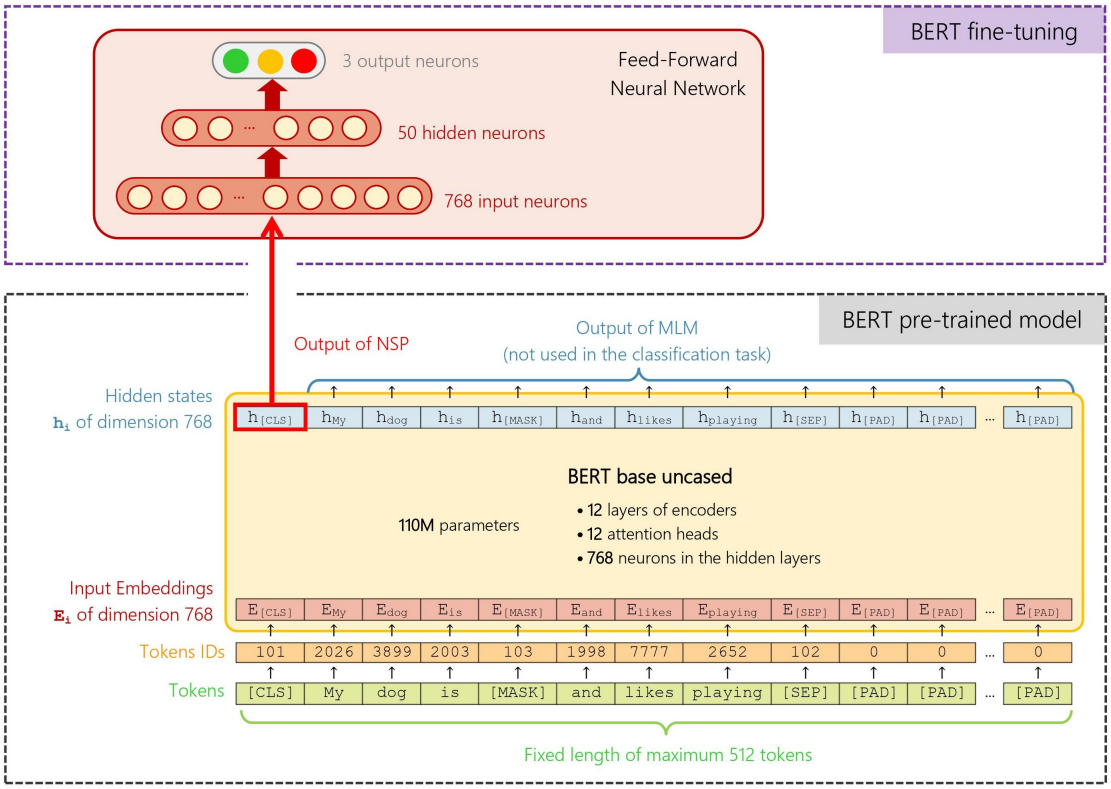


Figure 16: Illustration of BERT pre-training and fine-tuning in the case of 3 output labels (positive, negative, neutral)

For fine-tuning, the BERT model is first initialised with the pre-trained parameters and all of the parameters are fine-tuned using labeled data adapted to the NLP task we want to solve.

The HuggingFace's transformers [16] and PyTorch [17] library will be used to fine-tune a pre-trained BERT model for our classification task.

We will use the pre-trained *bert-base-uncased* model consisting of : 12 encoder blocks, 768 neurons in the hidden layers and 12 attention heads for a total of 110M parameters. In addition, the GPUs from Google Colab will be used to accelerate computation time since the neural network to be trained is very large. To implement BERT fine-tuning for sentiment analysis, we essentially relied on the tutorial [18].

VI.2.2 Text preprocessing

Cleaning of the text:

Since BERT has a deep understanding of language and context, the text does not need to be thoroughly cleaned. Stop words and punctuation removal, lemmatization or typing errors correction are not necessary. However, url links, unknown characters, newline characters and words containing digits have been removed from the text.

Tokenization of the text:

To apply the pre-trained model, tokenization must be performed by the BERT tokenizer provided by the transformers library [15]. Here, the *bert-base-uncased* tokenizer having a vocabulary of 30522 tokens is loaded.

Before tokenizing, we need to specify the maximum length of our sentences. The BERT model receives as input a fixed length of maximum 512 tokens, truncating anything beyond this length. However, the texts we have to analyse are long and exceed this maximum size. Therefore, we implement a function to truncate the text to 512 tokens while trying to preserve as much information as possible. One of the simplest, possible approaches is to select the beginning and the end of the text as suggested in the paper [19]. The first statement of a review usually expresses the main idea, and the last one summarizes it. We can assume that the main sentiment will be located at the beginning and at the end of the text. Based on [19], we select the 123 first tokens and 382 last tokens of each text for a total of 510 tokens.

Then, we add the special tokens [SEP] and [CLS] to the start and end of each comment. For comments that are shorter than the maximum length, paddings or empty tokens [PAD] are added to match the maximum size.

Finally, we create the Attention Mask vector indicating which tokens are padding and which are not. This mask tells the "Self-Attention" mechanism in BERT not to incorporate these [PAD] tokens into its interpretation of the sentence.

VI.2.3 Formatting training and validation datasets

To train and evaluate our machine learning model, we split the data between a training set (90% of the overall dataset) and a validation set (10% of the overall dataset).

To help prepare, manage and serve our data to the neural network and to accelerate computation, the Torch DataLoader class [20] is used. We obtain the training and validation DataLoaders formed of batches containing for each comment, its vector of tokens IDs, its attention mask vector and its true label.

VI.2.4 Fine-tuning the model

Now that the input data is properly formatted, we will implement functions for BERT fine-tuning. For this task, we first want to modify the pre-trained BERT model to give outputs for classification, and then we want to continue training the model on our specific dataset for sentiment analysis until that the entire model is well-suited for our task.

Creation of our own BERT classifier:

To define our own BERT sentiment classifier, we implement a *BertClassifier* class. To initialise this class, we first load the *bert-based-uncased* pre-trained model [16].

Then, we extract the last hidden state of the [CLS] token (of size 768) from the pre-trained model. This will constitute the input of our own classifier which is a one hidden layer feed-forward neural network. The classifier contains 768 neurons for the input layer, 50 neurons for the hidden layer and 3 neurons for the output layer to classify the comments as positive, neutral or negative as shown in figure 16. We change the number of neurons in the output layer to 2 to classify the comments as positive and negative. The ReLU activation function is applied in the hidden layer.

Our *BertClassifier* class also contains a function *Forward* that feeds input to our BERT classifier to compute logits. Logits are raw, unnormalized scores for each of the 3 classes for each comments and are the output of the neural network before going through a softmax or sigmoid activation function. No activation function is applied on the last layer since the loss function used by PyTorch library in the neural network's gradient calculations are based on the raw logits [21].

Initialisation of our model:

Neural networks are trained with a Stochastic Gradient Descent and a Backpropagation algorithms. Here the AdamW optimization algorithm [22] is used instead of the classical Stochastic Gradient Descent algorithm for training our model. AdamW algorithm is a variant of Adam optimizer [23] allowing to handle sparse gradients on noisy problems and resulting in better generalizing models.

We implement a function to initialise our model. This function defines a BERT classifier using our own *BertClassifier* class, an optimizer and a learning rate scheduler using AdamW optimizer and the scheduler from Transformers library. This function also allows to initialise a model from a previously fine-tuned model by retrieving the value of all the parameters, as well as the state of the optimizer and the scheduler, to resume the training for further fine-tuning.

Function to train the model:

A function to train our *BertClassifier* model over several epochs is implemented. For each batch of the training set, the function performs forward and backward passes to compute logits and gradients and update the neural network parameters. After the completion of each epoch, the model's performance on the validation set is measured and displayed and the model's parameters and current state are saved so that one can resume training for further fine-tuning. The Cross-Entropy loss function from PyTorch library [21] is used and takes as input the logits and the real labels of each comments of the training set.

VI.2.5 Predictions on the validation set

Get the probabilities on the validation set:

A function is implemented to get the predictions on the validation set. It performs a forward pass to compute logits and apply the softmax activation function to calculate probabilities. The softmax function: $\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$ takes as input the logits for each comment x which are vectors of $K = 3$ real numbers and outputs a probability distribution of K possible outcomes. We then obtain the probability of each comment belonging to each class (negative, positive or neutral) as illustrated on the table 1.

Get the predictions on the validation set:

To compute the final label of our dataset, we implement a function taking as an input the probabilities of each comment belonging to each class and consider two rules.

Comments in the validation set	Probability of belonging to class negative	Probability of belonging to class neutral	Probability of belonging to class positive
comment 1	0.9308365	0.0529936	0.01616999
comment 2	0.33216745	0.32451671	0.34331584
...	...	...	...
comment n	0.10061309	0.2020040	0.6973828

Table 1: Predictions obtained on the validation set.

For the first rule, the predicted label is simply the class with the highest probability. However, while in the case of comment 1 in the table 1, it is clear that the negative class should be predicted, the prediction for comment 2 is more ambiguous as the three probabilities are very close. To overcome this issue, we implement a second prediction rule where the label predicted is also the one with the highest probability. However, if the absolute difference between the probabilities is less than a certain threshold T to optimise, there is confusion between the probabilities and the label is put to neutral.

We implement a function to calculate the F1-score and the Accuracy of the predictions using metrics from scikit-learn library as well as the contingency table of predictions, for both prediction rules. In the case of the second rule, we will take for the threshold T, the value corresponding to the best Accuracy and F1-score.

VI.2.6 Training the model on the entire dataset

Finally, we train our model on the whole dataset (by concatenating the train set and the validation set) and save the results and parameters in an excel file.

VI.2.7 Two-phase fine-tuning and labeled datasets

As mentioned in section VI.2.1, BERT fine-tuning step needs labeled data. As our comments are not labeled, we decide to perform a *phase 1* first fine-tuning on labeled datasets for sentiment analysis found on the Kaggle website [24]. As we cannot find datasets that are similar to teachers' comments for students, we try to build the most general datasets possible, by finding them directly on Kaggle or by mixing several datasets. Table 2 shows the datasets used. We try not to overfit the model in *phase 1* fine-tuning by training on few epochs and we asses the predictions on the validation set using the F1-score metric.

In order to improve the predictions, we perform a *phase 2* fine-tuning. To do this, we manually label 200 comments which we will call `phase2_data`. Two people label all the comments and in case of disagreement, a third person helps to decide. We then retrieve the fine-tuned models from *phase 1* and resume the fine-tuning but this time, on the 200 `phase2_data` comments. We asses again the performance on the validation set.

However, it appears that 200 comments is too few and makes no difference in the *phase 2* fine-tuning since it gives identical probabilities to the *phase 1* fine-tuning, regardless of the number of epochs or the learning rate value. To increase the size of the 200 `phase2_data` comments, we consider two methods. The first method is to concatenate and shuffle several copies of `phase2_data` comments to give them more weight in the *phase 2* fine-tuning. The second method is to use text data augmentation. Data augmentation allows to increase the amount of data by

adding slightly modified copies of already existing data or newly created synthetic data from existing data. However, we have to make sure that the augmentation does not change the main sentiment of the comments. According to the paper [25], the Back-Translation (BT) method gives the best results for augmenting small or unbalanced datasets with BERT algorithm for sentiment analysis. The BT method consists on translating text data to another language and then translating it back to the original language, allowing to generate textual data distinct to original text while preserving the original meaning. We use the BT method from nlpaug Python library [26] by back-translating the 200 `phase2_data` comments into German, French and Chinese. We name the augmented datasets `phase2_BT_ge`, `phase2_BT_fr` and `phase2_BT_ch` respectively, each one containing 200 comments. Finally, we use textual augmentation by word embeddings from nlpaug library. This technique randomly substitute words in a sentence with a word having similar embedding. We name the augmented dataset `phase2_WE`, contaning 200 comments.

Labeled datasets:

Dataset	Link	Score	Description	Used for
1) Amazon Kindle Book Review	[27]	from 1 to 5	Product reviews from Amazon Kindle Store category from May 1996 - July 2014.	3 and 2 labels
2) Financial Sentences	[28]	from 1 to 3	Financial sentences with sentiment labels	3 labels
3) Amazon General Review	[29]	from 1 to 5	Reviews from various amazon products collected over a period of 18 years.	3 and 2 labels
4) IMDB dataset	[30]	from 1 to 2	Movie reviews from Rotten Tomatoes website (sentences of more than 200 words)	2 labels

Table 2: Datasets used in the *phase 1* fine-tuning

In the case of 3 labels, we transform scores from 1 to 5 into scores out of 3 with the following rule: if score ≤ 2 assign negative, if score ≥ 4 assign positive and if score $= 3$ assign neutral. In the case of 2 labels, we transform ratings from 1 to 5 into scores out of 2 with the rule: if score ≤ 3 assign negative and if score > 3 assign positive.

VI.3 Evaluation Results

In order to build our test set, two people (and a third in case of disagreement) labelled 50 comments. These 50 comments are different from the 200 comments used in the *phase 2* fine-tuning.

VI.3.1 TextBlob and VADER results

For the unsupervised algorithms TextBlob and VADER, the results are only available in the case of 3 labels, due to time constraints. As seen in section VI.1, we implement two functions to calculate the optimal thresholds t1 and t2 for the TextBlob and VADER methods. The functions returns the thresholds t1 and t2 (among a grid of possible values) which maximise the F1-score and the Accuracy on the validation set. Once the optimal thresholds are found, we use them to predict the sentiment of the comments of the test set. The results for 3 labels are presented in the Table 3.

We notice that TextBlob gives better results than VADER since the F1-score and Accuracy values are higher.

method	F1-score	Accuracy
TextBlob	0.41	0.68
VADER	0.34	0.58

Table 3: Results of TextBlob and VADER methods on the test set in case of 3 labels.

VI.3.2 BERT results

We perform several runs of our code in the following setups:

a) Fine-tuned model with only *phase 1* on 6000 comments from Kaggle datasets.

b) Fine-tuned model with only *phase 1* on 18000 comments from Kaggle datasets.

c) Fine-tuned model with *phase 1* on 6000 comments from Kaggle datasets and *phase 2* on 200 manually labeled comments augmented with the first method.

d) Fine-tuned model with *phase 1* on 6000 comments from Kaggle datasets and *phase 2* on 200 manually labeled comments augmented with the second method.

We perform model selection for each of the above configurations, by selecting the meta-parameters (datasets used, number of epochs, learning rate, batch size...) that give the best F1-score on the validation set. We then evaluate the performance of our model on our test set and obtain the results presented in the table 4 and 5.

Setup	Fine-tuning phase	Mix of dataset used	Number of comments	batch size	number of epochs	eps value in AdamW optimizer	lr value in AdamW optimizer	BERT performance on the test set with rule 1 (no threshold)		BERT performance on the test set with rule 2 (best threshold)		
								Accuracy	F1-score	Best Threshold	Accuracy	F1-score
a)	phase 1	6000 comments of dataset 3)	6000	8	2	1×10^{-8}	5×10^{-5}	0.66	0.508	0.7	0.78	0.573
b)	phase 1	18000 comments of dataset 3)	18000	8	2	1×10^{-8}	5×10^{-5}	0.24	0.231	0.9	0.70	0.518
c)	phase 1	6000 comments of dataset 3)	6000	8	2	1×10^{-8}	5×10^{-5}	0.80	0.620	0.7	0.82	0.620
	phase 2	5 copies of `phase2_data`	1000	8	2	1×10^{-8}	5×10^{-5}					
d)	phase 1	6000 comments of dataset 3)	6000	8	2	1×10^{-8}	5×10^{-5}	0.74	0.584	0.9	0.76	0.575
	phase 2	`phase2_data` + `phase2_BT_ge` + `phase2_BT_fr` + `phase2_BT_ch` + `phase2_WE`	1000	8	2	1×10^{-8}	5×10^{-5}					

Table 4: Results of BERT fine-tuning on the test set in the case of **3 labels**

Setup	Fine-tuning phase	Mix of dataset used	Number of comments	batch size	number of epochs	eps value in AdamW optimizer	lr value in AdamW optimizer	BERT performance on the test set with rule 1 (no threshold)		BERT performance on the test set with rule 2 (best threshold)		
								Accuracy	F1-score	Best Threshold	Accuracy	F1-score
a)	phase 1	1000 comments of dataset 1) + 1000 comments of dataset 3) + 4000 comments of dataset 4)	6000	8	2	1×10^{-8}	5×10^{-5}	0.72	0.563	0.8	0.86	0.642
b)	phase 1	12000 comments of dataset 3) + 6000 comments of dataset 4)	18000	8	2	1×10^{-8}	5×10^{-5}	0.66	0.520	0.7	0.74	0.578
c)	phase 1	1000 comments of dataset 1) + 1000 comments of dataset 3) + 4000 comments of dataset 4)	6000	8	2	1×10^{-8}	5×10^{-5}	0.86	0.573	0.2	0.88	0.592
	phase 2	5 copies of `phase2_data`	1000	8	2	1×10^{-8}	2×10^{-5}					
d)	phase 1	1000 comments of dataset 1) + 1000 comments of dataset 3) + 4000 comments of dataset 4)	6000	8	2	1×10^{-8}	5×10^{-5}	0.96	0.864	0.1	0.96	0.864
	phase 2	`phase2_data` + `phase2_BT_ge` + `phase2_BT_fr` + `phase2_BT_ch` + `phase2_WE`	1000	8	2	1×10^{-8}	5×10^{-5}					

Table 5: Results of BERT fine-tuning on the test set in the case of **2 labels**

We can see from Tables 4 and 5 that the best Accuracy and F1-score are obtained with setup c) in the case of 3 labels and with setup d) in the case of 2 labels. We also observe that using 2 labels gives better results than using 3 labels. The results obtained seem satisfactory as we obtain rather high F1-score and Accuracy values, in the case of 3 labels (0.80 for Accuray and 0.620 for the F1-score with rule 1) and 2 labels (0.96 for Accuray and 0.864 for the F1-score with rule 1).

We select the best BERT models in cases 3 and 2 labels and use them to predict the sentiment of all comments from the dashboard.

We also observe that BERT, which is a supervised algorithm, gives better results than the unsupervised methods such as TextBlob or VADER as seen in section VI.3.1. We will there-

fore use BERT rather than TextBlob or VADER to predict the sentiment of the dashboard's comments.

Finally, the sentiment of the dashboard's comments computed by BERT best models will then be used to predict the mark of students as we will see in the section VII.

VII Step 3: Prediction of students' marks

In this section we aim to predict the students' marks. To do this, we will use the main sentiment of the comments calculated above as well as other computed features.

VII.1 Creation of a new dataset

The students' marks are represented by the variable *Marks*. We transform *Marks*, which is a number out of 100, into a letter corresponding to the higher level academic grading in the Republic of Ireland as shown in Table A.1. *Marks* becomes a categorical variable.

In order to predict students' marks, we compute a new dataset composed of features extracted from the dashboard data, which we believe may have an influence on the mark. The new dataset contains the following elements:

- *Student:* student IDs.
- *module:* course followed by the student. This can take the values *mcm*, *ca326*, *ca400*, *ca472* or *pnu*.
- *Supervisor:* supervisor IDs.
- *nbMeetings:* number of meetings for each student.
- *worstProgress:* Students marks grouped by progress, constructed as follows:
  - If a student has at least 1 unsatisfactory progress, *worstProgress* is *unsatisfactory*.
  - If a student has 0 unsatisfactory but at least 1 moderate progress, *worstProgress* is *moderate*.
  - If a student has 0 unsatisfcatory and 0 moderate progress, *worstProgress* is *satisfactory*.
- *meanTimeBetweenMeet:* mean time between meetings for each Student in days.
- *meanTimeMeet:* mean time of a meeting in minutes.
- *timeFirstDecay:* time in days between the beginning of the academic year and the first unsatisfactory or moderate meeting.
- *mainSentiment:* most common sentiment among the meeting comments for each student. The sentiment of the comments was previously computed with the best models found in section VI.3.2.
- *is0Sentiment:* contains True if the meetings note contains a negative comment and False otherwise.

Once the dataset is created, we start by transforming the categorical variables into dummy variables.

Our new dataset contains missing values, especially for the feature *meanTimeMeet* (see section V.5). As many machine learning algorithms cannot handle missing data, we perform data imputation with missForest algorithm. MissForest is an iterative multiple imputation method

based on random forest. It does not require parameter tuning or assumptions on the distribution of the data and performs well with complex interactions or non-linear relations between variables of unequal scales and different type (continuous and categorical) [31]. We use the MissForest function from the missingpy Python library [32].

Finally, we split the dataset between train set (80%) and test set (20%) and standardise the continuous variables.

VII.2 Machine learning algorithm for prediction

VII.2.1 Multinomial logistic regression

To predict students' mark, we first fit a multinomial logistic regression model using the LogisticRegression function from scikit-learn python library [33].

Without feature selection:

We begin by fitting a model without variables selection. We use the solver 'lbfgs' and penalty 'l2' (for Ridge regression) in the LogisticRegression function. Ridge regression helps to reduce model complexity and prevent over-fitting. The penalty parameter C in Ridge regression is optimised with the function GridSearchCV from scikit-learn python library. GridSearchCV performs an exhaustive search over a parameter grid and selects the parameter giving the smallest cross-validated error. We use 5 folds and the scoring method 'f1_macro' (macroscopic F1-score) in GridSearchCV.

We redefine our multinomial logistic regression model with the best parameters found with GridSearchCV and compute the final cross-validated error with Python function RepeatedStratifiedKFold from scikit-learn, using 3 repeats, 10 folds and 'f1_macro' scoring. Stratified cross-validation ensures that the training and test sets in the cross-validation procedure have the same features proportion as in the original dataset, allowing to better approximate the generalisation error. RepeatedStratifiedKFold repeats the stratified cross-validation procedure multiple times to improve the approximation of the generalisation error, and reports the mean 'f1_macro' score across all folds from all runs.

Feature selection with Lasso regression:

Similar to Ridge regression, Lasso regression also reduces model complexity and prevent over-fitting. However, this type of regularisation can lead to zero coefficients in the multinomial logistic regression model and therefore performs feature selection. We use the solver 'saga', suitable for Lasso regression, and penalty 'l1' (for Lasso regression) in the LogisticRegression function and find the best penalty parameter C of Lasso regression with GridSearchCV function. We redefine the multinomial logistic regression model with the best parameter, compute the final cross-validated error with RepeatedStratifiedKFold and display the selected features.

Feature selection with Recursive Feature Elimination:

In this section, we define again a multinomial logistic regression model using the default 'lbfgs' solver and no penalty. Features are selected using the RFE function from scikit-learn. RFE stands for recursive feature elimination and consists of first training the model on the

initial set of features and then selecting the features recursively by pruning the least important ones until the desired number of features to select is reached. The number of features chosen by RFE can be found automatically using RFECV function. RFECV performs a cross-validation evaluation for different numbers of features and selects the number of features that resulted in the best mean score. We redefine the train and test set with the features selected by RFECV and calculate the final cross-validated error with RepeatedStratifiedKFold.

VII.2.2 Feedforward neural network

Another method to predict students' marks is to implement a neural network. Several Python libraries can be used for neural networks such as TensorFlow, PyTorch, Keras or scikit-learn. Here, we will rely on scikit-learn library, less powerful but simpler to implement in the time available. We will use MLPClassifier function, which stands for Multi Layer Perceptron Classifier and consists of a fully connected class of feedforward artificial neural network.

Without feature selection:

We begin by fitting a model without variables selection. We use the solver 'adam' and the 'relu' activation function in MLPClassifier. We use again the GridSearchCV function to find the optimal values for 'hidden_layer_sizes' (size and number of hidden layers), 'alpha' (strength of the L2 regularization term) and 'learning_rate' (learning rate schedule for weight updates) parameters in MLPClassifier. To determine a grid of possible values for the size and number of hidden layers, we rely on the book [34]. According to the author, two hidden layers can represent an arbitrary decision boundary and approximate any smooth mapping. Moreover, the author gives some rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers: the number of hidden neurons should be between the size of the input layer and the size of the output layer or 2/3 the size of the input layer, plus the size of the output layer. We will therefore use these rules to define the grid of values in GridSearchCV.

Once the best parameters are found, we calculate the cross-validated error with RepeatedStratifiedKFold.

Feature selection with Sequential Feature Selector:

Since the neural network does not give specific attributes of importance, the RFE function cannot be used and SFS from scikit-learn is applied instead. SFS stands for Sequential Feature Selector, and adds (forward selection) or removes (backward selection) features by sectioning at each stage, the best feature to add or remove based on the cross-validation score of the model. Unlike RFECV, there is no method to automatically determine the number of features. We use as a first guess, the same number of features as found previously with multinomial logistic regression. We specify 'backward' direction, 5 folds and 'f1_macro' scoring in SFS and redefine the train and test set with the selected features. Finally, we tune the neural network parameters with GridSearchCV, in the same way as above and compute the cross-validated error of the neural network defined with the selected features and the best paramters, with RepeatedStratifiedKFold.

VII.3 Mark Classification Results

We use the test set to evaluate the performance of the different models listed in section VII.2 (with and without feature selection). We consider several cases for the input explanatory variables:
- All the explanatory variables (features) are used.
- All variables are used except *mainSentiment* and *is0Sentiment* corresponding to the sentiment of the comments.
- All variables are used except *WorstProgress* and *timeFirstDecay* related to the supervisors' satisfaction.
- All variables are used except *mainSentiment*, *is0Sentiment*, *WorstProgress* and *timeFirstDecay* corresponding to the sentiment of comments and the satisfaction.

The results are presented in Table 6 and Table 7 in the case of 3 and 2 labels respectively (labels used in the sentiment analysis). The F1-score calculated on the train set in the Tables is computed with the function RepeatedStratifiedKFold.

		Computed on train set	Computed on test set	
model used	explanatory variables removed	F1-score	F1-score	Accuracy
Logistic Regression without feature selection	none	0.174	0.165	0.343
	sentiment analysis	0.181	0.168	0.323
	satisfaction	0.173	0.157	0.308
	sentiment analysis + satisfaction	0.176	0.184	0.358
Logistic Regression feature selection with Lasso	none	0.097	0.066	0.249
	sentiment analysis	0.097	0.066	0.249
	satisfaction	0.096	0.066	0.249
	sentiment analysis + satisfaction	0.096	0.066	0.249
Logistic Regression feature selection with RFE	none	0.251	0.224	0.363
	sentiment analysis	0.249	0.251	0.378
	satisfaction	0.247	0.208	0.358
	sentiment analysis + satisfaction	0.208	0.200	0.338
Neural Network without feature selection	none	0.091	0.039	0.065
	sentiment analysis	0.098	0.125	0.289
	satisfaction	0.101	0.076	0.184
	sentiment analysis + satisfaction	0.112	0.081	0.318
Neural Network feature selection with SFS	none	0.245	0.149	0.294
	sentiment analysis	0.252	0.151	0.338
	satisfaction	0.253	0.142	0.328
	sentiment analysis + satisfaction	0.216	0.140	0.333

Table 6: Results of the prediction for different models in the case of **3 labels**

We note that the results are globally unsatisfying, since regardless of the model used, the F1-score (less than 0.3) and Accuracy (less than 0.4) values are very low. The best configurations for each model have been highlighted in red.

In the case of 3 labels, the Logistic Regression feature selection with RFE model gives the best F1-score when removing the sentiment analysis explanatory variables. Therefore, it seems that sentiment analysis does not provide relevant information on the marks prediction since the models are all better when the explanatory variables corresponding to sentiment analysis are removed beforehand. On the other hand, satisfaction variables seems to be of interest as for almost all models (except Neural Network without feature selection), the F1-score decreases when the satisfaction is not considered compared to when all variables are considered. Finally, the variable importance in the Logistic Regression feature selection with RFE model when considering all

		Computed on train set	Computed on test set	
model used	explanatory variables removed	F1-score	F1-score	Accuracy
Logistic Regression without feature selection	none	0.175	0.180	0.353
	sentiment analysis	0.176	0.142	0.303
	satisfaction	0.162	0.179	0.343
	sentiment analysis + satisfaction	0.161	0.154	0.308
Logistic Regression feature selection with Lasso	none	0.097	0.066	0.249
	sentiment analysis	0.097	0.066	0.249
	satisfaction	0.096	0.066	0.249
	sentiment analysis + satisfaction	0.096	0.066	0.249
Logistic Regression feature selection with RFE	none	0.240	0.251	0.378
	sentiment analysis	0.246	0.251	0.378
	satisfaction	0.255	0.197	0.313
	sentiment analysis + satisfaction	0.210	0.199	0.338
Neural Network without feature selection	none	0.107	0.066	0.249
	sentiment analysis	0.101	0.066	0.249
	satisfaction	0.109	0.130	0.254
	sentiment analysis + satisfaction	0.081	0.066	0.249
Neural Network feature selection with SFS	none	0.245	0.112	0.259
	sentiment analysis	0.258	0.091	0.254
	satisfaction	0.235	0.174	0.333
	sentiment analysis + satisfaction	0.218	0.164	0.318

Table 7: Results of the prediction for different models in the case of **2 labels**

explanatory variables is: *nbMeetings*, *meanTimeBetweenMeet*, *meanTimeMeet*,*Supervisor*, *timeFirstDecay*,*module*, *worstProgress*, *mainSentiment* and *is0Sentiment*. We can therefore see that the sentiment analysis variables, ranked last, are the least important.

In the case of 2 labels, the best model is still Logistic Regression feature selection with RFE. For the Logistic Regression models, we observe that the F1-score decreases systematically when we remove the satisfaction variables so they seem to bring information to the students' marks, as seen in the case of 3 labels. Regarding the sentiment analysis variables, they seem to provide information for the Logistic Regression model without variable selection and have no influence on Logistic Regression feature selection with RFE and Lasso models. Neural network models seem to improve when satisfaction variables are removed, while removing sentiment variables worsens the models. Finally, the variable importance in the Logistic Regression feature selection with RFE model when considering all explanatory variables is: *nbMeetings*, *timeFirstDecay*, *module*, *worstProgress*, *mainSentiment* and *is0Sentiment*. Again, the sentiment analysis variables are ranked last.

Based on the above observations, we can assume that the sentiment analysis variables do not provide information on the students' marks. The variables that seem to influence the mark the most are:
- *nbMeetings*: The more meetings a student has with his/her supervisor, the more likely the student is to succeed in his/her project.
- *module*: some modules may be more difficult than others.
- *timeFirstDecay* and *worstProgress*: if a supervisor is quickly unsatisfied with the student, the student is less likely to succeed in the project.

However, these assumptions must be tempered as the models performed poorly in all cases.

VIII Critical review of the results obtained and areas for improvement

We have seen in the section VI that the supervisors' comments differ from the texts that we can find in the labelled datasets available on the internet. A tweet or product review tends to be short, sometimes containing acronyms and expressing a strong sentiment. In contrast, a teacher tends to write longer, more descriptive texts, using a more moderate vocabulary and sometimes implicit wording to address students. For example, the comment "the student did not show up" is seen as neutral by the algorithm while it is actually negative in the context of a supervisor's comment: the student did not show up for the project or meeting, resulting in a bad mark. Training our algorithm on large amounts of labelled supervisor comments may allow the model to detect these nuances, improving the predictions. We could also test other algorithms for sentiment analysis such as XLNet [6].

Concerning the prediction seen in section VII, the poor results could be explained by the fact that the algorithms used are not adapted to the problem or that the parameters have not been sufficiently optimised. In particular, due to lack of time, the neural network models were implemented with the Python library scikit-learn, which allows to build multilayer perceptrons in a simple way. However, scikit-learn does not allow the use of GPUs like the free Google Colab GPUs. The meta-parameters of the neural networks could therefore not be properly optimised due to the long computation time on a standard laptop. Testing other machine learning models for prediction and implementing neural networks with Python libraries using GPUs such as PyTorch or Tensorflow might improve predictions.

Finally, in the case where a model could correctly predict students' marks, an improvement would be to predict students' marks after only a few months (and not at the end of the year as done here) or to extract a typical profile of successful/failing students. This would allow students in difficulty to be detected as soon as possible and the supervisor to be alerted.

IX Conclusion

The internship consisted of working on a dataset extracted from a Student Dashboard used to supervise students from the School of Computing, DCU, in the conduct of their projects. The aim was to determine which variables influenced the students' marks, to determine the main sentiment of the supervisors' comments and to predict the students' final mark.

The section V gave us a first insight into the non-textual data from the dahsbord. We saw that the variables mostly influencing the student's marks are the *progress*, the *module* and the *Supervisor* while the variables *duration* for example have little effect on the marks.

In section VI, we first tested unsupervised NLP algorithms and then implemented the state of the art BERT algorithm giving better results than the unsupervised methods. We trained BERT in 2 phases: first on general datasets found on internet and then in a second phase on 200 labelled and augmented dashboard comments. We observed that training with two phases gave better results than training with only the first phase and we used the best parameters found in section VI.3.2 to predict the sentiment of the entire dashboard comments. We obtained rather satisfactory results with a F1-score of 0.80 in the 3-label case and 0.96 in the 2-label case.

Finally, we implemented multinomial logistic regression and neural networks algorithms to predict students' marks using the main sentiment of comments predicted by BERT and other computed features. We obtained unsatisfactory results with the best F1-score being 0.251 in the case 2 and 3 labels. Given more time, we could have tested other machine learning models for prediction or better optimise the parameters to improve the results.

On a personal level, this internship has been a very enriching experience for me. By going to Dublin, I had the chance to discover a new country and a new culture, to meet people from different backgrounds all having unique experiences to share and to improve my English. I was able to test my ability to adapt to a new environment that was completely unknown to me (new country, new language, new friends, working in a research centre). The assigned project, done independently with the guidance of my supervisor, helped me gain self-confidence, showing me that I was capable of searching for information on my own to solve problems. Finally, working in a research centre with research fellows and PhD students gave me an insight into how the research world works.

A Appendices

I First descriptive analysis on non-textual data

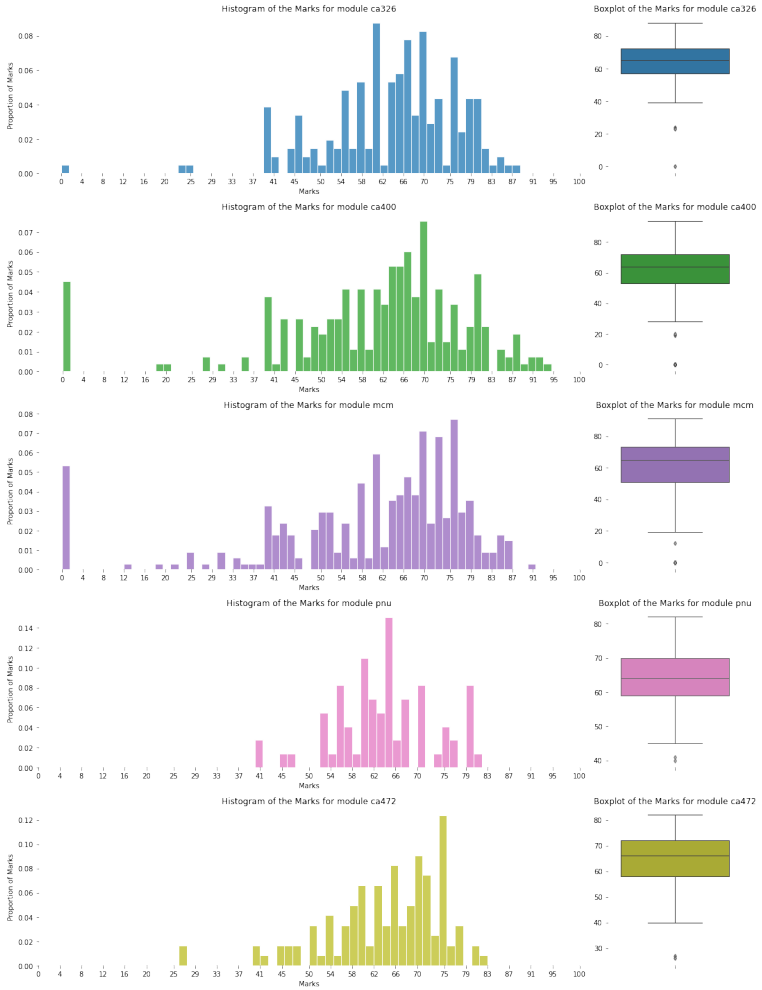
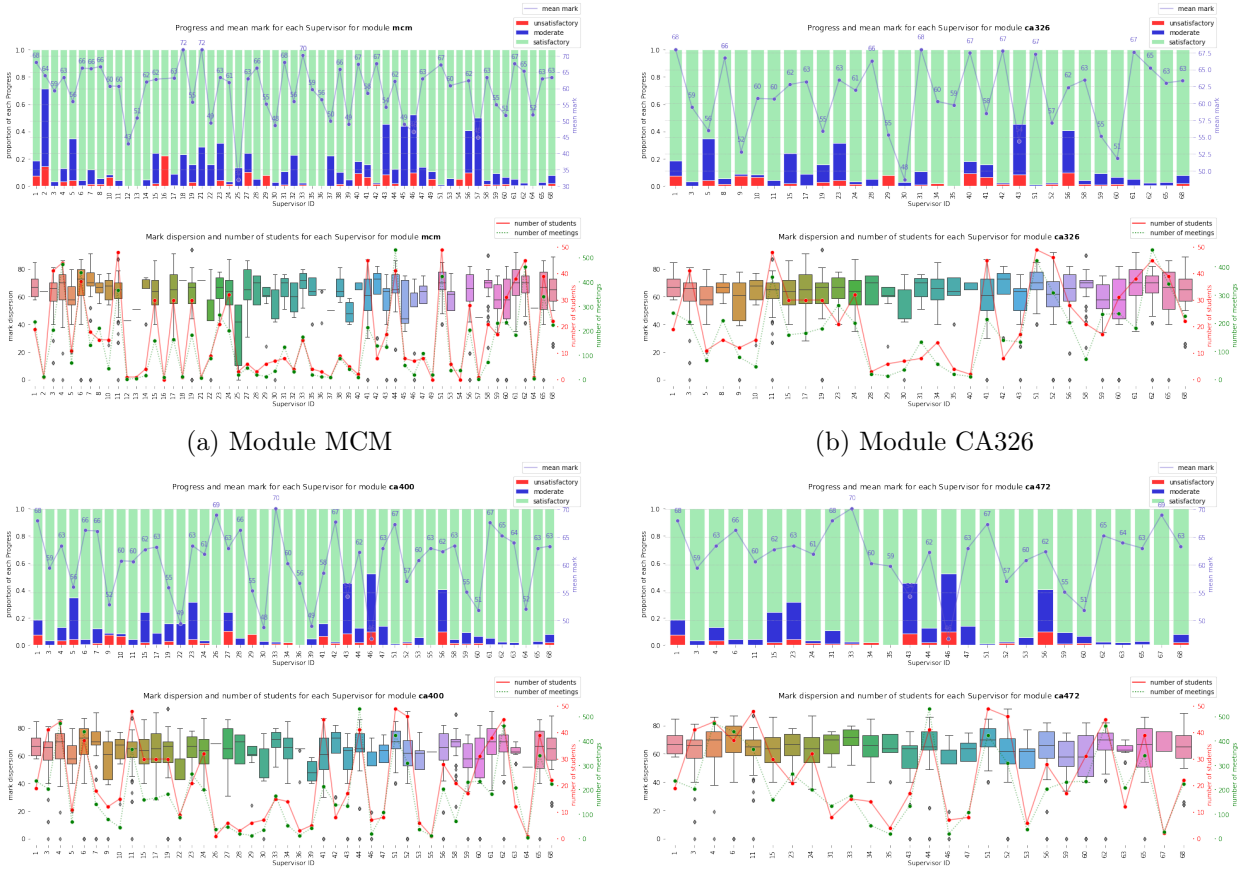


Figure A.1: Histogram and boxplot of the Students' marks for each module

II Prediction of students' marks



(a) Module MCM

(b) Module CA326

(c) Module CA400

(d) Module CA472

Figure A.2: Progress, mean mark, mark dispersion, number of students and number of meetings for each supervisor by module.

Higher Level Grade	Mark out of 100
H1	90-100
H2	80-89
H3	70-79
H4	60-69
H5	50-59
H6	40-49
N	0-39

Table A.1: Higher level academic grading in the Republic of Ireland

Bibliography

[1] "Adapt - sfi research centre for ai-driven digital content technology." https://www.sfi.ie/sfi-research-centres/adapt/. Online, accessed 17 September 2022.

[2] "About adapt." https://www.adaptcentre.ie/about/. Online, accessed 17 September 2022.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018. https://arxiv.org/pdf/1810.04805.pdf.

[4] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?," arXiv preprint arXiv:1905.05583, 2019. https://arxiv.org/abs/1905.05583. Online, accessed June 2022.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018. https://arxiv.org/abs/1810.04805. Online, accessed June 2022.

[6] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," arXiv preprint arXiv:1906.08237, 2019. https://arxiv.org/abs/1906.08237. Online, accessed June 2022.

[7] M. Cliche, "BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs," in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), (Vancouver, Canada), pp. 573–580, Association for Computational Linguistics, Aug. 2017. https://aclanthology.org/S17-2094. Online, accessed June 2022.

[8] M. Geetha and D. Karthika Renuka, "Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model," International Journal of Intelligent Networks, vol. 2, pp. 64–69, 2021. https://www.sciencedirect.com/science/article/pii/S2666603021000129. Online, accessed June 2022.

[9] L. Zhao, L. Li, and X. Zheng, "A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts," arXiv preprint arXiv:2001.05326, 2020. https://arxiv.org/abs/2001.05326. Online, accessed June 2022.

[10] S. Loria, "Textblob: Simplified text processing." https://textblob.readthedocs.io/en/dev/. Online, accessed 08 August 2022.

[11] S. Loria, "Github repository: sloria/textblob." https://github.com/sloria/TextBlob/blob/dev/textblob/en/en-sentiment.xml, commit 3079699 on 6 July 2014. Online, accessed 08 August 2022.

[12] A. Kulkarni, D. Chong, and F. A. Batarseh, "Foundations of data imbalance and solutions for a data democracy," arXiv preprint arXiv:2108.00071, 2021. https://arxiv.org/abs/2108.00071.

[13] C. Hutto, "Github repository: cjhutto/vadersentiment." https://github.com/cjhutto/vaderSentiment, commit 94fbd74 on 1 April 2022. Online, accessed 08 August 2022.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," arXiv preprint arXiv:1706.03762, 2017. https://arxiv.org/pdf/1706.03762.pdf.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Github repository: Google-research/bert." https://github.com/google-research/bert, commit eedf571 on 11 March 2020. Online, accessed 07 June 2022.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-Art Natural Language Processing," pp. 38–45, Association for Computational Linguistics, 10 2020. https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32 (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] C. Tran, "Tutorial: Fine tuning bert for sentiment analysis." https://skimai.com/fine-tuning-bert-for-sentiment-analysis/. Online, accessed 07 June 2022.

[19] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," arXiv preprint arXiv:1905.05583, p. 5, 2019. https://arxiv.org/abs/1905.05583.

[20] "torch.utils.data.dataloader class." https://pytorch.org/docs/stable/data.html. Online, accessed July 2022.

[21] "Crossentropyloss function." https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. Online, accessed July 2022.

[22] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arXiv preprint arXiv:1711.05101v3, 2017. https://arxiv.org/abs/1711.05101v3.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014. https://arxiv.org/abs/1412.6980.

[24] "Kaggle datasets." https://www.kaggle.com/datasets.

[25] H. Q. Abonizio, E. C. Paraiso, and S. Barbon Junior, "Toward text data augmentation for sentiment analysis," IEEE Transactions on Artificial Intelligence, 2021. https://ieeexplore.ieee.org/document/9543519, consulted on August 2022.

[26] E. Ma, "Github repository: nlpaug." https://github.com/makcedward/nlpaug, commit 23800cb on 7 July 2022. Online, accessed August 2022.

[27] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in Proceedings of the 25th International Conference on World Wide Web, (Republic and Canton of Geneva, CHE), p. 507–517, International World Wide Web Conferences Steering Committee, 2016. https://doi.org/10.1145/2872427.2883037. Dataset used in the paper cited above and retrieved on July, 20, 2022 from https://www.kaggle.com/datasets/meetnagadia/amazon-kindle-book-review-for-sentiment-analysis, file all_kindle_review .csv, Version 1.

[28] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts, version 4," Journal of the Association for Information Science and Technology, vol. 65, no. 4, pp. 782–796, 2014. Dataset used in the paper cited above and retrieved on July, 20, 2022 from https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis, file data.csv, Version 4.

[29] X. Zhang, "Text classification dataset (amazon_review_full_csv.tar.gz)," 9 sept 2015. Dataset retrieved on July, 20, 2022 from https://drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M?resourcekey=0-TLwzfR2O-D2aPitmn5o9VQ.

[30] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011. http://www.aclweb.org/anthology/P11-1015. Dataset used in the paper cited above and retrieved on July, 20, 2022 from https://www.kaggle.com/datasets/columbine/imdb-dataset-sentiment-analysis-in-csv-format, file Train.csv, Version 1.

[31] D. J. Stekhoven and P. Bühlmann, "MissForest—non-parametric missing value imputation for mixed-type data," Bioinformatics, vol. 28, no. 1, pp. 112–118, 2011. https://doi.org/10.1093/bioinformatics/btr597. Online, accessed August 2022.

[32] A. Bhattarai, "missingpy 0.2.0." https://pypi.org/project/missingpy/, Dec 10 2018. Online, accessed 20 August 2022.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. https://scikit-learn.org/stable/. Online, accessed august 2022.

[34] J. Heaton in Introduction to Neural Networks for Java, 2nd Edition, pp. 158–159, Heaton Research, Inc., 2008.