hyperrefOption ‘colorlinks’ set ‘true’ hyperrefHyper figures OFFHyperrefLink nesting OFFhyperrefHyper index ONhyperrefPlain pages OFFHyperrefBackreferencing OFF hyperrefImplicit mode ON; LaTeX internals redefined hyperrefBookmarks ON

hyperrefHyper figures OFFhyperrefLink nesting OFFhyperrefHyper index ONhyperrefbackreferencing OFFhyperrefLink coloring ONhyperrefLink coloring with OCG OFFhyperrefPDF/A mode OFF

hyperrefDriver (autodetected): hpdftex rerunfilecheckFeature “pdfmdfivesum is not available(e.g. pdfTeX or LuaTeX with package ‘pdftexcmds’).Therefore file contents cannot be checked efficientlyand the loading of the package is aborted

# Student Performance Analysis and Prediction
# Final project for the Foundations of Data Science course (a.a. 2023/2024)

Paolo Cursi 2155622, Tommaso Leonardi 1914546,
Arianna Paolini 1943164, Stefano Saravalle 1948684, Pietro Signorino 2149741

December 26, 2023

**Abstract**

Academic success is relevant to students satisfaction and consequent commitment in studying, leading them to become well formed professional figures. This project aims to understand which are the factors that influence the most the scores that the students get in their exams, by considering the case of the British Open University. Linear and non-linear machine learning models have been trained to predict the performance of the students in various assessments.

## 1 Introduction

The Open University is a public university in England that provides data about demographic information and academic performance of its students in order to allow learning analytics. We based our project on *The Open University Learning Analytics dataset*, which is also available on Kaggle (https://www.kaggle.com/datasets/rocki37/open-university-learning-analytics-dataset).

We were interested in this dataset as it provides data about students interaction with the *Virtual Learning Environmment* (*VLE*), an online platform holding teaching materials and resources for the university courses. This could be an opportunity to test how much new technologies can help the learning process. That, combined with the availability of information about the social status of the students (age, gender, disability, education level, etc.) and their academic career (scores in assessment, number of studied credits, etc.) makes the Open University dataset a good choice for our purposes.

We selected different machine learning models to implement the prediction of students scores in assessments, both in the form of a regression and a multinomial classification task (by dividing the students according to some thresholds on the score range): *linear regression*, *decision trees* and *KNN regression* will be used for regression, *logistic regression* and *neural networks* for classification. Having multiple trained models for each task allow us to compare the results, determine which model performs best and try to understand why. The code of our work can be found at https://github.com/twgever/FDS_Project.

## 2 Dataset

The Open University Learning Analytics dataset contains data about 22 courses with 32,593 registered students [**?**]. It is organized in seven relational tables, as show in Figure 1:

- **courses**: includes the list of all available courses (which are called *modules*) and their *presentations* (i.e. the occurence of a course in a specific academic year). The columns are:
  - *code module*: code name of the module, which serves as the identifier;
  - *code presentation*: code name of the presentation, which consists of the year and "B" for the presentation starting in February or "J" for the presentation starting in October;

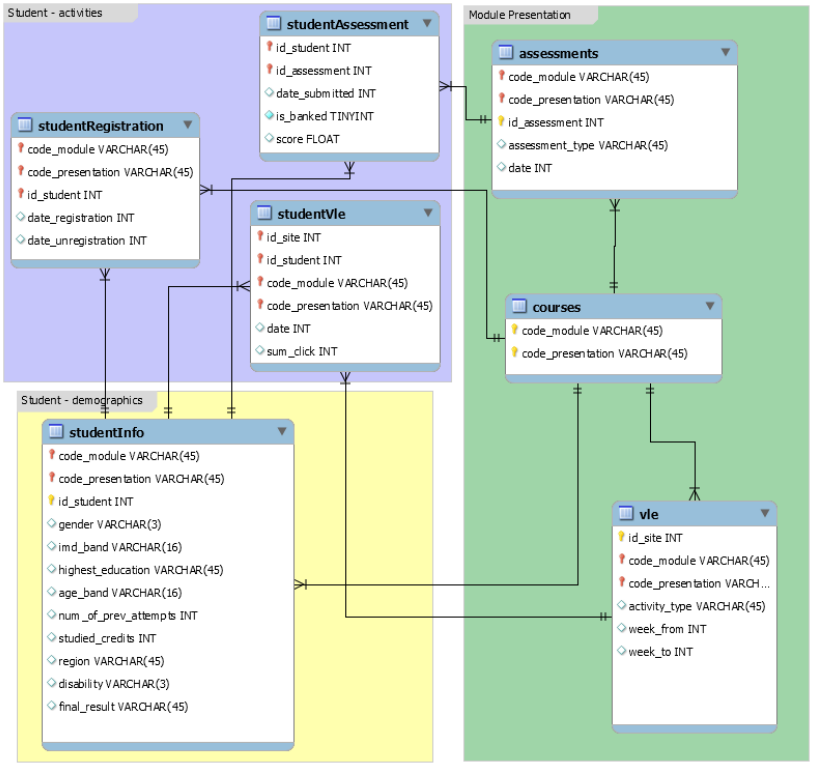



Figure 1: The organization of the tables in the Open University Learning Analytics dataset (https://analyse.kmi.open.ac.uk/open_dataset)

  - *length*: length of the module-presentation in days.

- **assessments**: contains information about assessments in module-presentations, which are typically a number of assessments followed by the final exam. The columns are:
  - *code module*: identification code of the module to which the assessment belongs;
  - *code presentation*: identification code of the presentation to which the assessment belongs;
  - *id assessment*: identification number of the assessment;
  - *assessment type*: type of assessment, it can be: Tutor Marked Assessment (TMA), Computer Marked Assessment (CMA) or Final Exam (Exam);
  - *date*: information about the final submission date of the assessment calculated as the number of days since the start of the module-presentation (the starting date of the presentation has number 0);
  - *weight*: weight of the assessment in percentage: final exams are treated separately and have weight 100%, the sum of all other assessments is 100%.

- **vle**: includes data about the available materials in the VLE (html pages, pdf files, etc.); students interactions with the materials are recorded. The table contains the following columns:
  - *id site*: an identification number of the material;
  - *code module* : an identification code for module;
  - *code presentation*: an 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.

- **studentInfo**: contains demographic information about the students together with their results. The columns are:
  - *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.
- **studentRegistration**: contains information about the time when the student registered or unregistered for the module presentation. The colums are:
  - *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, as 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 (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* table).
- **studentAssessment**: shows the results of students' assessments. If the student does not submit the assessment, no result is recorded. The final exam submissions is missing if the result of the assessments is not stored in the system. It contains the following columns:
  - *id assessment*: the identification number of the assessment;
  - *id student*: a unique identification number for the student;
  - *date submitted*: the date of student submission, measured as the number of days since the start of the module presentation;
  - *is banked*: a status flag indicating that the assessment result has been transferred from a previous presentation;
  - *score*: the student's score in this assessment; the range is from 0 to 100. A score lower than 40 is interpreted as Fail.
- **studentVle**: contains information about each student's interactions with the materials in the VLE. The columns are:
  - *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 VLE material;
  - *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.

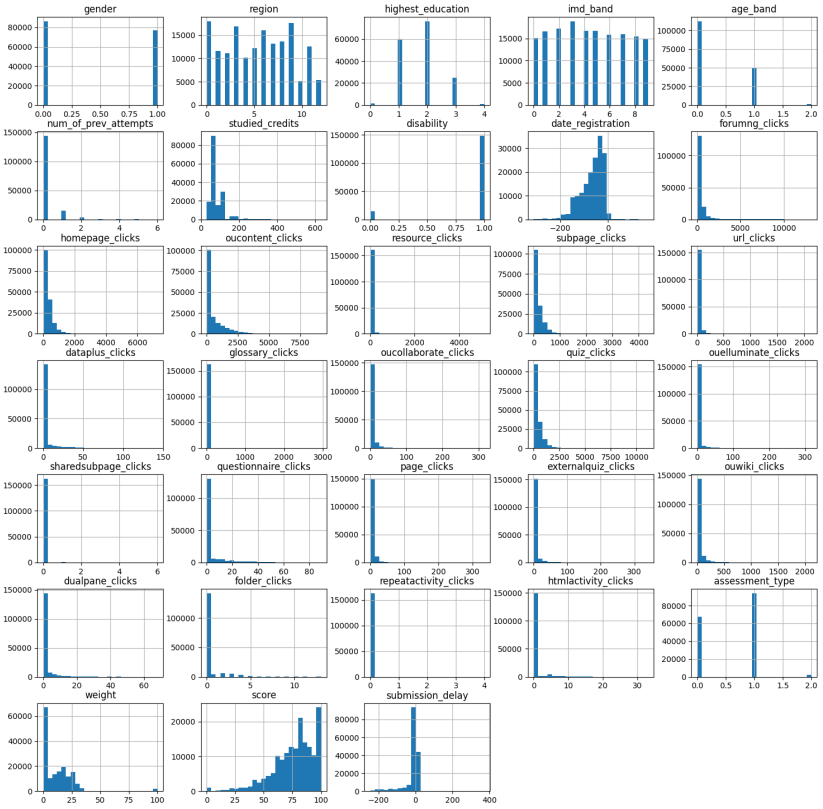



Figure 2: Histograms showing the data distribution in our dataset

# 3 Data Preprocessing

## 3.1 Feature selection

Since our goal was to predict the score of some student in any assessment during a module-presentation, we considered the *score* column in the *studentAssessment* table as our target variable. As for the features, we selected the columns resulting from the join of the *studentInfo*, *studentRegistration*, *studentAssessment*, *studentVle*, *assessments* and *vle* tables. We excluded the *final result* column of *studentInfo* to avoid trivializing the score prediction task and we ignored some columns that seemed less relevant to the student career (such as *is banked* from *studentAssessment*, *date* from *studentVle*, *week from* and *week to* from *vle*).

While joining *studentVle* and *vle* we decided to aggregate the number of clicks of a student on an online resource on the VLE for some module-presentation, by considering the total sum of clicks on each different type of resource (which is specified by the *activity type* attribute), in order to allow a deeper analysis on the VLE materials contribution to academic success.

We also decided to merge together the *date submitted* (from *studentAssessment*) and *date* (from *assessments*) attributes into a single *submission delay* feature, computed as the difference of the two, thus considering the time interval between the publication of an assignment and the student submission.

After dropping the *id* columns of the joined tables and deleting the rows corresponding to students with important information missing, we got a dataset of 163,387 rows $\times$ 33 columns. We normalized the features values by subtracting the mean and dividing by the standard deviation of each column, thus getting the data distribution shown in Figure 2.

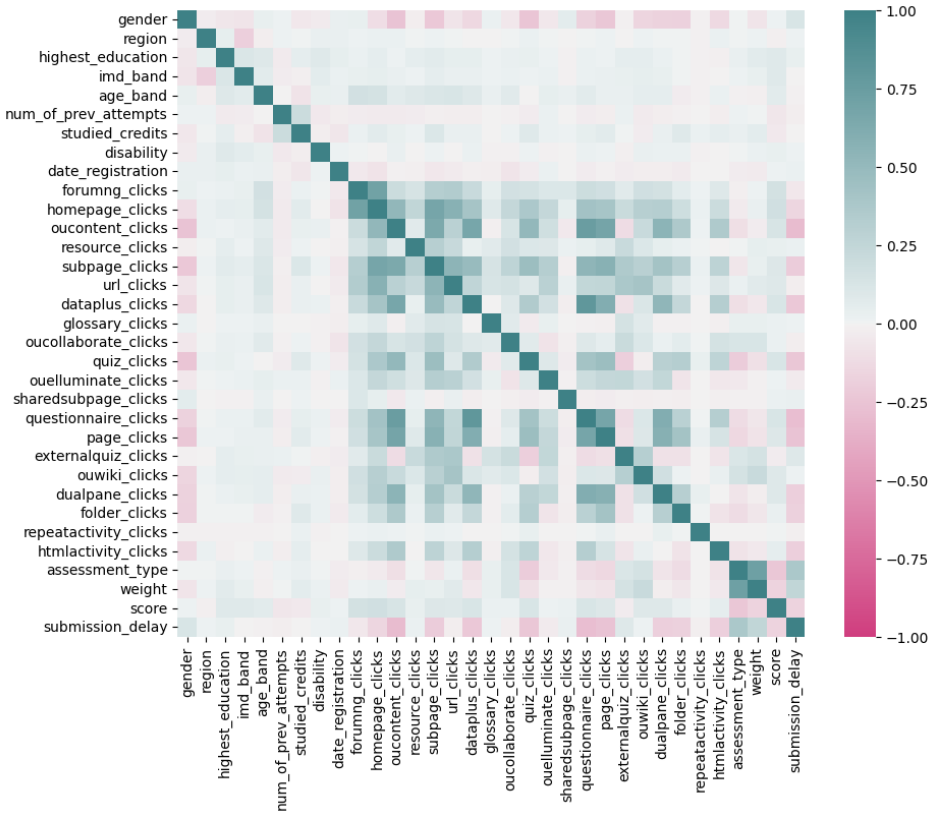



Figure 3: Correlation matrix for the dataset features

## 3.2 Data analysis

We computed the correlation matrix for the features of the dataset (Figure 3) and observed that there exist some positive correlation between the clicks on different types of online resources (e.g. *homepage clicks* and *forumng clicks*, *questionnaire clicks* and *dataplus clicks* or *oucontent clicks*), probably because they are often accessed sequentially during a single VLE session. There is also a slight negative correlation between the *submission delay* and the number of clicks on some type of resources (e.g. *oucontent*, *quiz*, *questionnaire*), suggesting that a student who is active on the VLE is more likely to submit an assignement with a small delay than a student who don't access online materials.

We also looked at some scatterplots expressing the relationship between each feature and the target variable (Figure 4), noticing that in some cases the feature don't seem to influence the score value (e.g. *gender*, *region*, *imd band*), whereas other plots show that bigger feature values are associated to larger scores (e.g. *num. of prev. attempts*, *forumng clicks*, *questionnaire clicks*). We count on the latter type of features for the effectiveness of our linear ML models.

Finally, we observed that the dataset contains much more information about students who got high scores in assessments than about students with worse exam outcome (Figure 5). Although having many students with high grades is good news, this might interfere with the correct prediction of low scores. We tried to counter the problem with oversampling (Section 5).

## 3.3 Train and test split

We chose to compute the training and test sets for our ML models by splitting the dataset with the *train_test_split* function from the *scikit-learn* library to use 80% of the original number of samples for training (about 130,700 rows) and the remaining 20% for testing (about 32,600 rows).

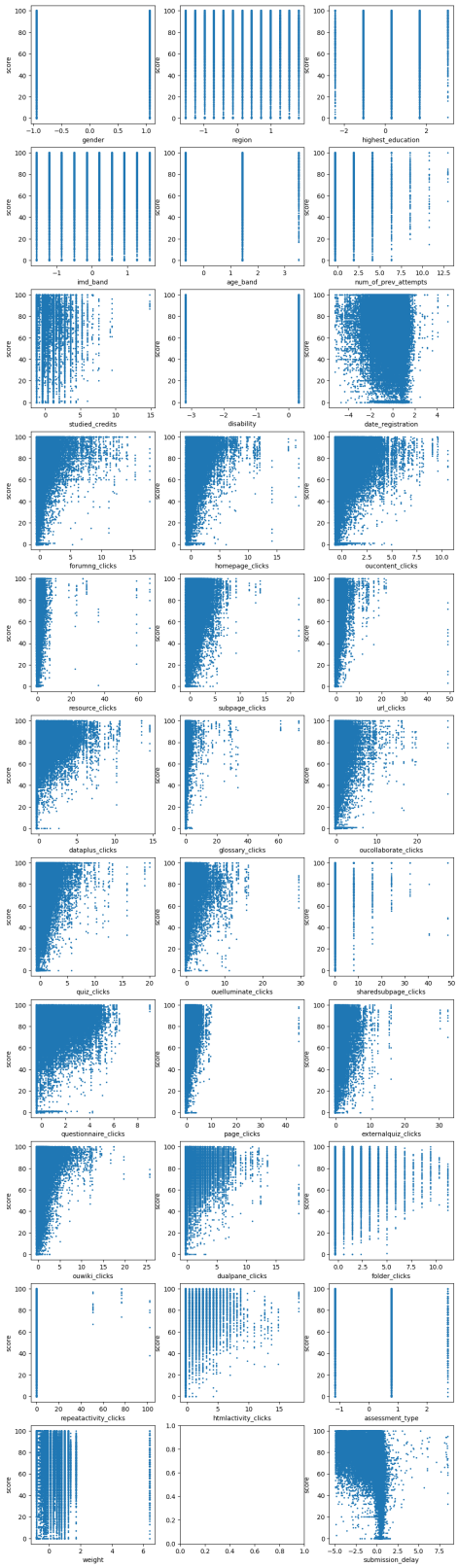



Figure 4: Scatterplots showing the relationship between each feature and the target variable

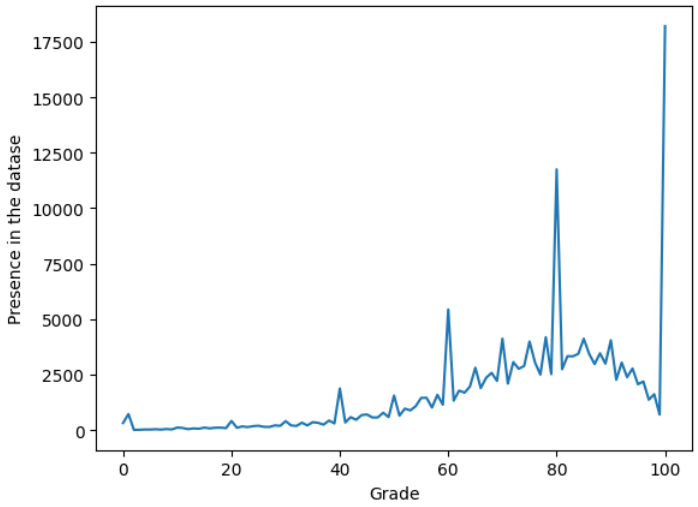



Figure 5: A graph showing the number of dataset rows for each score value

# 4 ML models training and testing

We leveraged the *scikit-learn* Python library to implement some ML models for the regression and classification tasks of predicting the score of a student in a module-presentation assessment. We trained linear (*linear regression*, *logistic regression*) and non-linear (*decision trees*, *KNN*, *neural networks*) models to see if there are some differences in the results they provide.

We evaluated our regression models by computing the *RMSE* and *R2* scores, while we considered the *accuracy* for our classification models.

## 4.1 Linear regression

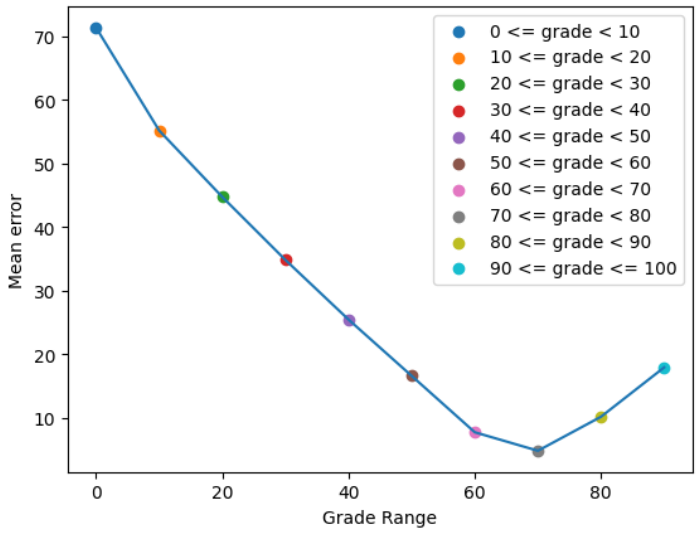



Figure 6: A graph showing the mean error for 10 ranges of grades

Linear regressin is a simple ML model. It's a linear model so there's no risk to overfit data.

We want to predict the score of a student, we use the *scikit-learn LinearRegression* class (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html). It doest have any relevant hyperparameters. We have reached a RMSE of about 17.57 and a R2 score

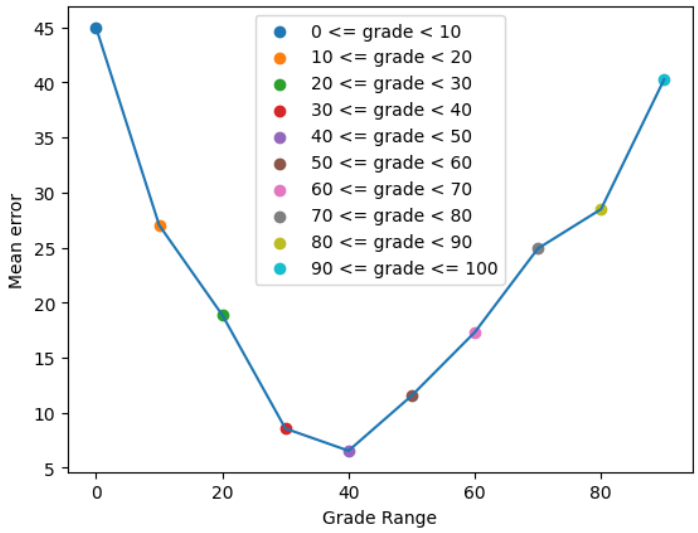



Figure 7: A graph showing the mean error for 10 ranges of grades on the oversampled set

of 0.11 on the test set and similar values on the trainint set since it can't overfit data. As we can from Figure 6, the model makes bigger mistakes when trying to predict lower scores (Figure 10), due to the big gap between the number of students with high and low scores that are present in the dataset.

### 4.1.1 Oversampling

Because of that we tried to do oversampling on our dataset using the sklearn function *RandomOverSampler*, now we have the same ammount of data for each grade. We got a RMSE of about 29.95 which is higher than the not oversampled vertion, as we can see on the Figure 7 we have a better accuracy on the lower grades but a worst accuracy on the higher.

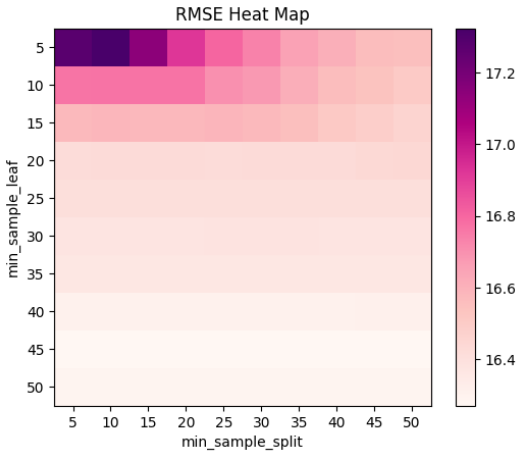
## 4.2 Decision Trees

Decision trees are a simple ML model that should be able to deal with the non-linear relationship that some features show with the target variable. However, they might be prone to overfitting, as they are built by splitting the training samples according to their values for the most important features.

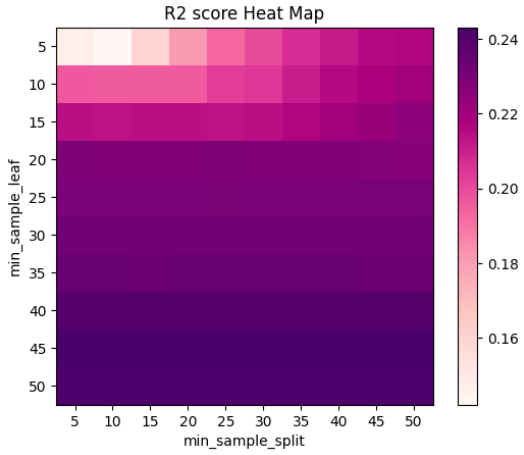
Since we aim to predict the exact score of a student in an assessment, we use the *scikit-learn DecisionTreeRegressor* class (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html). Such implementation of decision trees has many hyperparameters, but we selected *min_samples_leaf* and *min_samples_split* as the the most significant ones: the former determines the minimum number of samples required to be at a leaf node, the latter specifies the minimum number of samples required to split an internal node. So these parameters influence the size of the decision tree and the extent of overfitting.

Some hyperparameters tuning (Figure 8) was done to find out the values that allow to reach the best performance: with *min_sample_leaf* set to 45 and *min_sample_split* set to any value between 5 and 50 we can reach an RMSE score of about 16.36 and an R2 score of 0.23 on the test set. Since the scores for the prediction on the train set were $RMSE = 15.01$ and $R2 = 0.36$ the model doesn't seem to overfit too much.

By visualizing the first levels of the decision tree with the best hyperparameters values (Figure 9) we understand that the features that are more relevant to the score prediction are *assessment type*, *homepage clicks*, *weight*, *submission delay*, *quiz clicks*, *subpage clicks*, as they are the first features chosen to split the train set on.




[RMSE heatmap]




[R2 score heatmap]

Figure 8: Heatmaps showing the RMSE and R2 values for different choices of *min_sample_leaf* and *min_sample_split*

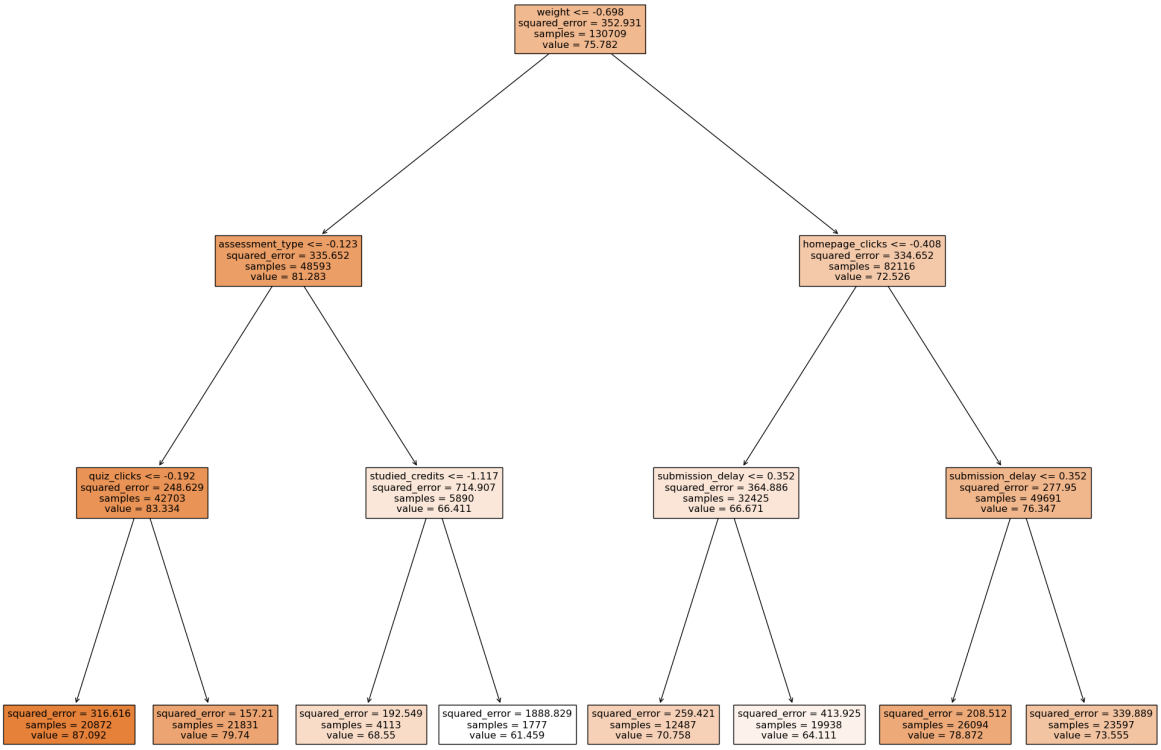



Figure 9: The first 4 levels of the DecisionTreeRegressor with *min_sample_leaf* set to 45 and *min_sample_split* set to 10

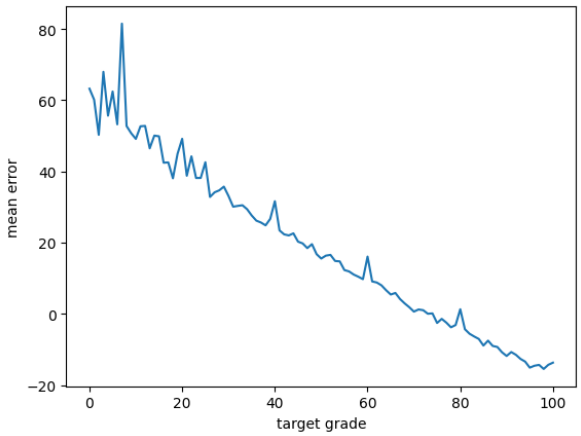



Figure 10: The mean error for each real score that the model tries to predict

As we expected, the model makes bigger mistakes when trying to predict lower scores (Figure 10), due to the big gap between the number of students with high and low scores that are present in the dataset.

## 4.3 KNN Regression

## 4.4 Logistic regression

## 4.5 Neural Networks

Neural networks are one of the most complex used model in this report. The first thing that we did was to choose a good network topology, and for this reason we tried with many different layers.

We found that the network physical composition wasn't this important for the predictions, since each of the attempts produced more or less the same result, so, to mantain things at a simple and manageable state, we opt for a very simple network, with just two layers and an actiavtion function (ReLU) in between.

The first layer had the numbers of features in input, and in output it had the same number multiplied by two. The last layer had the duty to predict the number of classes, so either ten or two in our experiments.

Since we want to predict, given the features of a student, the probability that he is gonna fall in the ranges of score we have defined, outside the last layer there is a *Softmax function*, which is in charge of parsing the outputs in probabilities. Understood what we said until this point, it's obvious that the loss function could not be anything other than a *Cross-Entopy*.

Regarding the training, it was done using the following hyperparameters:

- Defined our custom dataset, we used a **batch size** of 16.
- We choose a maximum of **100 epochs**, even if we didn't ever reached the limit in the training as we are gonna explain later on.
- We tried different **learning rates**, ranging from $10^{-1}$ to $10^{-5}$.

The train was done with the idea of stopping it when the loss was less than $5 \times 10^{-5}$. It can be clearly seen from the following graphs, that the training was almost every time stopped earlier, especially for the smallest learning rates, that made the network change a very little for each epoch.

To provide a summary about what we said until now, we can look at the following graph:

The dataset chosen was afflicted by a misrepresentation problem, with more student obtaining good results and less with lower scores. For this motivation, we tried to narrow down the number of classes, trying with just two, so higher than forty or lower.

We also changed the structure of the network, replacing the Softmax with a *Sigmoid function*.

Here we can look at the results:

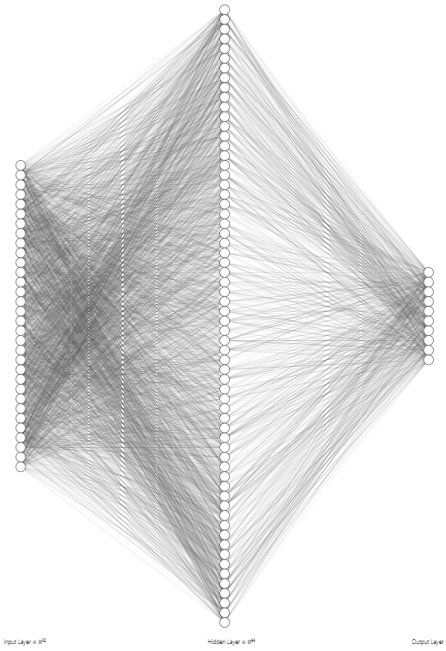


Figure 11: The neural network architecture we used.

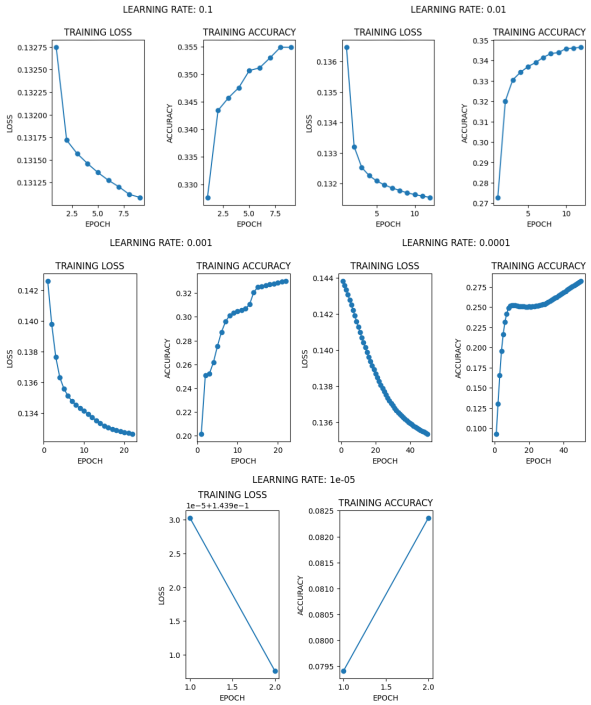



Figure 12: The training we did with the different learning rates.

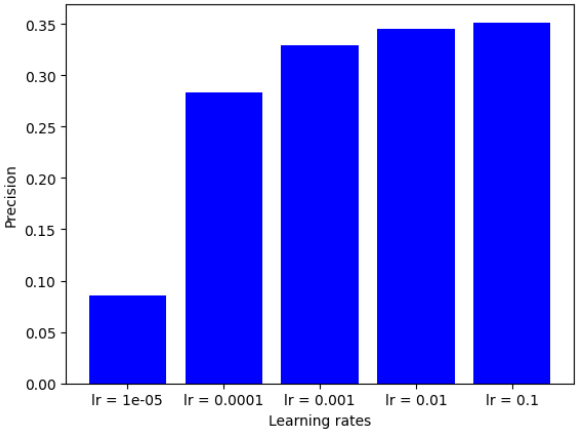



Figure 13: Summary about precisions.

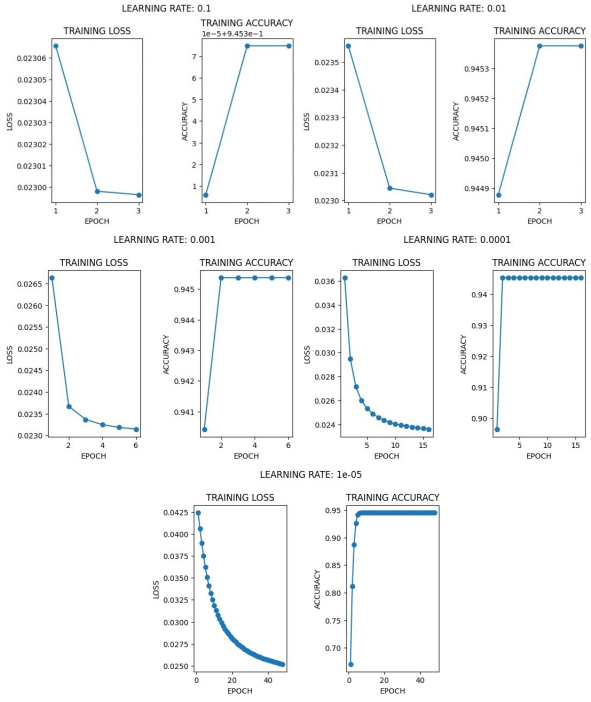



Figure 14: Training with different learning rates with binary classification

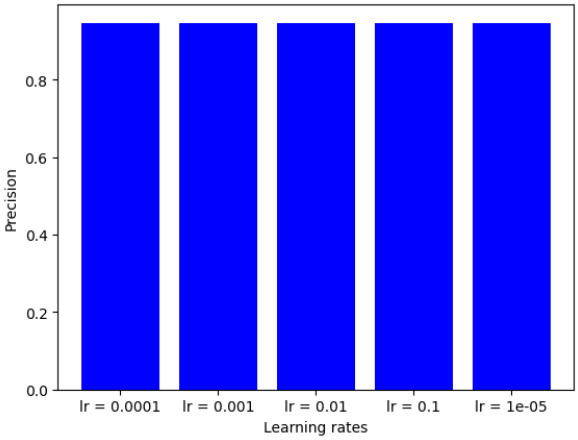



Figure 15: Summary of the results in binary classification.

And here we can see the following graph providing a summary on the results obtained. The misrepresentation problem appears again, for this motivation, the network just has to predict a value closer to one, rather than learning from the features, since a lots of examples in the dataset are higher than forty.

# 5 Oversampling

# 6 Results analysis

# 7 Roles of team members

We worked together on the dataset choice, data preprocessing and analysis, then each of us focused on a different ML model:

- Paolo Cursi: Linear Regression
- Arianna Paolini: Decision Trees
- Pietro Signorino: KNN Regression
- Tommaso Leonardi: Logistic Regression
- Stefano Saravalle: Neural Networks

Finally, we compared our results and discussed about them to understand what we learnt from this project.