Educational data mining

Project work - Group JYUEF01

Giulia Ortolani, Johanna Riihimäki

April 11, 2024

Abstract

In this assignment, educational data mining (EDM) techniques were used to analyze Learning Management System data. With classification and regression methods, the analysis examined the correlation between student activity on the course area and final scores. The main goal of this project was to analyze the given data and create different models for estimating the student's score on the course to which the data is related. The analyzed data consist of timestamps when the student has interacted in the course and the total score. The idea is to study the data and estimate can we find any connection between the timestamp data and the score from the course. The dataset contained 150 different students' timestamps of activity, out of which features like course completion time and preferred working time of day were extracted. Decision tree classification is utilized to predict student grades categorized into scales of 0-5 and 0-3, while linear regression model is utilized to predict the exact score. Both these methods had trouble predicting the lower scores, but were quite good in predicting high scores.

Contents

1 Introduction

Educational Data Mining (EDM) is the method of applying data mining in the educational field, in order to (for example) better the existing practices and provide support for decision making (Aldowah et al. (2019)). The data for this is plenty, thanks to things like Learning Managements Systems (LMS) or Massive Open Online Courses (MOOC) that collect massive amounts of usage data (Behdad Bakhshinategh (2018)).

The goal of the EDM is to understand how the students learn, so we could create better course arrangements or materials, use more effective learning methods or create more personalized learning environments. In EDM the data can be analyzed with data mining techniques, statistical models or machine learning. Some of the most frequently used methods in EDM are regression, clustering and classification (Behdad Bakhshinategh (2018)). In our analysis, we chose to use classification and regression.

Classification involves two main steps: training the model with one dataset to learn patterns, then testing its accuracy with another. Decision tree classifiers are popular because they are simple to implement and the results are easy to understand, even for humans.(Bunkar et al. (2012)) **Regression**, on the other hand aims to predict how one or more independent variables are related to a dependent or target variable using statistical methods.

1.1 Dataset

The data that we have consists of timestamps when a student has interacted with the course and the final score for each student. We assume that every data point with the same score is associated to the same student. Therefore, we were able to identify 150 different students, whose scores range from 1055 points to 7594 points. We can see in the Figure 1 the point distribution is a bit unusual, most of the students collected very high scores. The median value is 6945. We anticipate this skewing to cause problems later with the score estimation, since there might not be enough data points in the mid and low scores to produce reliable results.

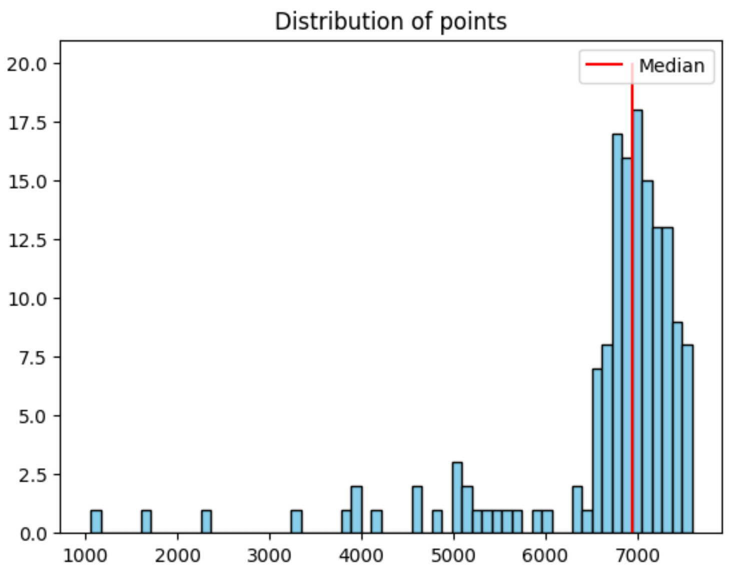


Figure 1: Distribution of scores

1.2 Hypothesis

After having a first look to the data, we started doing hypothesis about their distribution. In particular, we wondered if a student with more active days or more logs would get a better score. That is, being consistently active will lead to a better learning and consequently to a better score at the end of the course. Also, we thought that the preferred time of day for working on the course might have an effect on the score.

1.3 Extracting features

We started pre-processing our dataset, the features that we extracted are:

- `comp_t`: the time each student took to complete the course. It's the difference between the last and the first log of the student. This is a floating point number, since the difference was turned into seconds, and then into days by dividing it by $60 \cdot 60 \cdot 24$. After that, +1 was added to the final result to prevent division by zero[1]. The range of this variable in the dataset is 1.0-106.04.
- `logs`: the number of different timestamps that we have for each student, i.e. the number of total logs the system has registered of each student. The range of this variable in the dataset is 1-112.
- `pref_work_time`: the time of day in which a student tends to work (time of the day when the student has most time stamp logs). This is a categorical variables with three possible values:

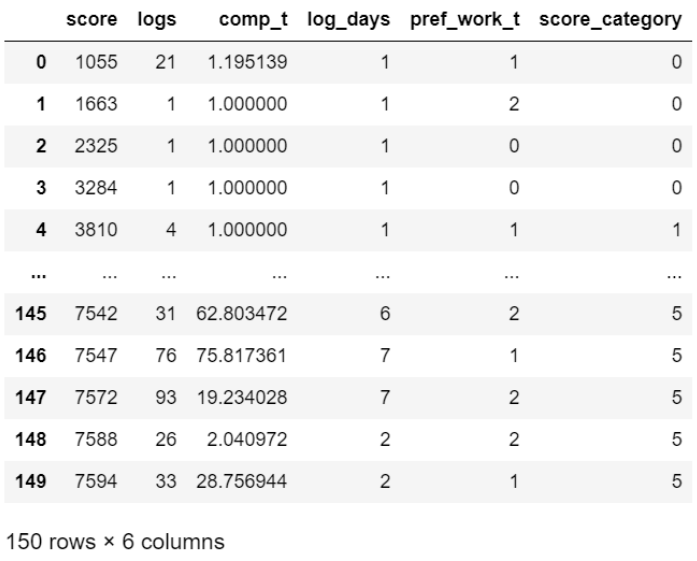
[1]This was done as a temporary fix to avoid division by zero, which became an issue later. It also does make sense: if you only have one log in the course, you have (technically) worked on the course one day, not 0 like the code would say. It makes less sense with values other than 0. In hindsight, perhaps this would have been smartest to be an integer also. *"There's nothing more permanent than a temporary solution." - Einstein, probably*

Value	Time Period	Time Range
0	Morning	06:00 - 12:00
1	Day	12:00 - 19:00
2	Night	19:00 - 06:00

- `log_days`: the amount of days in which the student has been active in the course area. This is an integer, the range in the dataset is 1-16.

Together with those, we had the following feature from the original dataset:

- `score`: the score that each student obtained.[2] Assuming that every student had a different score we could obtain the number of total students (150).



	score	logs	comp_t	log_days	pref_work_t	score_category
0	1055	21	1.195139	1	1	0
1	1663	1	1.000000	1	2	0
2	2325	1	1.000000	1	0	0
3	3284	1	1.000000	1	0	0
4	3810	4	1.000000	1	1	1
...	...	...	...	...	...	...
145	7542	31	62.803472	6	2	5
146	7547	76	75.817361	7	1	5
147	7572	93	19.234028	7	2	5
148	7588	26	2.040972	2	2	5
149	7594	33	28.756944	2	1	5

150 rows × 6 columns

Figure 2: The data

The data was collected into a `pandas.DataFrame`, a screenshot of which can be seen in Figure 2. The last variable, `score_category`, has not yet been introduced, it is an estimated grade (according to Finnish grading system, 1-5 being the passing grades, 5 being the best and 0 fail) based on the score. Later on, another score category was added. More on that in subsection 2.2 .

1.4 Data visualization

In Figure 3 we can see that we don't have any obvious correlation between these variables: the higher scores are associated with almost every value of logs, high or low.

[2]Disclaimer: We only noticed the "few words about data" less than 24 hours before deadline. When completing this assignment, we proceeded with the assumption that the dataset encompassed the entire course area, not solely one event. While this might have slightly influenced our interpretation, the analyses we conducted remain valid.

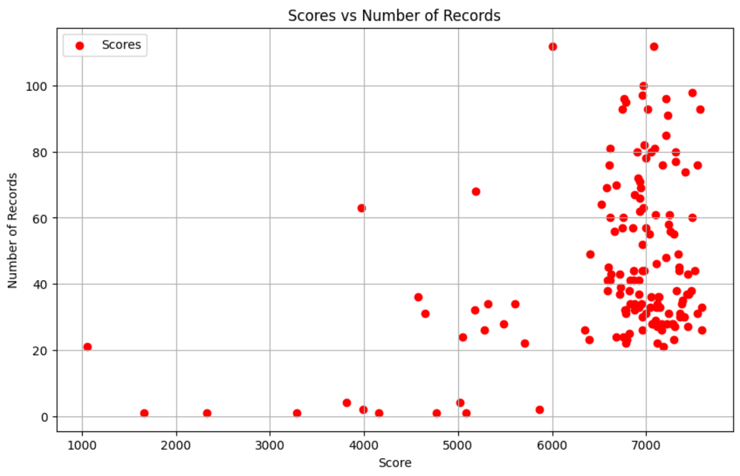


Figure 3: Scores vs Number of Records

In Figure 4 the data is plotted in more detail. The horizontal blue lines in all four graphs have the same y-value. The lower line represents the ”passing line”, meaning that any scores beneath it are equal to grade 0 and those above (or on) it are passing. The upper line represents the line between grades 3 and 4; scores above (and on the line) are graded as 4 or higher, and scores below it are graded at most 3.

Again, we can see that most of the data points are collected in the upper regions and it looks like a lot of the students were able to get good scores with relatively little time put into the course. Based on the plots, it is not obvious that the amount of work a student puts in reflects directly in their score, which is contrary to our presumptions. There seems to be a lot of students, who got a very high score within a very small time window. However, we can see that putting in more work (say at least a few days) will lead to at the minimum a passing grade, likely better. It would also seem that the amount of logs doesn’t have a huge effect on the score, but having more logs (40-ish and over) makes it it more likely to not get a grade less than 4. The preferred time of working (morning, day, night) doesn’t seem to have any distinguishable connection to the score.

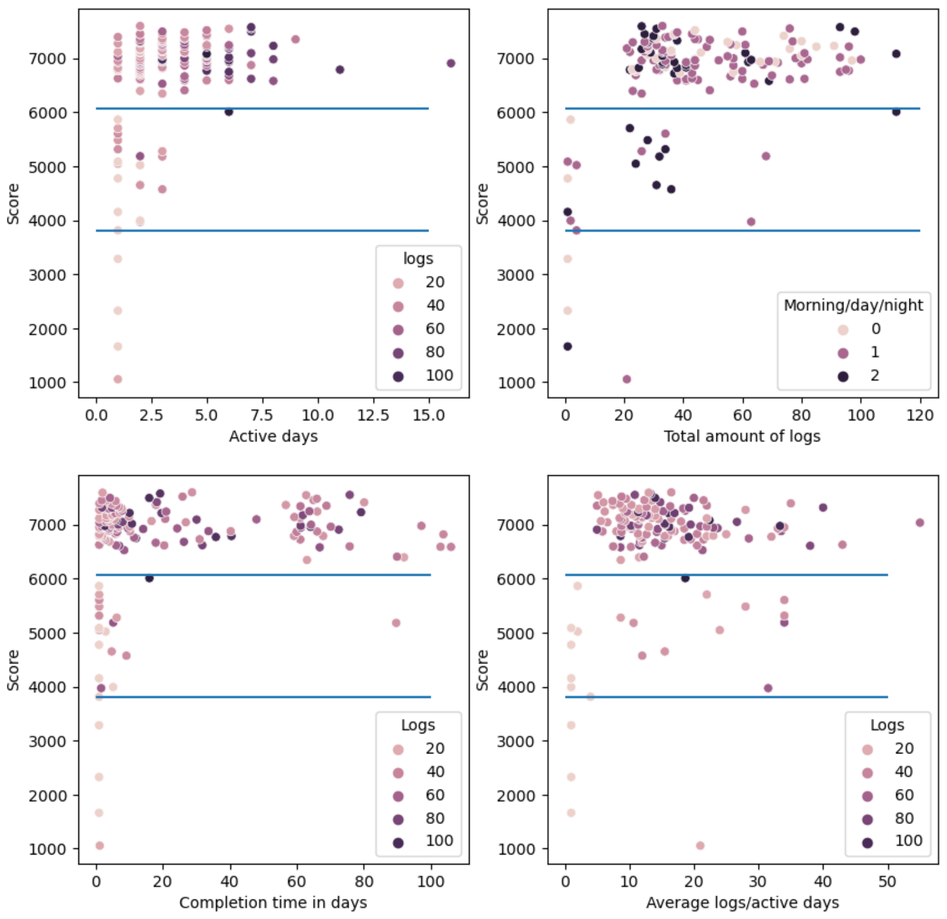


Figure 4: More detailed visualization of the data

2 Data manipulation

2.1 Filtering

We actually ran one model on the data without filtering and the results were not good. This is not surprising, given the data distribution. We decided to try and filter the data very conservatively based on the following logic: let's assume this course is worth at least 1 credit point, meaning an average student would have to work on the course about 27 hours (for a grade 5). Let's be generous and assume the student can work 10 hours/day effectively. It would take an average student about 2-3 days of active work to complete the course. Based on this it wouldn't be likely that a student would get a high grade with less than 2-3 days of active work. Therefore, we filter out those entries that have gotten a score 4 or 5 with only two days (or less) of active work. This reduces our dataset by a third, we are left with 102 data points. This is not ideal, but since our problem is relatively simple, we believe the dataset is still large enough for the purposes of this assignment.

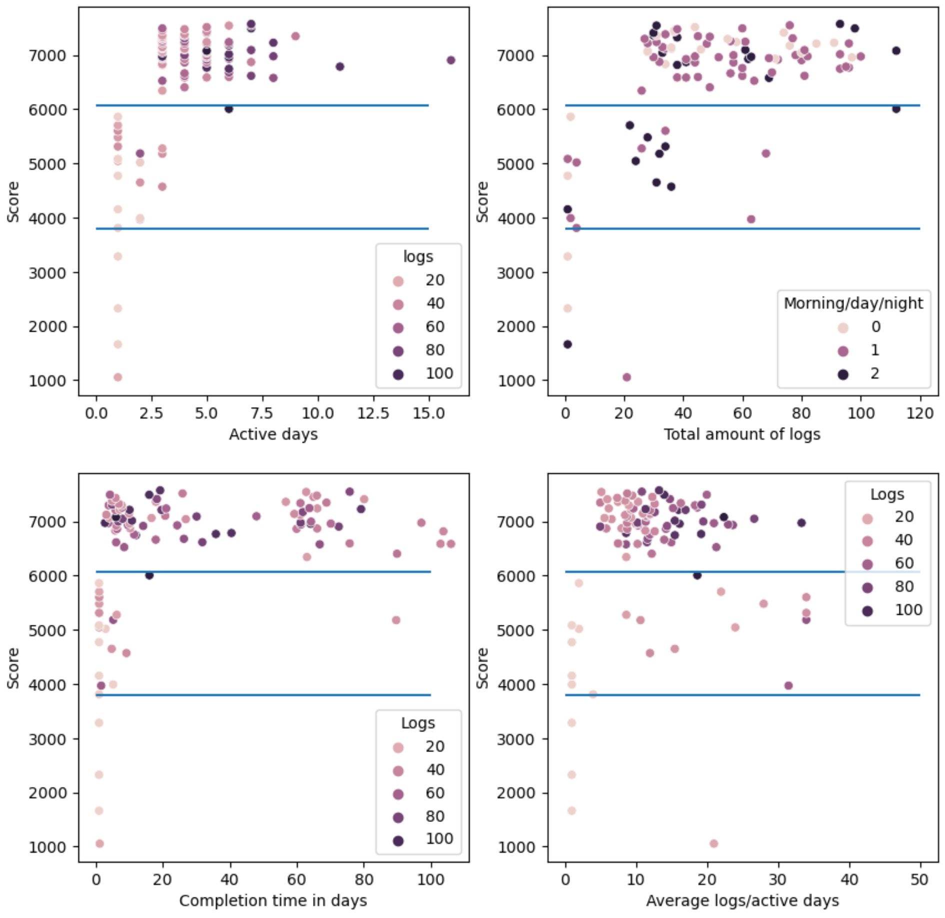


Figure 5: Filtered data visualizations

2.2 Categorization of the grades

In order to get something constructive out of the decision tree algorithms, we had to categorize the scores into smaller subgroups. This way the algorithm is not able to predict an exact point score, but it should provide an estimation of the point range the student's score would fall into. We categorized the points in two ways, each introduced below.

We decided to try two different categories to see which approach works best for our data. One method involved categorizing scores into ranges from 0 to 5, while the other method used ranges from 0 to 3. We introduced the 0-5 range because it aligns with the canonical way of grading in Finland. As for the 0-3 range, we chose it because we observed a lot of high scores and fewer lower scores. This way, we have fewer categories "in between", which might better capture the distribution of scores in our data.

2.2.1 Grades in a scale 0-5

The grades were sorted into 6 categories according to the Finnish university grading scale. The categorizing is based on the assumption that the maximum score of the students (7594) is the maximum amount of points a student can collect. Based on this, if a student got less than 50% of the maximum points, they would fail. If they got over 50% but under 60%, the grade would be 1 and so on. The class distribution is presented in Table 1 and Figure 6.

score_category	**Occurences**
0	4
1	4
2	10
3	5
4	17
5	62

Table 1: Table of the grades and their occurences

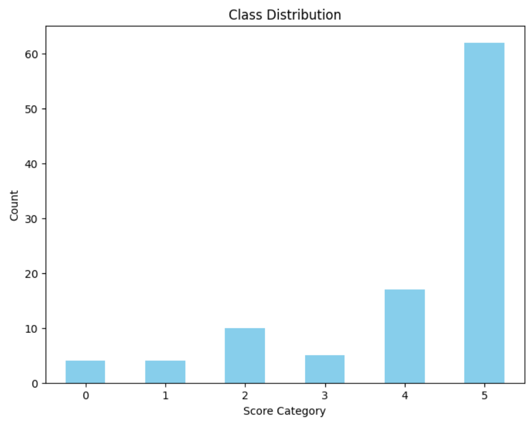


Figure 6: Class Distribution

2.2.2 Grades in a scale 0-3

Otherwise similar, but scores were divided into only four categories based on the percentages 90, 75, and 50. Still unbalanced, but less so than before.

score_category	Occurences
0	4
1	16
2	20
3	62

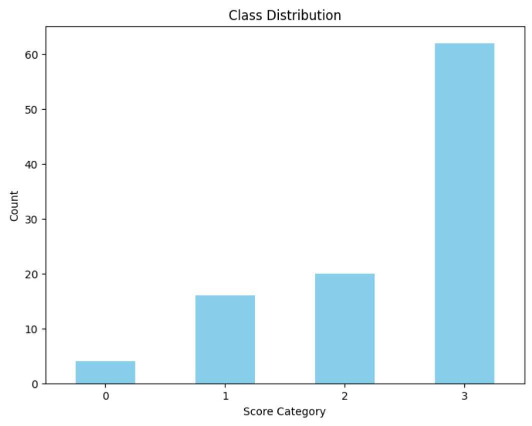


Figure 7: Class Distribution

2.3 Oversampling

We decided to apply oversampling[3] because the scarcity of data for lower scores (imbalance in the dataset), could lead to biased model predictions. Without a sufficient representation of lower scores, the model might not effectively learn the underlying patterns and relationships associated with these scores, resulting in reduced predictive accuracy and reliability. By oversampling the lower scores, we wanted to ensure that the model had an adequate amount of data to learn from across all score categories. This approach helped prevent the model from being skewed towards the more prevalent higher scores.

In this work, oversampling was crucial to ensure that the classification tree could effectively predict classes associated with lower scores. Without oversampling, the model might have been inclined to prioritize predicting the majority class (higher score) due to their prevalence in the dataset. This could have resulted in the classification tree being biased towards predicting higher scores while neglecting the lower score categories. By increasing the data samples of lower scores through oversampling, we provided the classification tree with a more balanced dataset, allowing it to learn equally about the classes associated with lower and higher scores. This approach prevented the model from focusing exclusively on the majority class (grouping the classes associated with lower scores into the class with score 0).

[3]Oversampling is a technique used to address class imbalance in a dataset by artificially increasing the number of samples in the minority classes. In our context, where there were very few samples from lower scores compared to higher ones, oversampling involved replicating instances of the lower scores to balance the dataset.

In the code, we used `SMOTE` (Synthetic Minority Over-sampling Technique, implemented in Python as part of the imbalanced-learn library `imblearn`) to balance the class distribution in the training data. It creates synthetic samples for the minority class to address class imbalance, ensuring better model performance. The resampled data is then used for training the decision tree classifier.

2.4 Standardization

In our analysis, we standardize our data before applying both the classification tree and regression models. This process involved scaling the features to have a mean of 0 and a standard deviation of 1. By standardizing the data, we ensured that all features were on a comparable scale, which is essential for the proper functioning of both models. This preprocessing step helps prevent features with larger scales from dominating the model's learning process and ensures that each feature contributes equally to the model's predictions. To do so, we used `StandardScaler` from `sklearn`.

3 Model: Decision tree

A decision tree is composed of a root node, branches and leaf nodes (Sharma et al. (2016)). Each non-leaf node tells to which sub tree the input sample should move until it reaches a leaf node that corresponds to some class (Sharma et al. (2016)). Basically, it's a simple contraption using if-else logic, where non-leaf nodes perform an evaluation on the data and send it down a branch depending on the evaluation outcome. In the ideal case the sorted data in the leaf nodes is homogeneous, in our case the data would have been sorted based on `score_category`.

To prevent overfitting the decision tree, stopping rules should be enforced. Some common ones are the minimum number of records in a leaf and the depth of the tree (Song and Ying (2015)), both of which we utilized in our model. Since our filtered dataset is on the smaller side, we want to keep the tree relatively shallow (3-5) and the minimum amount of samples in the node for splitting should be about 5-10% of the training data size. It should be noted, that in order to test our model's accuracy we divided the data into two sets, larger of which was used for training and the smaller for testing accuracy.

3.1 Evaluation metric

As we can see in Figure 6 and Figure 7, in both cases we have unbalanced classes. Therefore, accuracy (the fraction of correct predictions) is not the best metric for evaluation. The model could achieve high accuracy just by predicting the dominating class most of the time and be completely inaccurate about the less prevalent classes. We decided to use **F1-score**. F1-score is defined as the harmonic mean of precision and recall and ranges from 0 to 1, where 0 indicates poor performance and 1 indicates perfect performance. F1 scores closer to 1 are desirable, indicating a model with high precision and recall.

3.2 Hyperparameter tuning

We selected the parameters of the decision trees using hyperparameter tuning. This approach aims to optimize the performance of the decision tree classifier by systematically exploring different combinations of hyperparameters. We employed a grid search approach, the parameters tuned are `criterion` (the splitting criterion to be used in the tree), `max_leaf_nodes` (maximum number of leaf nodes), `min_samples_leaf` (minimum number of samples required to be a leaf node), and `max_depth` (maximum depth of the tree).

We performed a grid search with cross-validation using four to six folds to ensure robustness in the parameter selection process. To evaluate each parameter combination, we utilized the F1-score as our evaluation metric, with a micro-average to account for class imbalance. The best hyperparameters are the ones that give the highest F1-score.

We also experimented with fitting the classifier without oversampled data to assess its impact on performance. However, our results indicated that without oversampling, the model's performance notably degraded, underlining the importance of addressing class imbalance in our dataset.

3.3 Solutions obtained with the decision tree

We implemented the decision trees using the Python library `sklearn`. We split the data into training and test set, choosing to put 70% of our data in the training set and 30% in the test set. We did several different variations in order to try to improve the results. We used the Gini coefficient to measure the purity of the nodes. Values vary between 0 and 1.

Gini value 0 represents the perfect homogeneity, making it most desirable value.

3.3.1 Simple decision trees

The first decision tree we introduce was based on the filtered data with grade category 0-5 that had not been oversampled. The F1-score for this tree varied between ~ 0.61 and ~ 0.71, depending on the run. The maximum depth of the tree was kept small, only 3. We tested a few different options for the minimum amount of samples in a leaf node, in the Figure 8 it is set to 5. Since the data is so skewed and only a few students got scores in the lower ranges, it will be challenging to sort the data accurately with the lower scores. Since there is fewer data points in the lower categories than the minimum sample size of a leaf node, it is impossible to get good gini values for those. The algorithm seems to do a decent job separating at least the majority of the higher scores.

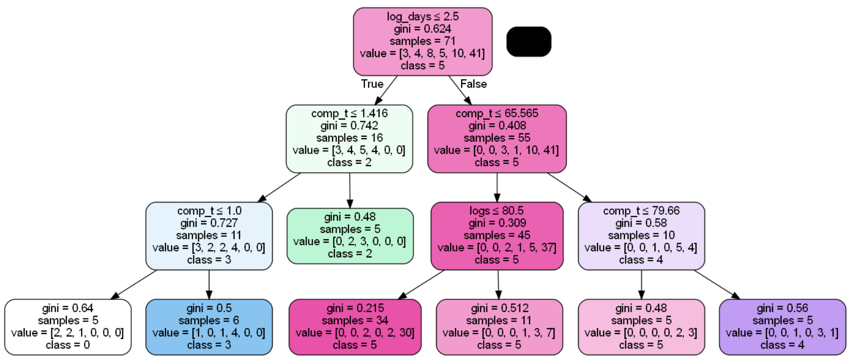


Figure 8: Decision tree with the filtered data, score scale 0-5

Below in Figure 9 we can see a decision tree with the only difference to the previous one being that this time we used the 0-3 classification for the score. Our F1-score is consistently ~ 0.74, which is (slightly) better than with the 0-5 classification. This does not necessarily mean that our estimations of the scores is better, since our categorization is looser.

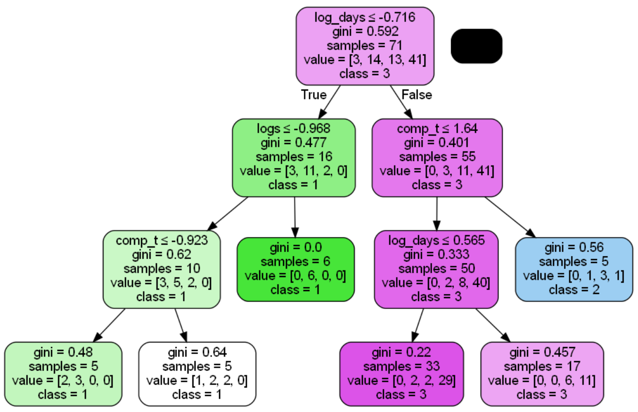


Figure 9: Decision tree with the filtered data, score scale 0-3

3.3.2 Decision tree with scores in 0-5 scale

From here onwards, we are working with the oversampled, standardized data. In order to maximize the model's accuracy, we used hyperparameter tuning. This gave us the following results:
`'criterion': 'gini', 'max_depth': 3, 'max_leaf_nodes': 5, 'min_samples_leaf': 1`

F1-score for the tree seen in Figure 10 alternates between ~ 0.70 and ~ 0.74, when set to the same random state as the previous decision tree. Results are better than with the simpler implementation of the tree. We can also see that the model seems to be able to divide lower and higher scores more "purely" than the previous tree.

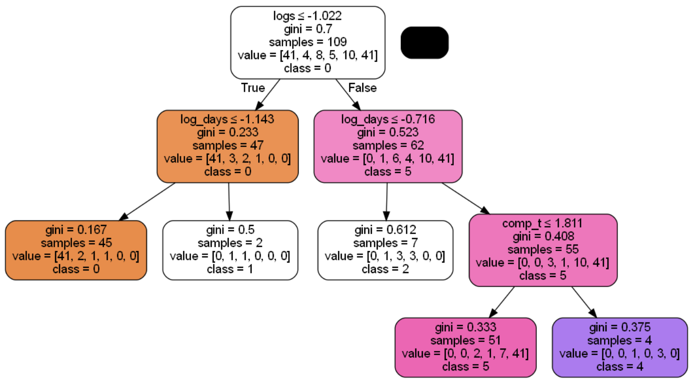


Figure 10: Decision tree with scores in 0-5 scale

3.3.3 Decision tree with scores in 0-3 scale

We had the same hyperparameter result values for this tree and it has the F1-score of ~ 0.74. We can see in the leaf nodes that each class can be found and their purities are rather good. It should be noted, that the F1-score is the same as in the "uncomplicated" version (even when using the exact same hyperparameter values that we used here). This would suggest, that our attempts at enhancing the quality of the data were unsuccessful with this version of the score categorization.

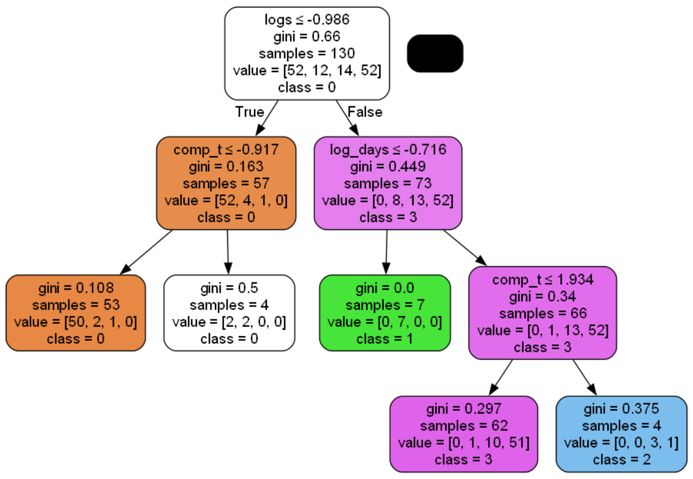


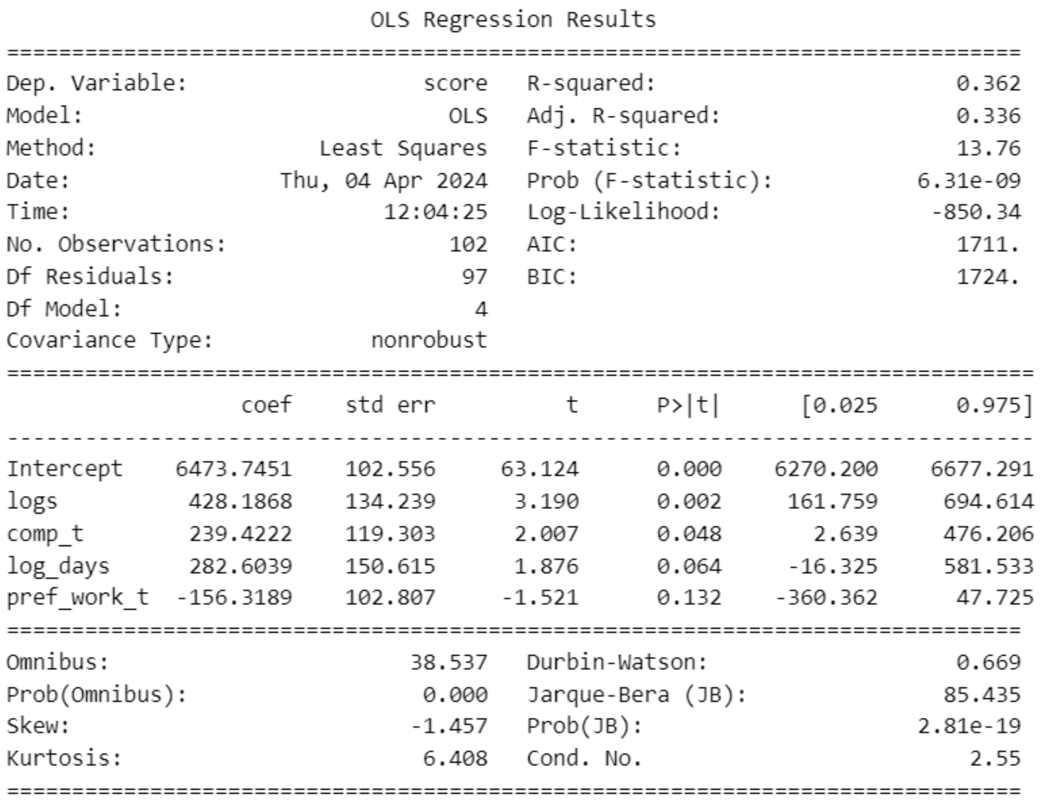
Figure 11: Decision tree with scores in 0-3 scale

4 Model: Linear Regression Model

Instead of categorizing the grades into three/five classes, we want now to predict the exact value of the score. To do so, we need a continuous model so we now try to fit a linear regression model in our dataset.
We use the filtered dataset, presented in Section 2.1.
The features that we select for the first fit are `logs`, `comp_t`, `log_days` and `pref_work_t`.
The model is then

$$\text{score}_i = \beta_0 + \beta_1 \text{logs}_i + \beta_2 \text{comp\_t}_i + \beta_3 \text{log\_days}_i + \beta_4 \text{pref\_work\_t}_i \qquad i = 1, 2, \ldots$$

The summary of this model is shown in Figure 12.



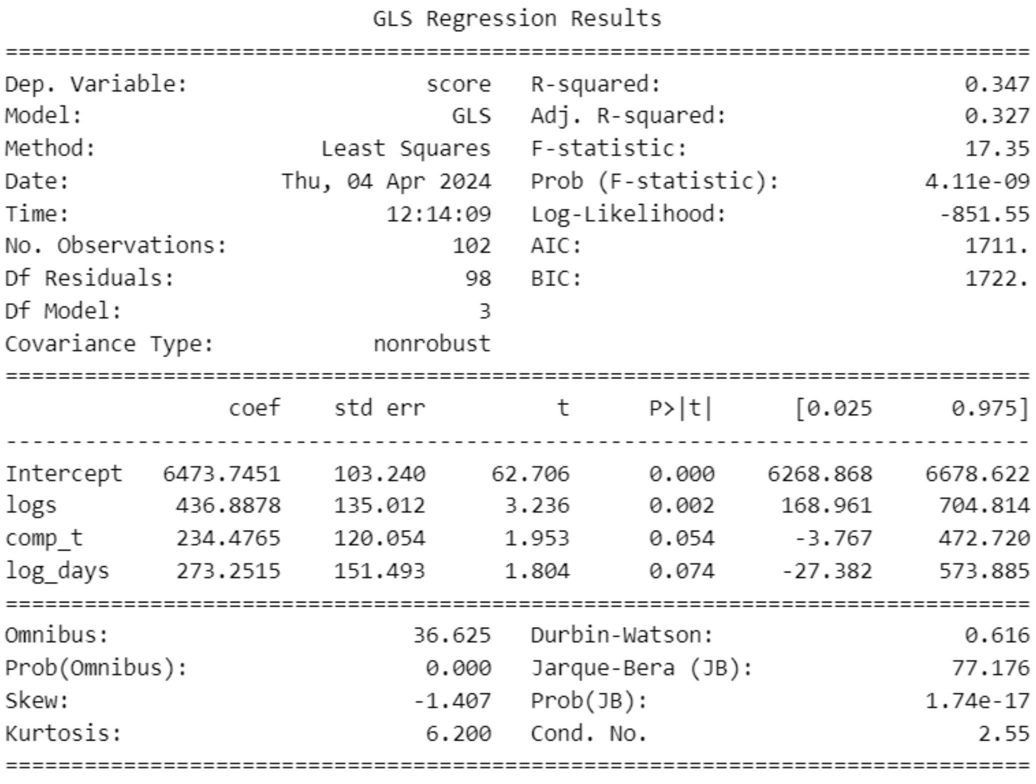
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.362
Model:                            OLS   Adj. R-squared:                  0.336
Method:                 Least Squares   F-statistic:                     13.76
Date:                Thu, 04 Apr 2024   Prob (F-statistic):           6.31e-09
Time:                        12:04:25   Log-Likelihood:                -850.34
No. Observations:                 102   AIC:                             1711.
Df Residuals:                      97   BIC:                             1724.
Df Model:                           4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept    6473.7451    102.556     63.124      0.000    6270.200    6677.291
logs          428.1868    134.239      3.190      0.002     161.759     694.614
comp_t        239.4222    119.303      2.007      0.048       2.639     476.206
log_days      282.6039    150.615      1.876      0.064     -16.325     581.533
pref_work_t  -156.3189    102.807     -1.521      0.132    -360.362      47.725
==============================================================================
Omnibus:                       38.537   Durbin-Watson:                   0.669
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               85.435
Skew:                          -1.457   Prob(JB):                     2.81e-19
Kurtosis:                       6.408   Cond. No.                         2.55
==============================================================================
```

Figure 12: Linear Regression Model (all the features)

The R-squared of this model is 0.362, it represents the proportion of the variance in the dependent variable that is explained by the independent variables in the model (goodness of fit of the model to the data). With this model we are explaining the 36.2% of the variance of our data, that is a quite good result. The remaining 63.8% of the variability is unexplained by the model and could be due to other factors not considered in the model or random variability.
We can select the most important features (i.e. the features that explain more variability) by looking at small values of the p-value in the summary of the model (`P>|t|` column). In this case, the selected features are `logs`, `comp_t` and `log_days`.
Fitting a new model with this new set of features we can see that the R-squared is almost the same (0.347, as shown in Figure 13) and now all the features selected are significant. In addition, we

have a simpler model, that can give faster results because it will require less computational time. The new model will still include the most important features for representing our data:

$$\text{score}_i = \beta_0 + \beta_1 \text{logs}_i + \beta_2 \text{comp\_t}_i + \beta_3 \text{log\_days}_i \qquad i = 1, 2, \ldots$$



```
                            GLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.347
Model:                            GLS   Adj. R-squared:                  0.327
Method:                 Least Squares   F-statistic:                     17.35
Date:                Thu, 04 Apr 2024   Prob (F-statistic):           4.11e-09
Time:                        12:14:09   Log-Likelihood:                -851.55
No. Observations:                 102   AIC:                             1711.
Df Residuals:                      98   BIC:                             1722.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   6473.7451    103.240     62.706      0.000    6268.868    6678.622
logs         436.8878    135.012      3.236      0.002     168.961     704.814
comp_t       234.4765    120.054      1.953      0.054      -3.767     472.720
log_days     273.2515    151.493      1.804      0.074     -27.382     573.885
==============================================================================
Omnibus:                       36.625   Durbin-Watson:                   0.616
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               77.176
Skew:                          -1.407   Prob(JB):                     1.74e-17
Kurtosis:                       6.200   Cond. No.                         2.55
==============================================================================
```

Figure 13: Linear Regression Model (final)

4.1 Examples of prediction

We then try to use the linear model

$$\text{score}_i = \beta_0 + \beta_1 \text{logs}_i + \beta_2 \text{comp\_t}_i + \beta_3 \text{log\_days}_i \qquad i = 1, 2, \ldots$$

to make some prediction, both on data points from the dataset and on data points arbitrarily decided. The results are shown in Tables 2 and 3.

logs	**comp_t**	**log_days**	**Predicted Score**
10	10	10	6377
89	10	3	6811
109	19	3	7196
9	100	5	6465
1	1	1	5115

Table 2: Predicted Scores for Arbitrary Data Points

score	**logs**	**comp_t**	**log_days**	**Predicted Score**
5601	34	1.050694	1	5638
6759	60	11.448611	5	6596
6862	44	59.961111	5	6714
7090	81	30.122917	7	7305
7346	45	68.870833	9	7264

Table 3: Predicted Scores for Data Points from the Dataset

Since we have very few instances of low scores in our dataset, the model might struggle to accurately predict these lower scores. However, it tends to perform better in forecasting higher scores, where we have more data available. This means that the model's predictions may be less reliable for low scores compared to high scores.

5 Summary

In this assignment, we used decision trees and regression to model the problem. In the beginning, we hypothesized that the student's score might be affected by the amount of active days, total log amount, the completion time and the preferred working time of the day. Based on the decision trees, it looks like only the three first ones had an effect on the score in the way one would expect: spend more time working on the problem (measured in time and the amount of activity), get better results.

Looking back, using random decision forests instead of decision trees might have been better. Random decision forests correct the tendency of decision trees to focus too much on the training data (overfitting), which could have given us more reliable results. Moreover, further enhancing our analysis with feature engineering techniques could have potentially improved the model's performance by identifying and leveraging relevant patterns and relationships within the data. For instance, with an R-squared value of 0.347 in the linear model, it's likely that we haven't captured all the pertinent features influencing our target variable, suggesting that there are additional factors contributing to the variability in the data that our current model hasn't accounted for. Exploring further feature selection and engineering methods could help uncover these overlooked variables and potentially enhance the model's explanatory power.

A Project Code Repository

For the source code and related materials used in this project, please visit our GitHub repository: `https://github.com/fskgiulia/educational-data-mining`. The repository contains the scripts, datasets, and any additional resources used in the project.

References

Aldowah, H., Al-Samarraie, H., and Fauzy, W. M. (2019). Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telematics and Informatics*, 37:13–49.

Behdad Bakhshinategh, Osmar R. Zaiane, S. E. . D. I. (2018). Educational data mining applications and tasks: A survey of the last 10 years. *Educ Inf Technol*, 23:537–553.

Bunkar, K., Singh, U. K., Pandya, B., and Bunkar, R. (2012). Data mining: Prediction for performance improvement of graduate students using classification. In *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, pages 1–5.

Sharma, H., Kumar, S., et al. (2016). A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5(4):2094–2097.

Song, Y.-Y. and Ying, L. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130.