Project Bachelor Year 1
Project 1.1

"Comprehensive Analysis of Student Performance Data Across Diverse Subjects for Informed Decision-Making"

Octavian-Costantin Rujan, Ariannys Cermeño, Jakub Bujak, Jesse Hoydonckx, Kordian Marcin Klimas, Jakub Tomasz Polichnowski

1. Abstract

This project is driven by the necessity to extract meaningful insights from data of an academic institution which are: the grades across various subjects of graduated students, of current students, and their different attributes. The challenge lies in finding ways to extract information from the data provided in order to comprehend student performance and improve the academic institution's organisation.

Our approach involves the development of a Java program that examines missing data, predicts values, and identifies patterns within the existing datasets through statistical and Machine Learning tools. The principal outcome is the development of a user-friendly Java program that discerns course difficulty levels, predicts missing grades, and foresees future academic performance by visually displaying data using graphs, enabling a quick and efficient grasp of school and students' performance.

The results provide information that can be used in a plethora of ways like, for example: giving extra preparation to students that the algorithm predicts will perform poorly in a course, rearranging course order based on the difficulty of the courses, etc.

In conclusion, this study holds the development of a robust program capable of extracting actionable insights from student data allowing educators and administrators to make informed decisions, enhance educational support, and identify areas that require intervention, leading to better academic performance.

2. Introduction

For many years, education systems have been methodically gathering data for reporting and ongoing development (Cannata et al., 2016). This data allows Educational Data-Driven Decision Making, which plays an important role in the improvement of teaching systems. It does so by offering insights into students' activities and behaviors allowing educators to make informed decisions to provide personalized learning experiences (Mougiakou et al., 2023). Currently, these insights are obtained through Exploratory Data Analyses (EDA) which collects, analyses, and examines data effectively for interpretation (Courtney, 2021). In addition, Data mining tools are currently being developed to offer more information on hidden patterns in the data.

However, the current state of research reveals that existing systems fall short of addressing missing data and extracting key insights (McFarland et al., 2021). This is a challenge that we have to overcome when it comes to the missing data that comes as NG. These limitations are the starting point for this study: the need for improved techniques to handle missing data and enhance the assessment of educational parameters. Hence, this study is focused on answering the research question: How can we derive comprehensive insights from the provided student data to improve our understanding of the institution?

Based on our primary research query, we have delineated specific sub-questions for targeted exploration designed to delve into the core of fundamental data interpretation for educational decision-making and have put them into two different clusters:

1. **Statistical Methods:** These methods aim to answer sub-questions such as: How can we effectively analyze the difficulty of courses? How is it possible to identify the top-performing students? How can we analyze existing correlations between courses? What valuable information lies within missing grades? Is there any correlation between student's properties and their grades?

2. **Predictive Methods:** This cluster aims to answer the sub-question: How can we predict students' performance? and to also answer the very important problem that comes along with this question: How accurate are the predictions made by the algorithm?

To tackle these sub-questions, we designed our approach around two clusters of questions: the first cluster contains algorithms of statistical measures such as mean, standard deviation, and Pearson correlation and implements a grouping algorithm that separates the courses in different years; the second cluster employs different types of prediction algorithms and tests their efficiency. This structured approach allows us to systematically delve into each aspect, providing a comprehensive understanding of student performance data.

These algorithms are the backbone for the Java program developed to communicate the information obtained and allow educators to gain effective and quick insights through Visual Analytics (Vieira et al., 2018). The following report is structured with sections on the methods used, their practical implementations, details about the experiments, and their outcomes. It also includes a discussion of the results and concludes by summarising the main findings and proposing potential directions for future research.

3. Methods

Our methodology consists of the development of algorithms that employ statistical measures along with the development of a grouping algorithm that assures that there is a distinction between the order of the courses taken and the development of different prediction algorithms to choose from with a varied range of complexion and accuracy. The choice of datasets is a crucial aspect of our methodology. We utilized datasets encompassing a range of subjects, student grades, and attributes of the students (Suruna Value, Hurni Level, Lal Count, and Volta). These datasets allow efficient coverage and representation of the educational landscape that we have been provided with and that must be explored.

3.1. Statistical Measures Algorithms

3.1.1. File-to-Array Algorithm

To deal with the data sets given we have decided to use a file-to-array method that transcribes the data sets we have gotten into arrays using a scanner object. We have decided to deal with the data sets in this manner for two reasons which are the ease of use and the ease of understanding that this approach provides. To deal with the missing values in the Current Graduate data set (NG) this algorithm transforms all of the NG values into -1 values to easily distinguish between them and have an array of doubles.

3.1.2. Statistical Measures Algorithms

We developed algorithms capable of obtaining the main statistical measures by applying their mathematical formulas (see Appendix A) to the arrays created. The specific measures taken are the mean of the grades of a student or course, offering insight into the average performance of students being able to identify top performers or the difficulty of the course; the standard deviation of the grades, highlighting the variability within each course; and Pearson correlation, measuring the similarity between the courses. These statistical tools enable the comparison between datasets, enabling a deeper understanding of the data provided.

3.1.3. Course Distribution Algorithm

To analyze the way students are structured we developed a grouping algorithm that groups students based on whether they are taking the same set of courses.This method works on the CurrentGrades data set and utilizes the missing information that was priorly transformed into -1 in the CurrentGrades array by the File to Array method. The array is then checked in each course for the -1 value and groups the courses based on the number of NG values that the course has. This intends to show the order the courses are being taken (year 1, year 2 year 3). . Using visualising tools this algorithm shows a clear distinction between the years the courses are taking place.

3.1.4. Property Grouping algorithm

The property grouping algorithm works by taking the average of all the people with a specific attribute in a specific course, this groups up the properties along with the level those properties are at giving them a tangible number that can be displayed. The algorithm starts by looping over each person with the attribute searched and using their id to get their grade for a specific course, afterwards, it sums them up and takes the average of the grades all students with that attribute have as output.

3.2. Prediction Algorithms

3.2.1. First Iteration of the Prediction Algorithm

To predict grades we based our prediction on the performance of other courses as well as the properties of students. The algorithm is based on the data sets of CurrentGrades and StudentInfo. Its purpose is to identify the best attribute for a prediction using a decision stump utilizing variance reduction to check which attribute is the most relevant to that specific course (variance being inversely proportional to correlation). After finding the best property the algorithm checks for the mean of those with the same property and the same value of that property for the target course excluding the ones with the NG value, the algorithm provides that mean as a simple prediction of the student's grade for the course.

3.2.2. Predicting with a Binary Regression Tree

To enhance the predicting algorithm we employed the machine learning technique of decision trees. The Node class represents decision tree nodes, distinguishing between decision and leaf nodes. The BinaryDecisionTree class constructs the decision tree with parameters for minimum split and maximum depth and builds the decision tree by recursively splitting the dataset based on feature values and selecting the best split that maximizes variance reduction.

1. Create a class for nodes of the decision tree with the parameters attributeIndex, threshold, Node left, Node right, variance reduction, and value.

2. Create the class to create Binary Decision Trees with the parameters minimum split and maximum depth.

3. Create a Method to Build the Decision Tree with an array dataset as the parameter.

    a. Extract the attributes (X) and course values (Y) from the data
    b. While stopping conditions are not met
        i. Check if the best split is valid
        ii. Build recursively the left and right subtrees with the split datasets.
        iii. Returns a decision node with the current split data
    c. Returns leaf node with a prediction value

4. Method to split the dataset given an array a feature Index and a Threshold

    a. Initialize arrays for left and right dataset
    b. Check the condition for splitting, the threshold
        i. If true, add to the left dataset; if not, add to the right dataset
    c. Returns the split datasets

5. Method to find the best split based on variance reduction

    a. Initialise a String[][][] array to assign the BestSplit
    b. Loop over all features and get the unique values of the current feature
    c. Loop over all unique feature values and get the current split
    d. If children are not null
        i. Extract labels (y, left_y, right_y)
        ii. Compute variance reduction

iii. If variance reduction is higher, assign the current Split to the BestSplit
e. Return Bestplit

6. Method to predict a student's grade with the parameter of an array of strings of the student attributes
    a. Extract the attribute values for the current node from the Student attributes
    b. If the current Node is a leaf node it returns the value of the node
    c. Else check if the feature value equals the node's threshold
        i. If it does, recur into the left node subtree with the makePrediction method
        ii. Otherwise, recur into the right node subtree with makePrediction

7. Method to train the data set with parameters of the array of attributes, course grades, and course index.

    a. Create a new array concatenating features and the course we want to build the tree with, adding as the last column of the array of attributes the grades for the course we want to predict the student's grade for.

    b. Use the buildTree with the previous array

3.2.3. Random Decision Forest

To create a more accurate prediction we implemented a decision forest algorithm. To do so, we implemented a new type of decision tree that has attribute sampling, meaning each node randomly chooses a random set of attributes to evaluate and bootstrapping, this way each decision tree of the forest will be trained on a slightly different dataset increasing the accuracy of the final forest as it is trained with random noise. Based on the input of the user, the algorithm generates as many different trees as desired, and at the end, it takes the average predicted value out of all the trees that have been created in the forest and offers that as a prediction for the specific student grade. The pseudocode for the decision tree is located underneath this text.

1. Create a Class for ForestTrees that Inherits the Binary Decision Tree
    a. Constructors for the Forest trees is the number of random attributes
    b. Override method to choose the best split
        i. Instead of going over all attributes, select attributes at random without duplicates
        ii. loops over random attributes
    c. Rest of the Original Code

2. Create a Class for RandomForest using as parameters the forest size, number of variables selected in each tree, splits of each tree, depth of each tree, size of the training set for each tree, and array that stores all the trees.

3. Create a Method to fit the forest with parameters of an array with attributes, an array with grades, and the index of the specific course in the array of grades.
    a. Generate a tree with specified settings.
    b. Generate a bootstrapped set to train the tree on.
    c. Fit tree onto data.
    d. Add a tree to an array that stores the trees.
    e. Repeat until the array that stores all trees is full.
4. Create a Method to predict the RandomForest with the properties of a student.
    a. Loop over all trees and Store the predicted grade for each tree.
    b. Take the average of all predictions.
    c. Returns the prediction as a double.

3.2.4. Prediction Algorithm Accuracy Test

To determine the accuracy of the algorithm we use a measure called Mean Average Error. This is the average difference between the grade and the prediction. To do so first we divided the data sets provided into training sets and validation sets.This way we can compare and contrast how well the algorithm predicts the validation data set. As the tree and the forest are both fitted to one course we use all students in the validation dataset that have a grade for that course. Below is the pseudo-code for this test.

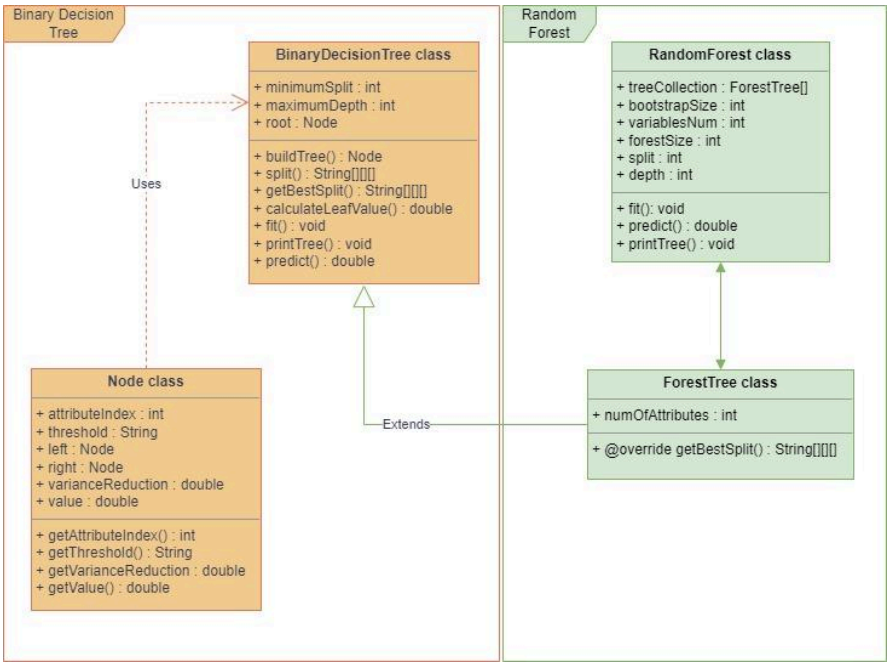
1. Create a DataPreparation class to obtain validation datasets and training datasets
    a. Use as parameters the entirety of the data set with grades and dataset of student properties
    b. Overload to work for both strings and doubles
    c. Create a getTrainingData method with course index as a parameter.
        i. Chooses student IDs with grades for a given course
        ii. Returns the first 85% of student IDs
    d. Create a getValidationData with course index as parameter
        i. Chooses student IDs with grades for a given course
        ii. Returns the last 15% of student IDs

2. Create a Method to measure the accuracy with a binary tree and course index as a parameter
    a. Gets validation data from the DataPreparation class
    b. Check if there is validation data (some courses have no grades)
    c. Loop aver all students in validation data and predicts their grades
    d. Takes an average of the error of all students
    e. Returns the Mean Average Error as a double.

4. Implementation

All of the statistical and grouping methods are situated in a helper class named Methods. Other classes interact with this class by calling the method they need. This is the only interaction between classes outside of the prediction algorithms.

The UML figure below portrays the code implementation of the random forest. The decision tree is included in the UML diagram, as it is partly used in the random forest.



The GUI of the application was made using the external libraries of JavaFX. Its development was motivated by the need for a visualizing tool to better understand the data we have been given and for the ease of use it brings to a user. The main frame presents a menu with multiple drop-down menus where a user can choose between different types of data to view in different types of graphs.

5. Experiments

The primary objective of the experiments was to check whether the outcomes of our program matched those generated by traditional tools such as Excel. The experiments were structured to address each specific research question, allowing for a comprehensive examination of our algorithms' performance.

The first experiment involved the calculation of the mean using our algorithm and then manually inputting the data into Excel for comparison. The settings included using a diverse dataset encompassing grades across various subjects. The results obtained from both approaches were cross-verified to ensure accuracy and consistency.

Subsequent experiments followed a similar pattern, focusing on different aspects of our methodology. This included calculating standard deviation and Pearson correlation using our algorithms and validating the results against Excel-generated outcomes. The datasets used were tailored to represent diverse scenarios, covering a range of subjects and student performance profiles.

Secondly, the objective was to optimize the decision algorithm. This can be done by tweaking the parameters such as tree depth. By finding the best possible value for the parameters we hope to achieve a better and more accurate prediction. We implemented a method to test different parameters and return the accuracy of the prediction algorithm with a certain parameter. This is done for both the decision tree and the random forest.

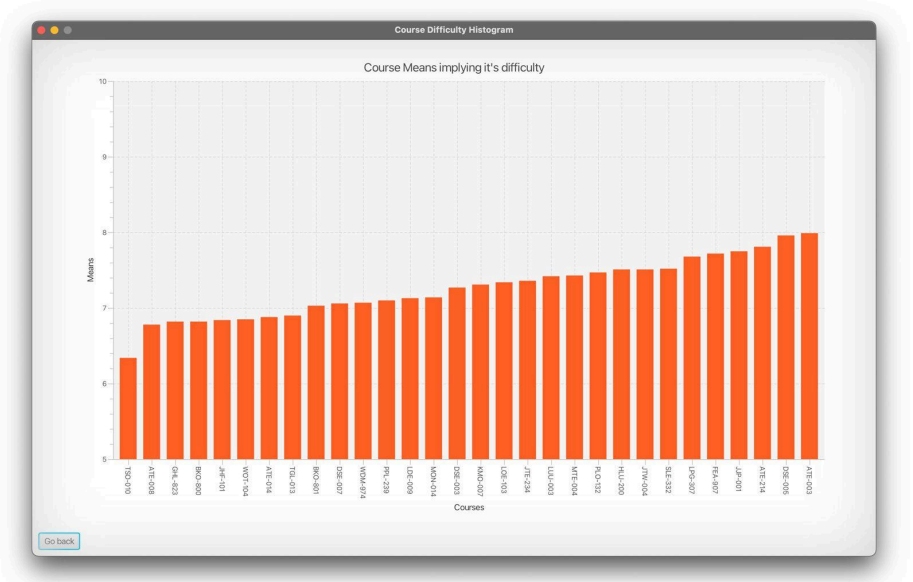
In a decision tree, we will test the depth. A random forest has more parameters we can test, namely the depth for each tree, the size of the forest, the size of the bootstrapped training sets, and the amount of randomly selected variables in a node. All of the tests can be executed similarly. The pseudo-code below represents a general approach that we can use for testing every parameter.

1. Testing Parameters: Inputs bounds (minimum, maximum)
2. Loop Over All Parameters
    a. Create a Forest or Tree
    b. Gets Accuracy of the Prediction for All Courses
    c. Print Results: Current parameter value, Accuracy value
3. End Loop

6. Results

6.1. Algorithms of Statistical Measures:

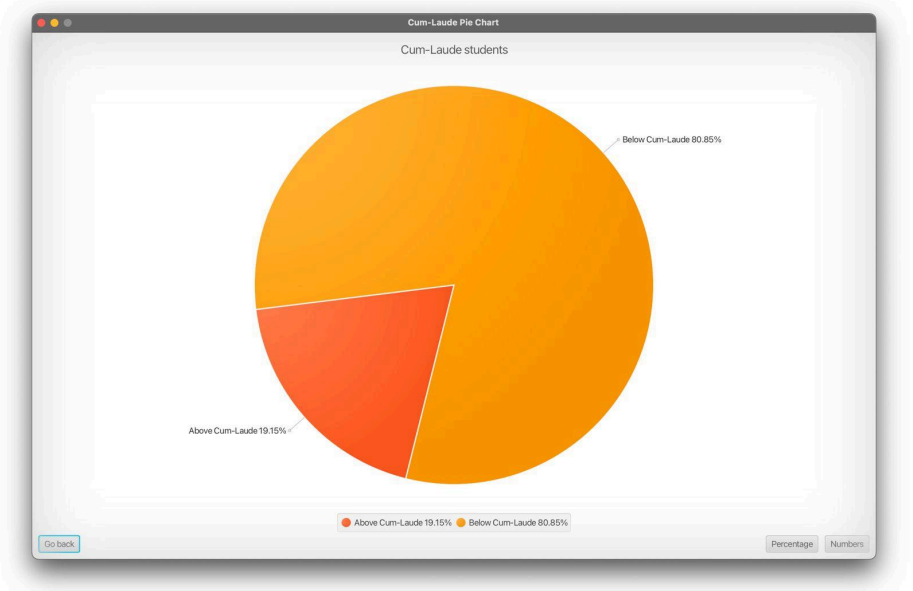
The calculation of statistical measures, including mean, standard deviation, and Pearson correlation, has provided valuable insights into overall performance trends. The mean, a central tendency measure, offered a glimpse into the average performance across subjects. Circling back to the research question: Can we analyze the difficulty of the courses? We concluded which course was the hardest and which was the easiest through their average score.. Figure 1 illustrates the distribution of mean scores across different courses, allowing for a visual assessment of course difficulty.



[Figure 1: Bar graph of Mean Scores Across Courses]

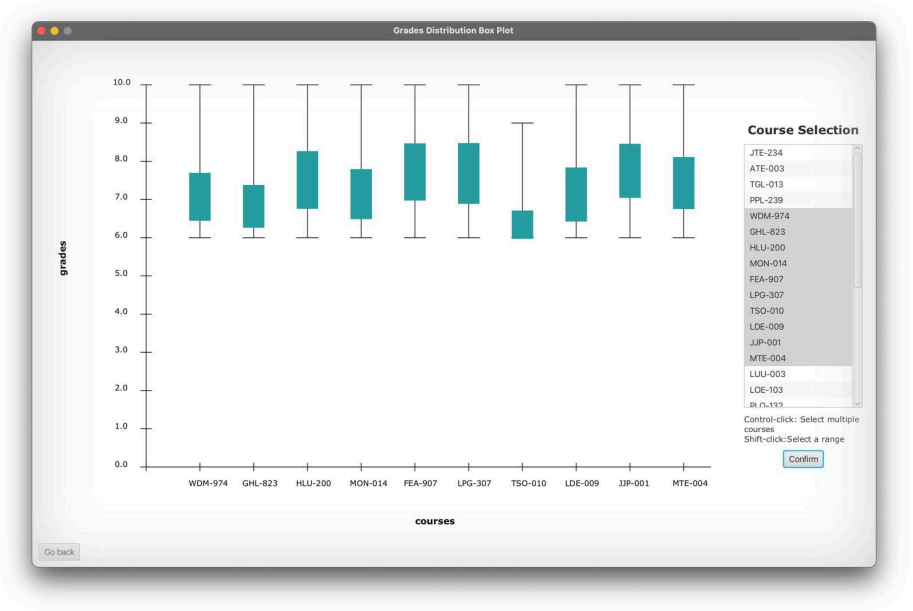
Using the visualization tool provided we can conclude that the hardest course we have analyzed is TSO-010 and the easiest one happens to be ATE-003.

Moreover, we were also able to address the question: Can we identify the best students? The mean of every graduated student allowed us to discern those who graduated cum-laude. Figure 2 shows the percentage of students who graduated cum-laude.



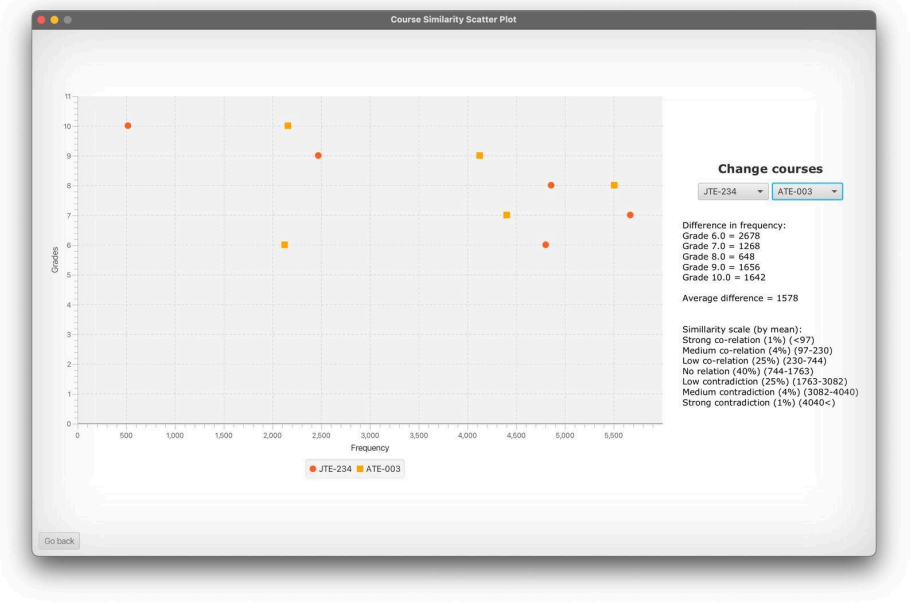
[Figure 2: Pie Chart representing Cum-Laude Students]

Simultaneously, the standard deviation highlighted the dispersion of grades within each course, portraying the level of variability in student performance. Figure 3 provides a graphical representation of standard deviation values for different courses, aiding in the identification of courses with diverse performance ranges.



[Figure 3: Box Plot of Grades Distribution]

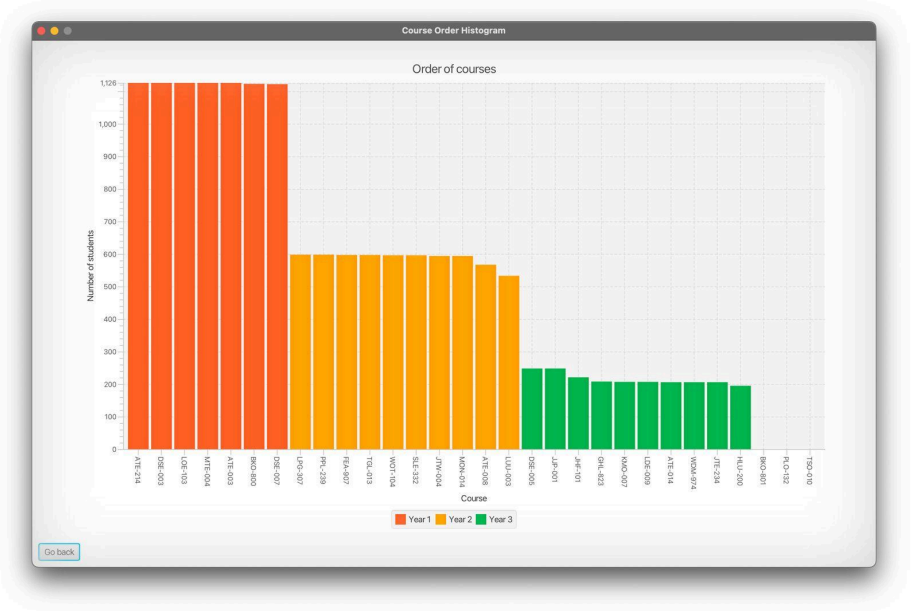
The Pearson correlation facilitated the examination of relationships between different subjects, aiding in the identification of variations and similarities. Figure 4 visually represents the correlation matrix, demonstrating the interconnectedness of subjects and informing our understanding of course relationships. Hence tackling the question, Are there similarities between courses?



[Figure 4: Scatter Plot of similar courses]

Using the Pearson correlation formula we can see that some courses are heavily related to one another and also the opposite. The GUI that presents this information also brings to the forefront the meaning of the values by the relation they have.

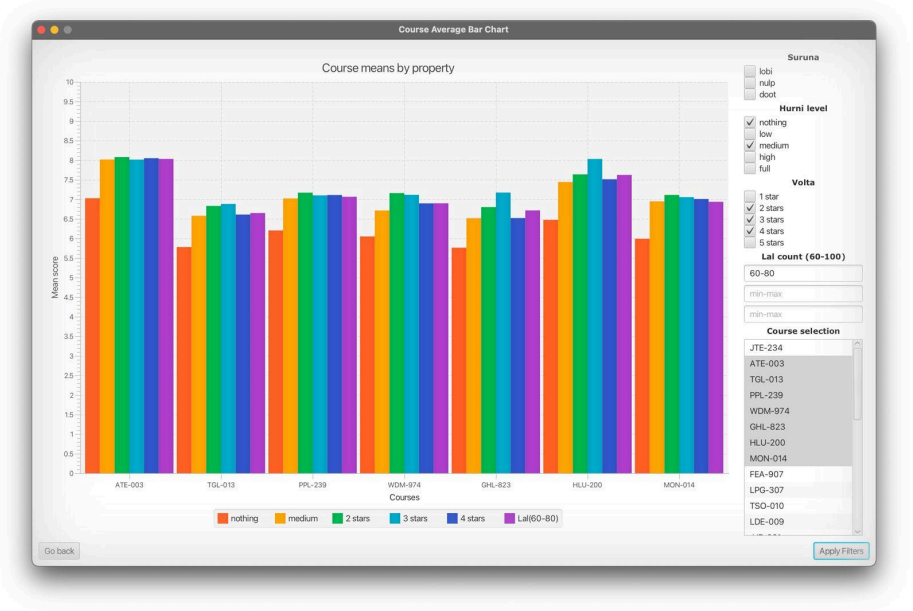
Regarding the question: Is there information in the missing grades? We were able to obtain information that allowed us to group students according to the number of NG values in a course uncovering noteworthy patterns. Figure 5 illustrates the disparities in missing scores.



[Figure 5: Graphic Representation of the order of courses]

After visualizing the information we have we can easily conclude that the missing grades equal courses that have not been taken yet by those respective students. We can also find out the year those courses may belong to and how far away are students from taking those classes.

Our implemented property grouping algorithm, designed to group students based on the property average grades they have received on courses, yielded meaningful clusters within the dataset. These clusters answer the research question: Is there any correlation between student's properties and their grades? Figure 6 showcases the resulting clusters, providing a visual representation of how students with similar academic properties were grouped.



[Figure 6: Bar Graph of courses means by property]

In Figure 6 we can see there are noticeable differences in the property average grades that exist for each discipline in particular, this shows that there is a big correlation between a student's properties and their grade in a specific course.

6.2. Predicting Algorithm

The question "Can we predict the students' grades?" is a focal point of our study, and our implemented predicting algorithm played a crucial role in providing substantial insights. We optimized our algorithm using the test explained in the experiments section. The following figures represent the results of the different experiments and why we choose which variables.

6.2.1. Experimenting with different tree depth

Figure 7 shows the result of testing the depth between 0 and 6 for a decision tree. Through this experiment, we determined that 3 is the optimal depth.

```
TREE DEPTH TESTING
depht: 0 -- accuracy: 0.7643927076656892
depht: 1 -- accuracy: 0.6918583919580296
depht: 2 -- accuracy: 0.6397836233492427
depht: 3 -- accuracy: 0.611562698379958
depht: 4 -- accuracy: 0.6310954123248742
depht: 5 -- accuracy: 0.6650898232766799
```

[Figure 7: Results of the accuracy of trees with different depths]

For the testing of the random forest, we used the same settings for all of them except for the parameter we are testing. (forest depth = 3, splits for every tree = 2, size of training data (bootstrap size) = 25, forest size = 50, number of variables = 3). The results for testing the depth of each tree in the forest show similar results. Here the best depth is also 3, as depicted in Figure 8.

```
FOREST DEPTH TESTING
depht: 1 -- accuracy: 0.6624176454310068
depht: 2 -- accuracy: 0.6199750748711955
depht: 3 -- accuracy: 0.591834745511378
depht: 4 -- accuracy: 0.5954969981487844
depht: 5 -- accuracy: 0.613579306275516
depht: 6 -- accuracy: 0.6316411637589887
```

[Figure 8: Results of the accuracy of forests with different depths]

6.2.2. Experimenting with different forest sizes

The size of the forest mostly has an impact on performance. As we add more trees, the forest will take longer to predict. The size doesn't have an impact on the accuracy. A random forest will not overfit as we add more trees. The size we chose is 50. After this, we see diminishing returns.

```
FOREST SIZE TESTING
size: 0 -- accuracy: NaN
size: 10 -- accuracy: 0.5959687855627508
size: 20 -- accuracy: 0.5977283207362167
size: 30 -- accuracy: 0.5942868255474957
size: 40 -- accuracy: 0.5921901090151988
size: 50 -- accuracy: 0.5930870185676718
size: 60 -- accuracy: 0.5929506414430784
size: 70 -- accuracy: 0.5908509234010015
size: 80 -- accuracy: 0.5896695535650317
size: 90 -- accuracy: 0.591559523021654
size: 100 -- accuracy: 0.5905664345294402
```

[Figure 9: Results of the accuracy of forests with different size testing]

6.2.3. Experimenting with the number of variables (attributes)

The best value for the number of variables according to its accuracy is 3, as indicated by Figure 10 down below.

```
FOREST NUMBER OF VARIABLES TESTING
#variables: 1 -- accuracy: 0.6871435911090105
#variables: 2 -- accuracy: 0.6135426303238924
#variables: 3 -- accuracy: 0.5932685029495025
#variables: 4 -- accuracy: 0.6115626983799577
```

[Figure 10: Results of the accuracy of forest with different variables]

6.2.4. Experimenting with the size of the training set

The last parameter we tested is the size of the training set for the trees in the random forest. Here, we determined the optimal value to be between 20 and 30 as represented in Figure 11. Hence, we decided to use a size of 25.

```
BOOTSTRAP SIZE TESTING
size: 10 -- accuracy: 0.5920437974733779
size: 20 -- accuracy: 0.5915116048054091
size: 30 -- accuracy: 0.5910071476239593
size: 40 -- accuracy: 0.5927582825988333
size: 50 -- accuracy: 0.5917886533257734
```

[Figure 11: Results of the accuracy of forests with different bootstrap sizes]

6.2.5. Predicted Grades

Finally, after obtaining the most optimal set of parameters for the Random Forest (Depth: 3, Splits for every tree: 2, Bootstrap size: 25, Forest size: 50, Number of variables: 3) we obtained a predicting algorithm. Figure 12 is an example of the predicted grades for 10 Student IDs for all the courses.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1000039	6.15	7.11	5.82	6.37	6.39	5.84	6.86	6.48	7.3	6.7	-	6.62	6.47	6.77	6.73	6.74	-	5.83	6.61	-	6.51	7.25	5.6	6.35	6.17	5.74	6.71	5.07	6.8	6.02
1000140	7.09	8.06	6.21	6.55	6.39	5.78	6.82	6.82	7.33	7.72	-	7.17	6.97	6.91	6.98	7.47	-	6.42	7.45	-	6.94	7.44	6.42	6.9	6.61	7.31	7.59	5.89	7.63	6.7
1001349	7.06	7.94	5.56	6.56	6.18	5.88	6.98	6.8	7.39	7.54	-	7.14	6.97	6.94	6.92	7.68	-	6.37	7.38	-	6.94	7.73	6.42	6.9	6.61	7.34	7.56	5.86	7.63	6.62
1001975	6.3	6.87	5.71	6.25	6.48	5.86	7.03	5.56	6.54	5.63	-	6.03	6.42	6.85	6.27	5.7	-	5.29	5.56	-	6.13	7.41	5.6	6.52	5.62	4.95	6.7	4.88	6.82	5.32
1002038	5.81	7.59	6.21	6.38	6.5	6.11	6.97	6.33	6.65	6.26	-	6.03	6.48	6.87	6.4	6.21	-	6.2	6.52	-	6.08	7.42	5.61	7.06	6.1	6.21	6.57	5.42	6.82	6.1
1002210	6.98	7.87	7.11	7.76	7.68	7.55	7.01	7.5	8.15	7.54	-	7.35	8.11	8.22	8.09	7.34	-	6.92	7.53	-	7.11	8.51	6.66	8.27	7.04	7.01	7.98	6.32	9.41	6.61
1002222	5.76	7.59	6.21	6.38	6.5	6.17	7.08	6.42	6.69	6.55	-	6.03	6.48	6.87	6.6	6.21	-	6.2	6.52	-	6.04	7.4	5.61	7.05	6.32	6.18	6.57	5.48	6.53	5.94
1002253	8.36	9.26	7.41	8.09	8.48	7.09	8.95	7.84	9.15	8.9	-	8.01	8.74	8.99	8.25	9.25	-	8.04	8.81	-	8.43	9.0	8.7	8.49	8.06	8.62	9.09	8.68	9.8	7.76
1002292	6.15	7.1	5.86	6.38	6.27	5.9	6.86	6.4	7.32	6.7	-	6.58	6.52	6.77	6.7	6.74	-	5.83	6.54	-	6.53	6.99	5.36	6.49	6.17	5.74	6.71	4.86	6.8	6.02
1002319	6.13	7.59	6.22	6.74	6.5	6.51	5.72	6.47	7.17	6.62	-	6.04	7.8	7.37	6.83	6.25	-	6.23	7.16	-	6.46	8.07	6.05	7.22	6.1	6.21	7.55	6.14	7.12	6.3

[Figure 12: Table of the predicted grades for all the courses for 10 students]

7. Discussion

The results from our study provide valuable insights into key aspects of student performance, addressing the research questions. Utilizing statistical measures such as mean, standard deviation, and Pearson correlation, we gained a comprehensive overview of overall performance trends. Consistent validation of our program's results against traditional tools like Excel underscored the accuracy and reliability of our methods.

However, an issue arose concerning the influence of outliers on predictive models. While our variance reduction strategies proved effective, refining model accuracy could be achieved by addressing outliers

through robust statistical techniques. At the same time, the limited amount of data available for some courses causes the accuracy of our predictive algorithm to diminish. For future studies, considering additional outlier detection mechanisms would further enhance predictive capabilities.

8. Conclusion

In conclusion and answering our research question: How can we derive comprehensive insights from student grade data across diverse subjects to enhance our understanding within the educational domain? Our research dives into the complexities of student performance, utilizing statistical measures to assess trends and developing predictive algorithms for future academic performance. Therefore identified patterns are visually transmitted offering actionable insights for educational strategies.

During this study, we also found ways to answer the sub-questions: How can we effectively analyze the difficulty of courses? ; As explained in the methods section, taking the average grade of each course and putting them all into a graph shows the hardest and easiest courses. How is it possible to identify the top-performing students?: We can identify the top-scoring students by analyzing which students graduated cum-laude. How can we analyze existing correlations between courses?: Using the Pearson Correlation Coefficient we can find out if two courses are correlated. What valuable information lies within missing grades?: In the missing grades we can find the order in which the courses are being taken. Is there any correlation between student's properties and their grades?: There is a big relation between a student property and there as we have found out using the visualization tool showing the average of each property for a specific course and comparing it to the normal average of the course. How can we predict students' performance? To predict students' performance we use a random forest algorithm to come as close to reality as possible. How accurate are the predictions made by the algorithm? The prediction accuracy test looks at how far away is the prediction from the actual grade they were supposed to have.

Looking ahead, we could use advanced predictive modeling techniques like neural networks to elevate the precision of grade predictions, surpassing the limitations of linear regression. The introduction of these innovative approaches holds the potential to refine existing methodologies in addition to those implemented during this study.

In summary, our study provides a comprehensive understanding of student performance, utilizing statistical measures and predictive algorithms. The identified patterns serve as valuable indicators guiding decisions in educational strategies. Continuous exploration and improvement are essential, as acknowledged limitations, particularly in the scope of considered variables, highlight the need for ongoing improvement.

9. References

Cannata, M., Redding, C., & Rubin, M. (2016). In *Continuous Improvement in action: Educators' evidence use for school improvement*. Washington DC; Distributed by ERIC Clearinghouse.

Courtney, M. B. (2021). Exploratory Data Analysis in schools: A logic model to guide implementation. *International Journal of Education Policy and Leadership*, *17*(4). https://doi.org/10.22230/ijepl.2021v17n4a1041

Educational Data Mining - Javatpoint. www.javatpoint.com. (n.d.). https://www.javatpoint.com/educational-data-mining

McFarland, D. A., Khanna, S., Domingue, B. W., & Pardos, Z. A. (2021). Education data science: Past, present, future. *AERA Open*, *7*, 233285842110520. https://doi.org/10.1177/23328584211052055

Mougiakou, S., Vinatsella, D., Sampson, D. G., Papamitsiou, Z., Giannakos, M., & Ifenthaler, D. (2023). *Educational data analytics for teachers and school leaders*. Springer.

Vieira, C., Parsons, P., & Byrd, V. (2018). Visual learning analytics of educational data: A Systematic Literature Review and Research Agenda. *Computers & Education*, *122*, 119–135. https://doi.org/10.1016/j.compedu.2018.03.018

10. Appendix

Formulas for statistical measure algorithms;
Pearson correlation :$r = \Sigma((Xi - \bar{X})(Yi - \bar{Y})) / \sqrt{(\Sigma(Xi - \bar{X})^2 \Sigma(Yi - \bar{Y})^2)}$
Mean formula : $\bar{x} = (\Sigma xi) / n$
Standard deviation: $\sigma = \sqrt{(\sum x - \bar{x})2 /n)}$