Mini Project 2 Report

Maša Ćirković

I. Introduction

With many courses taking place online nowadays, it is important to be able to recognize and motivate or offer help to students who are at risk of under-preforming, making online learning a better and more insightful experience for them.

This report focuses on addressing the challenge of predicting students' final grade in an online machine learning course. The course, hosted on Moodle, spanned nine weeks, with various activities and assessments contributing to the final grade. The prediction task is framed as a multi-class classification problem, where the goal is to forecast the final grade as an integer from 0 to 5 based on features representing students' engagement, performance, and activity over the course.

Two approaches are being explored: using regression models with post-processing (rounding predictions to the nearest grade) and directly applying classification models. The motivation is to assess which method produces better predictive performance and more meaningful insights into students' learning behaviors.

This analysis can potentially help educators and course administrators identify students at risk of under-performing and offer timely interventions, ultimately improving the learning experience in online education platforms.

II. Data Processing

Data Processing involves understanding what the data is about, exploring different data properties and transforming it into a suitable representation for further analysis or training the models.

A. Data Explanation

The dataset contained anonymous information relating to 107 enrolled students.

The data included students' grades (from 3 mini projects, 3 quizzes and 3 peer reviews and the final overall grade) as well as the course logs.

- **Status0**: course / lectures / content related (Course module viewed, Course viewed, Course activity completion updated, Course module instance list viewed, Content page viewed, Lesson started, Lesson resumed, Lesson restarted, Lesson ended).
- **Status1**: assignment related (Quiz attempt reviewed, Quiz attempt submitted, Quiz attempt summary viewed, Quiz attempt viewed, Quiz attempt started, Questions answered, Questions viewed, Submission re-assessed, Submission assessed, Submission updated, Submission created, Submission viewed).
- **Status2**: grade related (Grade user report viewed, Grade overview report viewed, User graded, Grade deleted, User profile viewed, Recent activity viewed, User report viewed, Course user report viewed, Outline report viewed).
- **Status3**: forum related (Post updated, Post created, Discussion created, Some content has been posted, Discussion viewed).
- **9 grades** (Week2_Quiz1, Week3_MP1, ... Week7_MP3).
- **36 logs** (Week1_Stat0, Week1_Stat1, Week1_Stat2, Week1_Stat3, ... Week9_Stat0, Week9_Stat1, Week9_Stat2, Week9_Stat3)

B. Approach

Dimensionality of the data is high, and not every feature contributes to the final grade as much. Before choosing features, correlation to the final grade is explored and shown in the image 1.

	Grade
Grade	1.000000
Week8_Total	0.972348
Week7_MP3	0.968130
Week5_MP2	0.953488
Week5_PR2	0.907837
Week3_MP1	0.901788
Week3_PR1	0.887352
Week7_PR3	0.865616
Week6_Quiz3	0.849920
Week4_Quiz2	0.810920
Week6_Stat1	0.771988
Week2_Quiz1	0.689783
Week4_Stat1	0.662946
Week3_Stat0	0.643789
Week6_Stat0	0.635807
Week4_Stat0	0.625359
Week3_Stat1	0.596824
Week5_Stat0	0.590146
Week8_Stat1	0.584425
Week9_Stat0	0.545532
Week9_Stat1	0.496753
Week5_Stat1	0.484030
Week8_Stat0	0.450807
Week7_Stat3	0.439733
Week7_Stat1	0.424807
Week2_Stat1	0.406120
Week7_Stat0	0.377195
Week2_Stat0	0.293296
Week2_Stat2	0.292262
Week6_Stat2	0.288761
Week4_Stat3	0.278034
Week6_Stat3	0.270239
Week5_Stat3	0.264079
Week1_Stat0	0.256311
Week8_Stat3	0.234907
Week3_Stat3	0.227106
Week2_Stat3	0.202950
Week4_Stat2	0.171987
Week9_Stat2	0.147822
Week9_Stat3	0.094227
Week7_Stat2	0.087466
Week5_Stat2	0.073326
Week1_Stat3	0.072546
Week8_Stat2	0.009186
Week1_Stat2	-0.129440
Week3_Stat2	-0.162950
Week1_Stat1	NaN

Fig. 1: *Correlation of features in regards to the final grade before removing features*

Based on this, a threshold was defined as $|0.4|$, meaning all features with correlation > -0.4 or < 0.4 will be removed. Those columns/features are: Week1_Stat1, Week3_Stat2, Week1_Stat2, Week8_Stat2, Week1_Stat3, Week5_Stat2, Week7_Stat2, Week9_Stat3, Week9_Stat2, Week4_Stat2, Week8_Total, Week2_Stat0, Week2_Stat2, Week6_Stat2, Week4_Stat3, Week6_Stat3, Week5_Stat3, Week1_Stat0, Week8_Stat3, Week3_Stat3, Week2_Stat3. Column Week1_Stat1 had all 0 values, so it was removed as well. Lastly, column Week8_Total had an accumulation of values, so it was not beneficial in terms of observing individual features.

21 columns were removed in total, and 16 statistics columns remained.

Additionally, results for mini projects, peer reviews and quizzes were summed within their respective groups, reducing the features in dataset by 6 (instead of having 3 separate entries for 3 separate activities, now there is only one entry for each activity type). Similarly statistics were grouped. Instead of having Stat0, Stat1, Stat2, and Stat3 for each week, they were grouped by week and now there are Week1_Stat, Week2_Stat, etc.

In the beginning data had 47 features/columns, and after all the processing, it was left with 12 features/columns.

Correlation is calculated once again for the remaining features and given in the table I.

Feature	Correlation
Grade	1.000000
MP	0.981128
Peer	0.925995
Quiz	0.850059
Week6_Stat	0.697617
Week4_Stat	0.679393
Week3_Stat	0.665728
Week5_Stat	0.591442
Week9_Stat	0.578879
Week8_Stat	0.514961
Week7_Stat	0.424443
Week2_Stat	0.406120

TABLE I: *Correlation with grade after removing features*

Grade is the variable which needs to be predicted, and its distribution is given in the image 2. It can be observed that the grade 1 does not appear, even though it exists in the grading system.

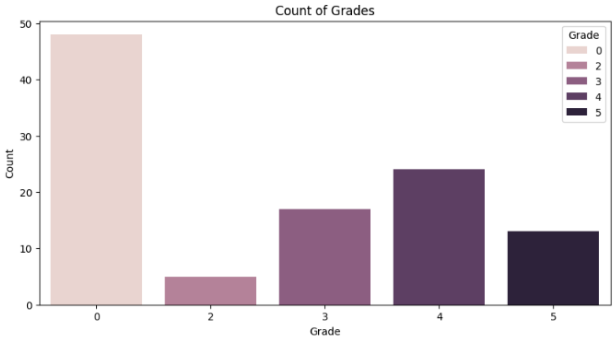

Fig. 2: *Distribution of grade variable*

C. Methods

Two methods were chosen for this project. Both regression and classification will be explored, with variants of XGBoost [1] and Random Forest [2] models. In order to be able to compare the two methods, predictions of regression were rounded to the nearest integer and afterwards treated as a classification problem.

Since the dataset is quite small, with only 107 entries, cross-validation was employed as a means to test and train the models. Even though the results it gave were not as good as the train-test split method, it is more suitable for a dataset as small as this in order to prevent over-fitting.

Models were compared based on size n, which is the number of decision trees used, MSE, which is the Mean Squared Error metric, R^2, which is coefficient of determination, and accuracy of the model.

R^2 (Eq. 1) metric measures how well the regression model fits the data. It represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1}$$

- y_i is the actual value of the target variable.
- $\hat{y}_i$ is the predicted value of the target variable.
- $\bar{y}$ is the mean of the actual values.
- n is the number of observations.

The numerator represents the sum of squared residuals (prediction errors), and the denominator represents the total sum of squares (variance of the actual values). An R^2 value closer to 1 indicates a better fit.

MSE (Eq. 2) measures the average squared difference between the actual and predicted values. It indicates how well the regression model performs in terms of prediction accuracy.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

- y_i is the actual value of the target variable.
- $\hat{y}_i$ is the predicted value of the target variable.
- n is the number of observations.

Lower MSE values indicate better predictive performance, as they imply that the predicted values are closer to the actual values.

Accuracy represents the number of correct predictions vs the total number of predictions the model made, and is in the range [0-1].

Number of decision trees used was a hyper-parameter with possible values [100, 200, 300, 400, 500]. The rest of the parameters were default, and even though exhaustive grid search was performed in order to try to find the best parameters, it did not yield significant results.

III. Data Analysis

Full accuracy overview sorted by accuracy is given in the table II.

Model	n	MSE	R^2	Accuracy
XGBoost Regressor	100	0.147619	0.962189	0.879654
XGBoost Regressor	300	0.147619	0.962189	0.879654
XGBoost Regressor	400	0.147619	0.962189	0.879654
XGBoost Regressor	500	0.147619	0.962189	0.879654
XGBoost Regressor	200	0.147619	0.962189	0.879654
RF Regressor	100	0.139827	0.960501	0.860173
RF Regressor	200	0.139827	0.960501	0.860173
RF Regressor	300	0.139827	0.960501	0.860173
RF Regressor	400	0.139827	0.960501	0.860173
RF Regressor	500	0.139827	0.960501	0.860173
XGBoost Classifier	400	0.215584	0.906936	0.812987
XGBoost Classifier	500	0.215584	0.906936	0.812987
XGBoost Classifier	100	0.215584	0.906936	0.812987
XGBoost Classifier	300	0.215584	0.906936	0.812987
XGBoost Classifier	200	0.215584	0.906936	0.812987
RF Classifier	500	0.413853	0.898751	0.803429
RF Classifier	400	0.414286	0.898751	0.803429
RF Classifier	200	0.480952	0.884640	0.794372
RF Classifier	300	0.452381	0.888630	0.794372
RF Classifier	100	0.470996	0.884630	0.775758

TABLE II: *Model performance comparison*

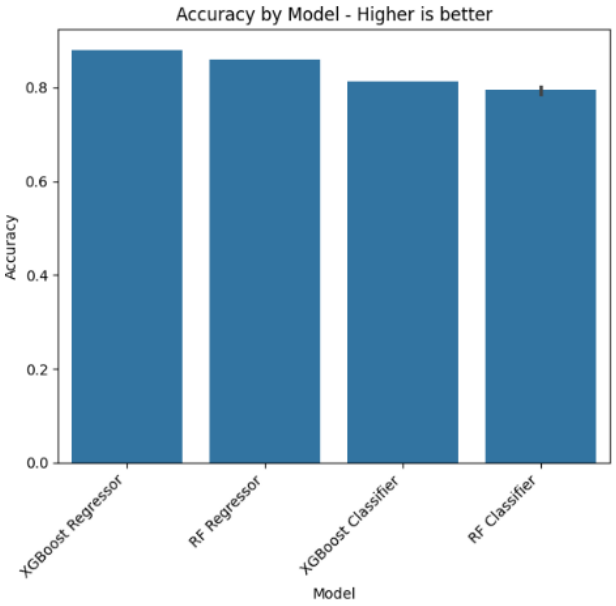

Fig. 4: *Accuracy by model*

Accuracy scores vs R^2 scores, colored by model, are given in the image 3.

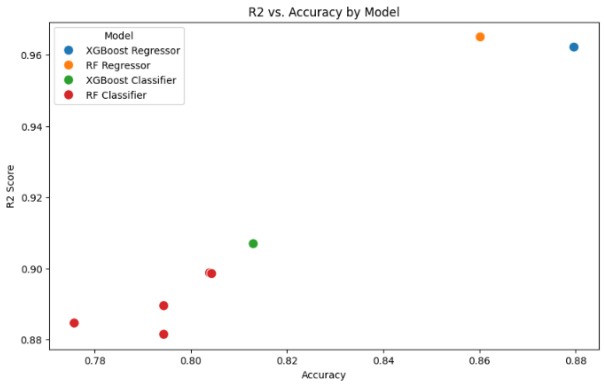

Fig. 3: *Accuracy vs R^2 by model*

Comparison of all three metrics by models is given in the images 4, 5, 6.

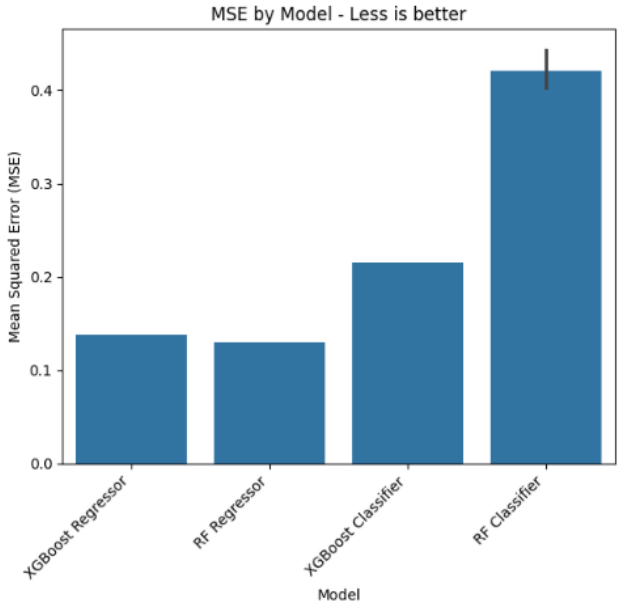

Fig. 5: *Mean Squared Error by model*

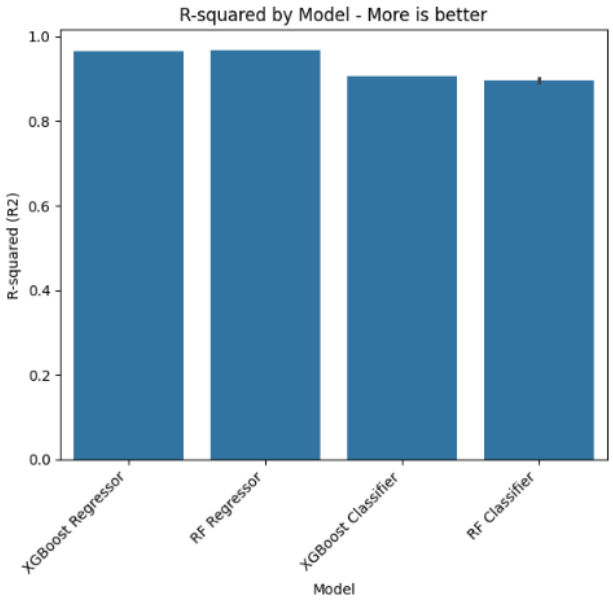

Fig. 6: R^2 *by model*

A. Visualization

Given that both XGBoost and Random Forest were trained with the number of estimators in the set [100, 200, 300, 400, 500], multiple observations can be made.

Accuracy vs the number of estimators comparison for each model is given in the images 7, 8, 9, 10. It can be observed that the accuracy changed **only** for Random Forest Classification, implying that the number of estimators doesn't lead to a change in the decision prediction, for some datasets (including this one). For the reference, ungrouping mini projects lead to more fluctuations in the accuracy based on the number of estimators, but the overall accuracy was reduced.

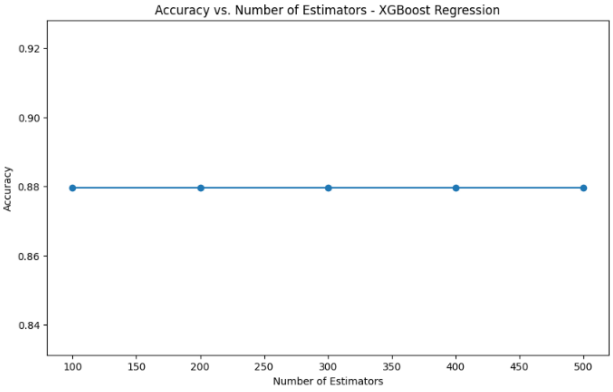

Fig. 7: *Accuracy by number of estimators for XGBoost Regression*

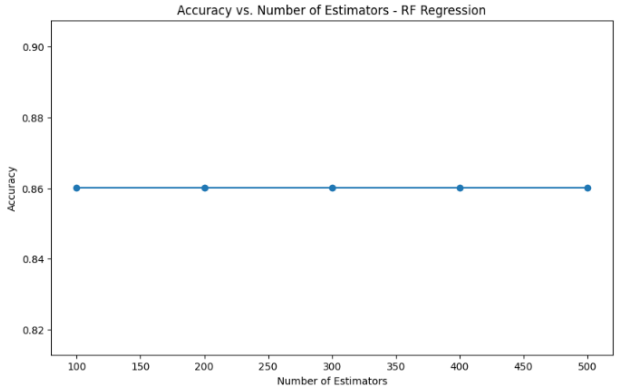

Fig. 8: *Accuracy by number of estimators for Random Forest Regression*

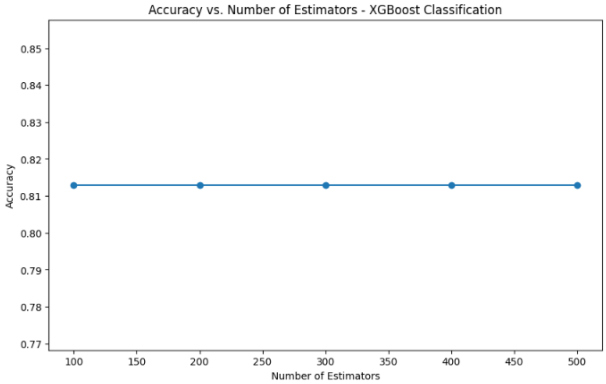

Fig. 9: *Accuracy by number of estimators for XGBoost Classification*

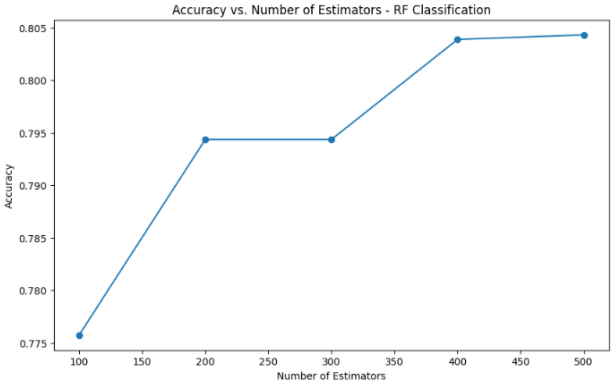

Fig. 10: *Accuracy by number of estimators for Random Forest Classification*

The best performing model from each of the categories was chosen and its confusion matrix and classification report are given. XGBoost Regression is given in the images 11, 12, Random Forest Regression in images 13, 14, XGBoost Classification in images 15, 16, and Random Forest Classification in images 17, 18.

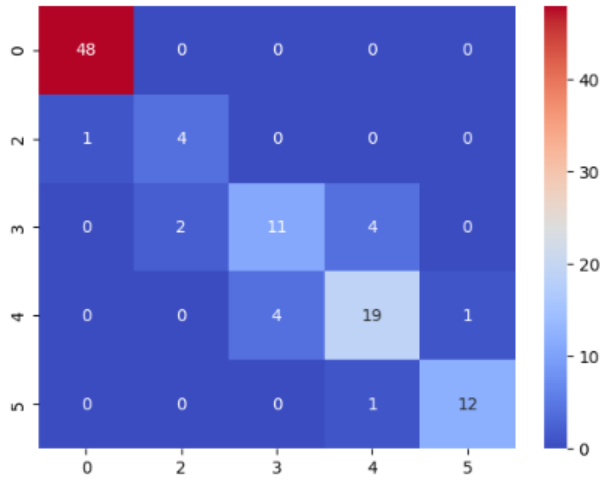

Fig. 11: *Confusion matrix for XGBoost Regression*

	precision	recall	f1-score	support
0	0.98	1.00	0.99	48
2	0.67	0.80	0.73	5
3	0.73	0.65	0.69	17
4	0.79	0.79	0.79	24
5	0.92	0.92	0.92	13
accuracy			0.88	107
macro avg	0.82	0.83	0.82	107
weighted avg	0.88	0.88	0.88	107

Fig. 12: *Classification report for XGBoost Regression*

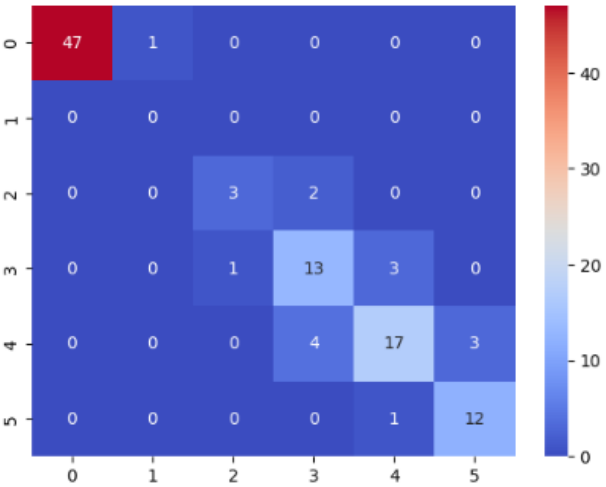

Fig. 13: *Confusion matrix for Random Forest Regression*

	precision	recall	f1-score	support
0	1.00	0.98	0.99	48
1	0.00	0.00	0.00	0
2	0.75	0.60	0.67	5
3	0.68	0.76	0.72	17
4	0.81	0.71	0.76	24
5	0.80	0.92	0.86	13
accuracy			0.86	107
macro avg	0.67	0.66	0.67	107
weighted avg	0.87	0.86	0.86	107

Fig. 14: *Classification report for Random Forest Regression*

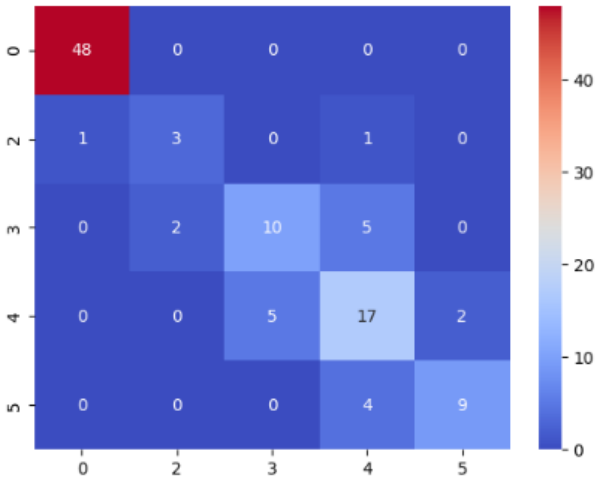

Fig. 15: *Confusion matrix for XGBoost Classification*

	precision	recall	f1-score	support
0	0.98	1.00	0.99	48
1	0.60	0.60	0.60	5
2	0.67	0.59	0.62	17
3	0.63	0.71	0.67	24
4	0.82	0.69	0.75	13
accuracy			0.81	107
macro avg	0.74	0.72	0.73	107
weighted avg	0.81	0.81	0.81	107

Fig. 16: *Classification report for XGBoost Classification*

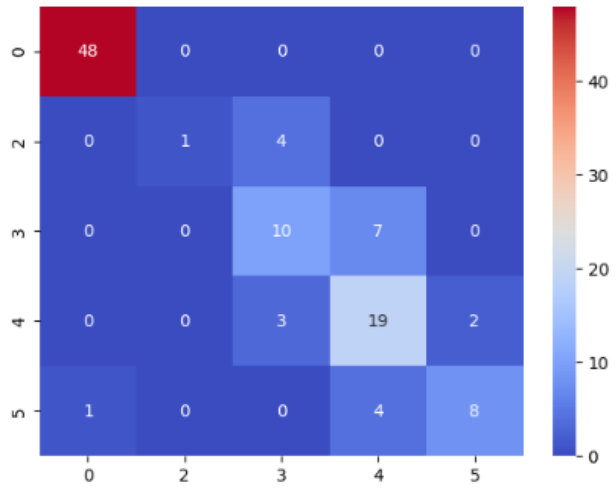

Fig. 17: *Confusion matrix for Random Forest Classification*

	precision	recall	f1-score	support
0	0.98	1.00	0.99	48
2	1.00	0.20	0.33	5
3	0.59	0.59	0.59	17
4	0.63	0.79	0.70	24
5	0.80	0.62	0.70	13
accuracy			0.80	107
macro avg	0.80	0.64	0.66	107
weighted avg	0.82	0.80	0.80	107

Fig. 18: *Classification report for Random Forest Classification*

B. Comparison

As it can be seen from the table II, the accuracy ranges from about 78% to 89%. This accuracy is not the highest possible, but there is little data to learn from. For reference, using cross-validation reduced the accuracy. Using train-test split made accuracy go as high as 92%. Additionally, combining columns for mini projects, peer reviews and quizzes all in one also increased accuracy to 96%, while accumulating all statistics columns increased it to 100%, but this could be over-fitting and important information could be lost, thus it wasn't used.

From the confusion matrices it can be observed that the models struggle with grades 3 and 4 the most, usually predicting them as each other, but grade 4 gets mistaken for grade 5 as well, and grade 3 for grade 2. This can suggest that the boundaries between these are not as clear to the models.

It can also be observed that XGBoost outperforms Random Forest in both regression and classification approaches, and given below are some reasons as to why.

- **Boosting vs. Bagging**: XGBoost uses boosting, where trees are built sequentially, and each subsequent tree aims to correct the errors made by the previous ones. This makes it more effective at minimizing prediction errors. Random Forest, on the other hand, uses bagging, where trees are built independently, making it less adaptive to errors.
- **Handling of Over-fitting**: XGBoost has built-in regularization parameters (like lambda and alpha) that help control over-fitting. This can be especially useful in datasets with smaller sizes (like the one used here) or high noise. Random Forest tends to rely on averaging multiple trees to reduce over-fitting.
- **Feature Importance Handling**: XGBoost assigns more importance to features that contribute more to reducing errors. This allows it to better capture the relationships between the most influential features and the target variable.
- **Gradient Descent Optimization**: XGBoost uses gradient descent in its learning algorithm, allowing it to find optimal parameter values for minimizing the loss function more effectively.

Finally, it can be seen that XGBoost Regressor with number of estimators in the set of [100, 200, 300, 400, 500], MSE equal to 0.15, R^2 equal to 0.96 and accuracy equal to 0.88 was the best performing. The three most important features for the model are given in the table III.

Feature	Importance
MP	0.969770
Quiz	0.007573
Week6_Stat	0.007007

TABLE III: *Feature importance for best performing model*

IV. Conclusion

In general we can see that regression models performed better than classification models. Their accuracy, MSE and R^2 error are all better than the classification models, which leads to the belief that even though this can be treated as a classification problem with 6 classes ([0-5]), it is better to treat it as a regression problem. Additionally, it can be noticed that XGBoost outperforms Random Forest in both approaches. Research shows that XGBoost often outperforms Random Forest in regression tasks due to its ability to handle more complex patterns in the data and perform better optimization. Even though grid search was performed for hyper-parameter tuning, it did not produce better results. If going only for the maximum accuracy, manipulation of data in the sense of accumulation leads to better results, but it is a question how generalized this is since the dataset is very small. Better hyper-parameter space exploration is always an option to try to predict more accurately for small datasets, and cross-validation is another approach to dealing with small datasets. Ultimately, it is up to the person using this to determine what they are most interested in and what methods suit their needs the best.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 11 of *KDD '16*, page 785–794. ACM, August 2016.

[2] L Breiman. Random forests, October 2001.