Mini Project II – Students Performance

Ali Kaya
Åbo Akademi University (ÅAU)
ali.kaya@abo.fi

October 14, 2023

Abstract

This machine learning initiative centered on the classification task of predicting the final grades for 107 enrolled students in a fully online, nine-week machine learning course, meticulously administered via the Moodle learning management system. The primary objective was to employ students' performance metrics, encompassing mini projects, quizzes, peer reviews, and the ultimate final grade, in conjunction with comprehensive course logs, to forecast their eventual grade scores. Our analytical journey encompassed essential phases, including exploratory data analysis, data preprocessing, feature selection, algorithm selection and model selection, with subsequent hyper-parameter fine-tuning for KNN, decision trees, and random forests. Remarkably, the culmination of our efforts consistently yielded highly accurate models, all surpassing the 77% accuracy threshold, exemplifying the effectiveness of our approach within the context of this classification problem.

1 Introduction

In the rapidly evolving landscape of education, the digital transformation has ushered in a new era of learning, where online platforms empower students to engage with courses and educational resources remotely. Within this paradigm, the accurate prediction of student performance in online courses has taken on pivotal significance. This predictive capability equips educators with the tools to proactively identify and support at-risk students, aligning interventions with individual needs.

In our pursuit of this goal, we acquired a dataset brimming with anonymized data from 107 enrolled students. Our project unfolds systematically, adhering to a structured workflow that navigates the essential stages of predictive modeling. It commences with a thorough exploration of the dataset, meticulously designed to unveil its inherent structure and unearth promising features for subsequent analysis. Notably, the data exploration phase is streamlined, facilitated by the dataset's comprehensiveness and cleanliness.

The critical juncture of feature selection invokes strategic methodologies such as correlation analysis and domain knowledge integration, facilitating the distillation of the most informative attributes for predicting student grades. Techniques like feature multicollinearity reduction play a pivotal role in streamlining the feature set. Subsequently, our project meticulously scrutinizes and compares various machine learning algorithm, initiating with default settings as the benchmarks of the models. The culmination of our efforts is the fine-tuning of pivotal models, including K-Nearest Neighbors (KNN), Decision Trees, and Random Forests, achieved through rigorous hyper-parameter optimization.

2 EDA, Data Preprocessing and Feature Selection

2.1 Exploratory Data Analysis (EDA)

EDA is the initial step in understanding our dataset. It involves examining and visualizing the data to gain insights into its structure, distribution, relationships between variables, and potential patterns. The dataset we used contains the following highlights:

- The dataset originally contains 107 rows(i.e. student records) and 48 columns (1 ID + 3 quiz grade + 3 mini project grades + 3 peer review grades + 1 total + 36 logs + 1 final grade)
- All columns in the dataset are of numerical data type except for the "ID" column, which is in string format
- There is no missing values, outliers in the dataset
- Min, Max for quizzes 1, 2, 3 are 0 and 5, type of continuous data

- Min, Max for project 1, 2 and 3 is 0 and 15, 0 and 20, 0 and 35, respectively, type of continuous data

- Min, Max for total are 0 and 99.710000, type of continuous data

- Min, Max for final grade are 0 and 5, type of discrete data

With the insights gained from the previous findings, the question becomes obvious: Given a dataset containing 47 features, how can we identify a robust classifier that can effectively predict a student's final grade, categorizing it within the possible grades from 0 to 5?

2.2 Data Preprocessing

Data preprocessing in this case is relatively straightforward, given the dataset's cleanliness, absence of missing values, and lack of outliers. However, we did perform the following specific preprocessing steps:

- Remove features with a single, uniform value because they do not provide meaningful information to aid our classification task('Week1 Stat1')

- Remove the 'ID' feature as it contains entirely distinct values with no discernible patterns.

2.3 Feature Selection

We, now, have 107 rows with 45 features and 1 labels. We can say that we have encountered a scenario where we are dealing with a dataset characterized by high dimensionality and a limited number of samples.

In order to tackle it, we choose to do Feature Selection/Reduction based on correlation analysis1. By doing this, we:

- Eliminate features exhibiting strong correlations (based on objective criteria), a standard preprocessing step known as feature correlation or multicollinearity removal.

- Retain only those features that display high correlations (based on subjective evaluation) with the target variable, considering the data's characteristics and problem objectives.

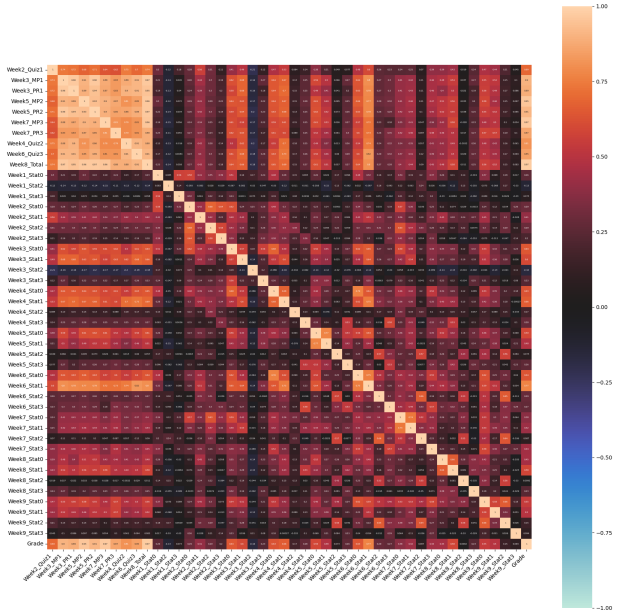

Figure 1: Heat-map for Correlation across all features and label

Specifically, we observe distinct patterns within the dataset. There is a notable high correlation pattern in the upper-left corner(1), as well as in the last row at the bottom-left(2). They are:

- (1) Robust correlations exist between individual scores (e.g., quiz, mini project, etc.) and the overall total score

- (2) Strong correlations are evident between all individual scores (e.g., quizzes, mini-projects, total scores, etc.) and the final grade.

After carefully weighing the pros and cons and aiming to preserve the original data structures and their information to the maximum extent possible, we opt to exclude feature 'Week8 Total', and filter out features that do not exhibit strong correlations with the final grade($<50\%$).
Now, we retain 18 features and 1 label in our dataset, they are 'Week2 Quiz1', 'Week3 MP1', 'Week3 PR1', 'Week5 MP2', 'Week5 PR2', 'Week7 MP3', 'Week7 PR3', 'Week4 Quiz2', 'Week6 Quiz3', 'Week3 Stat0', 'Week3 Stat1', 'Week4 Stat0', 'Week4 Stat1', 'Week5 Stat0', 'Week6 Stat0', 'Week6 Stat1', 'Week8 Stat1', 'Week9 Stat0', 'Grade'

3 Algorithm Selection

Given that we are addressing a classification problem, we can employ re-sampling techniques such as cross-validation to obtain estimates of how well each model is likely to perform on unseen data. We opt to employ various methods to assess the estimated accuracy of different machine learning algorithms and make an informed decision on the final selection. They can be:

"Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", "Decision Tree", "Random Forest", "Neural Net", "AdaBoost", "Naive Bayes", "QDA", "GradientBoosting", "SGD" and "Perceptron".
The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data. Here we use RepeatedStratifiedKFold with parameters n_splits=3, n_repeats=5, random_state=1 to get the corresponding cross validation scores for each algorithm and visualize them in a box-plot graph2.

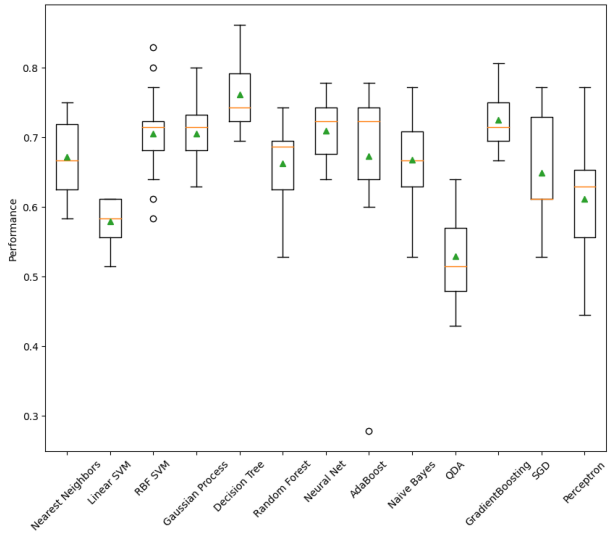

Figure 2: Model Comparison with default Hyper-parameters

Based on the results, it is evident that the Decision Tree outperforms other algorithms, achieving a notable performance advantage. Furthermore, all algorithms exhibit a minimum accuracy of 50%. Notably, the Decision Tree model, when using default hyper-parameters, displays manifest signs of overfitting.

4 Model Selection and Hyper-parameter Tuning

To fulfill the project's objectives and to explore various algorithms, we opted to employ a total of three algorithms for model training:

- k-Nearest Neighbors (k-NN): middle-level performance, we would like to see whether we can enhance the performance after fine-tuning the hyper-parameters
- Decision Tree: Higher performance, we would like to see whether overfitting exists and how good we can get for test data if we employee a fine-tuned Decision Tree model
- Random Forest, we would see if the overfitting exists for Decision Tree, how can an ensemble method that builds upon the foundation of Decision Trees can enhance the performance for test data

4.1 K Nearest Neighbours

K Nearest Neighbors (KNN) is a simple yet powerful supervised machine learning algorithm used for both classification and regression tasks. It is based on the principle of similarity, which assumes that data points with similar characteristics are often found in close proximity to each other in the feature space.
In this project, we try to tune KNN of our classification problem in the combination of the following ways:

- Pruning Features to keep the ones which are more correlated with the label (target variable) via setting higher correlation values.
- Employing grid search to optimize the hyper-parameter "K," which represents the number of neighbors to consider in the KNN algorithm
- Thresholding the accuracy of single training score, cross validation score and testing score to exam the reliability of grid search

4.1.1 K, the important hyper-parameter in KNN

The choice of K has a significant impact on the model's performance. Smaller values of K make the model more sensitive to local patterns, potentially capturing noise in the data, while larger values of K can make the model less sensitive but might smooth out important patterns. Here we illustrate how sensitive k-NN classification accuracy is to the choice of the 'k' parameter3.

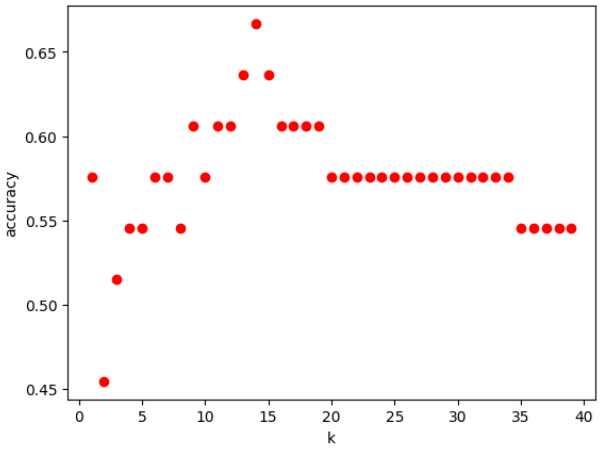

Figure 3: test accuracy score for K from 1 to 40 with 18 features2.3

4.1.2 Employ Grid Search to find 'best' K with different Feature combinations

Grid search is a hyper-parameter tuning technique used to systematically search for the best combination of hyper-parameter values for a machine learning model.
Moreover, Feature selection is important for K Nearest Neighbors (KNN). Remember although we remove the 'Week8 Total' before, some of the features are still highly correlated to each other(e.g. quiz scores, mini project scores and peer review scores, etc.). Thus, we calculate the best Ks using Grid Search with different correlations between features and labels to try to find the optimized combinations. The results are:

- Best k: 22 for correlation larger than 0.6, Test Accuracy: 0.59
- k: 2 for correlation larger than 0.7, Test Accuracy: 0.77
- Best k: 1 for correlation larger than 0.8, Test Accuracy: 0.86
- Best k: 2 for correlation larger than 0.9, Test Accuracy: 0.81

Among them, the corresponding features are:

- correlation $>$ 0.6: Week2 Quiz1, Week3 MP1, Week3 PR1, Week5 MP2, Week5 PR2, Week7 MP3 ,Week7 PR3, Week4 Quiz2, Week6 Quiz3, Week3 Stat0, Week4 Stat0, Week4 Stat1, Week6 Stat0, Week6 Stat1
- correlation $>$ 0.7: Week2 Quiz1, Week3 MP1, Week3 PR1, Week5 MP2, Week5 PR2, Week7 MP3 ,Week7 PR3, Week4 Quiz2, Week6 Quiz3, Week6 Stat1
- correlation $>$ 0.8: Week3 MP1, Week3 PR1, Week5 MP2, Week5 PR2, Week7 MP3 ,Week7 PR3, Week4 Quiz2, Week6 Quiz3
- correlation $>$ 0.9: Week3 MP1, Week5 MP2, Week5 PR2, Week7 MP3

But the result is not optimal since for more features(correlation = 6, 7), the test accuracy are low, while for less features(correlation = 8, 9), the 'best' k is too small, it is very likely to be over-fitted. We have to find other ways to determine the optimal K.

4.1.3 Thresholding the accuracy to explicitly find all the possible combinations of K

As mentioned before, the results given by the grid search is not that comprehensive. Here we rewrite the grid search with extra considerations, and try to explore more details when tuning about k:

- iterate correlation with feature selections: [0.6,0.7,0.8,0.9]
- iterate K: [1: 40]
- set up threshold for single train score, single test score and cross validation score45

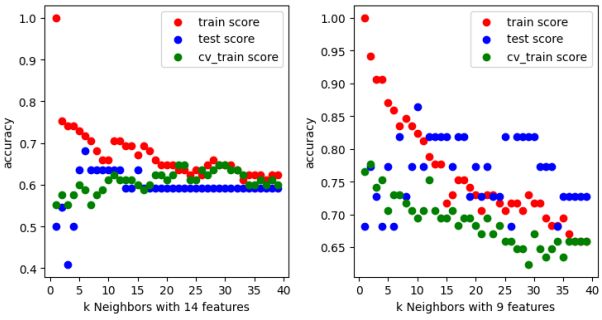

Figure 4: Best K for different correlation larger than 60% and 70%, respectively

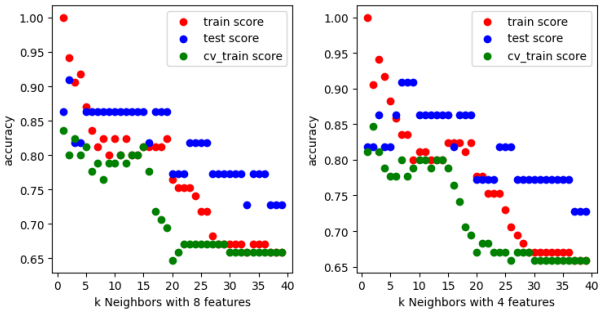

Figure 5: Best K for different correlation larger than 80% and 90%, respectively

the results can be seen as interpreted as follow:

- The cross validation scores(K=1)for training dataset is undoubted the highest one among all the others when correlation is set to 0.8. While the validation scores for test data performed same when K = 1, 5 and 15 in figure6
- The cross validation scores(K=2)for training dataset is undoubted the highest one among all the others when correlation is set to 0.9. While the validation scores for test data performed better when K = 3 in figure6

	K	Feature_Correlation	Training_Score	Testing_Score	Cross_vaalidation_Score_Mean
0	1.0	0.8	1.000000	0.863636	0.835294
1	3.0	0.8	0.905882	0.818182	0.823529
2	5.0	0.8	0.870588	0.863636	0.811765
3	15.0	0.8	0.811765	0.863636	0.811765
4	1.0	0.9	1.000000	0.818182	0.811765
5	2.0	0.9	0.905882	0.818182	0.847059
6	3.0	0.9	0.941176	0.863636	0.811765

Figure 6: Best K for different correlation larger than 80%

After considering the facts of feature retention, overfitting as well as testing performance, we opt to the

optimal K as 5 or 156 with 8 features4.1.2: Week3 MP1, Week3 PR1, Week5 MP2, Week5 PR2, Week7 MP3 ,Week7 PR3, Week4 Quiz2, Week6 Quiz3.

4.2 Decision Tree

A decision tree is a supervised machine learning algorithm that is used for both classification and regression tasks. It is a visual representation of a decision-making process that resembles an inverted tree. Each node of the tree represents a decision or test on a specific feature, and each branch represents the outcome of that decision.
It has several merits like interpretability, explicit variable importance, ready-to-use, and etc., while the drawbacks are obvious as well, like overfitting, instability and etc. In our case, we will see both of them in details.

4.2.1 Training with Grid Search

When Grid Search is applied to a decision tree model, it can help identify the optimal hyper-parameters that lead to the best model performance. With the selected features mentioned before2.3, We achieve this by using the following strategies:

- Define the Hyper-parameter Grid like: 'max_depth':[None, 2, 3, 4, 5, 6, 7, 8], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'class_weight': [None, "balanced"], etc.
- Perform Grid Search and Fit the Model
- Retrieve the Best Parameters and evaluate the corresponding model using some scoring metric.

The result of the best hyper parameters are: 'class_weight': None, 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2. Please be aware that the last three hyper-parameters come with internal constraints for a fixed dataset.
The final accuracy of Decision Tree with above-mentioned hyper parameters on training set is 0.98, while on testing set is 0.77. We can tell that overfitting exists since the performance on the training dataset is way better than the testing dataset.

4.2.2 Visualization of Decision Tree

It is quite straightforward to visualize a decision tree and the interpretability can be told through the graph here7. We can easily separate the target dataset tier by tier by just following the criteria given in the tree structure.

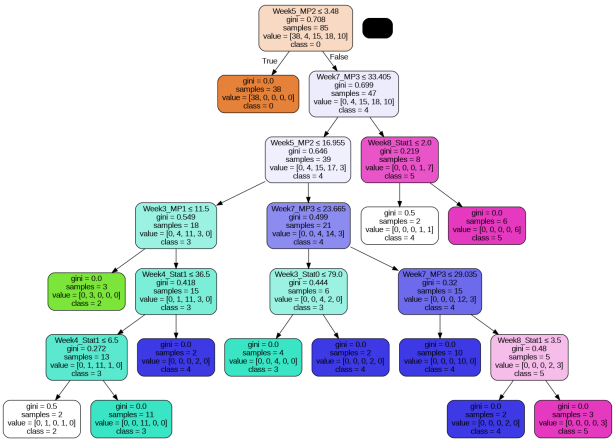

Figure 7: Decision Tree with Min_Sample_Left 2 and Min_Sample_Split 2

We note that only the node in the lower left corner have undivided data which cause the training accuracy to be less than 100%(0.98).

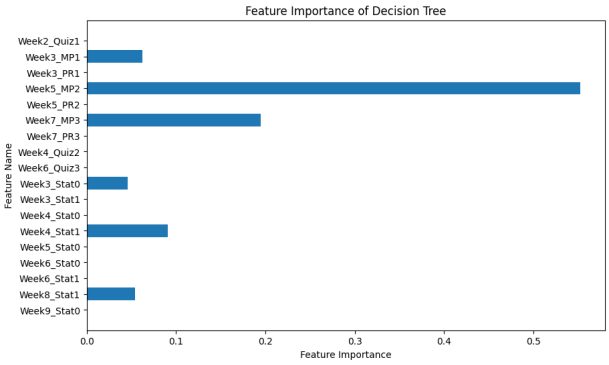

Figure 8: Feature Importance for Decision Tree

Besides the tree structure, we can also easily draw the feature importance as well8. The most influenced feature is Week5_MP2, followed by Week7_MP3, etc. This is evidently accurate as the scores for mini projects significantly influence a substantial portion of the total scores, which are subsequently converted into a final grade ranging from 0 to 5. However, we have to admit that the current optimal decision tree model is sensitive to small variations in the training data, meaning that a slight change in the data can result in a completely different tree structure.

4.3 Random Forest

Random Forest is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting. By averaging the predictions of multiple trees, Random Forest reduces the variance, making it less prone to overfitting compared to a single decision tree.
We can also employ grid search to identify the optimal hyper-parameters for a Random Forest, and the advantages of Random Forest typically outperform those of a standalone Decision Tree.

4.3.1 Training with Grid Search

When Grid Search is applied to a Random Forest model, it follows the same steps mentioned in the Decision Tree. With the selected features mentioned before2.3, We achieve this by using the following strategies:

- Define the Hyper-parameter Grid like: 'n_estimators': [100, 200], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [None, 10, 20], etc.
- Perform Grid Search and Fit the Model
- Retrieve the Best Parameters and evaluate the corresponding model using some scoring metric.

The result of the best hyper parameters are: 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200. It tells us that 200 sub trees will be used to form this random forest to give the best fit for the training dataset.
The Random Forest, using the specified hyperparameters, achieves a final accuracy of 1.00 on the training set, and on the testing set, it surpasses 0.8, reaching approximately 0.82. When compared to the Decision Tree, this ensemble algorithm demonstrates superior performance, excelling not only on the training dataset but also on the testing dataset.

4.3.2 Visualization of Decision Tree

Eventually we will have 200 sub trees in the forest and here we only illustrate No. 113 tree9 for visualization purpose.

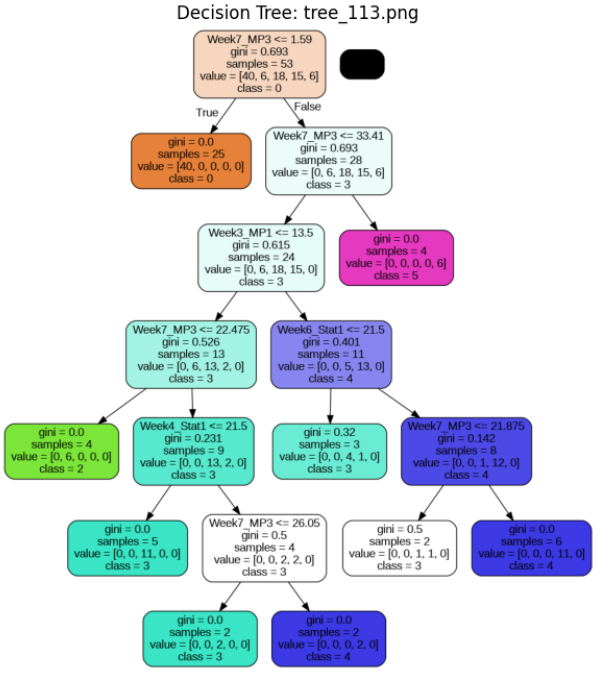

Figure 9: No.113 Sub Tree in Random Forest

It is evident that every leaf node within subtree No. 113 completely separates the given dataset into distinct and independent classes.

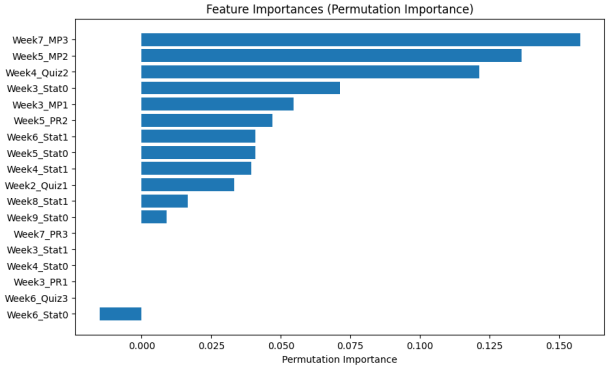

Figure 10: Feature Importance for Random Forest

Besides the tree structure, we can also easily draw the feature importance as well10. Comparing to the features importance graph of decision tree8 before, we find that more features are involved as the classifiers, which will help the model to reduce variance as well as strengthen generalization. Still, the most influenced two features are Week5_MP2 and Week7_MP3, which is logical and explainable.

5 Conclusion

5.1 Model Comparison

We employed three different models to conclude this project, and their performance showed slight variations. The Decision Tree achieved the lowest accuracy on the testing dataset, at 77%, while the K Nearest Neighbors (KNN) model managed to reach an accuracy of 86%.
The key advantage of K Nearest Neighbors (KNN) lies in its "lazy learner" nature, which eliminates the need for a dedicated training phase. Instead, we simply need to determine the optimal value of K to achieve the best-performing metric. However, the drawbacks are apparent, particularly in high-dimensional spaces, where the "curse of dimensionality" can impair KNN's effectiveness, as distances between data points lose their significance. This is why we introduced a feature pruning process based on correlations throughout the entire training procedure to enhance scoring performance.
In contrast, Decision Trees are less affected by feature dimensionality and remain highly interpretable, offering valuable insights into the decision-making process. However, the inherent challenge lies in their tendency to overfitting without proper pruning. When pruning is applied, it can of course result in lower scoring metrics, as the tree is generated from the root to the leaf nodes in its entirety.
Random Forest, an ensemble algorithm based on De-

cision Trees, mitigates some of the drawbacks associated with the latter. It effectively addresses the overfitting concern by amalgamating multiple trees, resulting in improved generalization. Consequently, Random Forest is capable of producing high-accuracy models while ensuring balanced feature utilization. However, as with many advantageous techniques, there is a trade-off: the computational complexity is notably high due to the intensive process of generating sub-trees.

5.2 Scientific Bottlenecks

During this project, I encountered two significant limitations. The first pertained to data inadequacy, highlighting the insufficiency of available data for a comprehensive analysis. The second limitation was attributed to my limited knowledge in the domains of data science, algorithms, and evaluation methods.
In a broad context, this specific classification problem is founded on a simple premise. The grading mechanism primarily relies on one of two approaches: it's either calculated using a polynomial equation based on the Week 8 total scores or directly adjusted according to these scores. Furthermore, the Week 8 total scores are essentially an amalgamation of individual scores from quizzes, mini-projects, and peer reviews, blended together in a specific manner. This situation implies that many of the log features play a less significant role when constructing models and making predictions. In the event that more data becomes available in the near future, it could potentially unveil implicit relationships between learning activities, i.e., logs, and the final grades.
This mini project marks my inaugural venture into a pure data science project, and it has been a significant learning experience. I've had to explore various facets within the realm of machine learning, which has presented its own set of challenges. As time continues to progress, my aspiration is to steadily gain mastery over a broader array of algorithms and evaluation methods. This, in turn, will enable me to construct more robust models with greater efficiency and precision.