<——————————————————————————————————————
https://archive.ics.uci.edu/dataset/856/higher+education+students+performance+evaluation
——————————————————————————————————————»

The dataset I chose for this coding challenge project is from the UCI Machine Learning Repository and was collected in 2019 from students in the College of Engineering and the College of Educational Sciences. This dataset provides information about demographic and academic factors and aims to predict students' final performance. My focus question in this project is: How do demographic and academic factors affect students' final grades? This coding challenge project aims to predict students' final grades through classification methods because we can identify patterns in demographic and academic factors and classify performance into predetermined outcomes. One initial hypothesis I proposed is that the cumulative grade point average (GPA) from previous semesters is an important factor affecting students' final grades, and factors such as attendance and study habits may also have an important impact on grades.

For data cleaning, I previewed the first few rows to confirm the data's structure, checked for missing and duplicate values and removed unnecessary columns. One observation was that the dataset included a Course ID, which does not hold any predictive value regarding student performance, so I removed it. This cleaning process ensures that the model was trained on a dataset free from noise and irrelevant information.

Two machine learning algorithms were actually tested for this project, such as decision trees and random forests. The general idea was to use machine learning algorithms to predict the students' performance and find the importance of each factor. I focused primarily on decision trees because they have several advantages. First, decision trees are highly interpretable because they can visually display the model's decision-making process, making it easier to understand and communicate the importance of features. Additionally, decision trees are particularly effective on categorical data, making them a good fit for solving this classification problem. They can also capture non-linear relationships between features, allowing for modeling complex interactions between factors such as GPA, study time, and final grades. The decision tree accuracy score was very low in the early stages, only 0.07. I thought this might be due to irrelevant data, but I tried using different values in the parameters to help improve the accuracy. I thought that changing the percentage of testing set from 0.2 to 0.3, helps with improving the accuracy by a little bit. It shows the problem might be my machine was trained well during the training (with a large sample) but performed poorly when the testing set was small, and had a bad prediction on unseen data.

I also tested the random state parameter, which controls the randomness of certain processes, like data splitting and shuffling, ensuring that the results are reproducible. From values 0 to 100, I found the highest accuracy with a random state of 0, though the overall performance was not highly sensitive to specific random splits. However, the accuracy doesn't fluctuate much, which is good, as it indicates that the model's performance is highly dependent on the data split.

The depth of decision trees impacts my accuracy by a bit, if the depth of the tree is too high, it results in very high accuracy on the training set but low accuracy on the test set due to overfitting. If the depth is too low, it is underfitting that it doesn't have enough layers to capture the patterns in the training data. Another concern would be the overlap of the nodes, but I also played with the depth of the tree, the width, length and font size to reduce the overlapping as much as possible.

Another problem that I identified was the "zero division" when I submitted to GitHub. It says, "Precision is ill-defined and being set to 0.0 in labels with no predicted samples. "I tried to use the 'zero division = 1' parameter in the decision tree to control this behavior as it prompted, but it only helped with some parts.

Later in my project, I experimented with the random forest algorithm to assess whether it could improve the accuracy of my model. The random forest algorithm operates as an ensemble of decision trees, each trained on a different random subset of the data, and predictions are made by averaging the results of each tree. By combining multiple decision paths, this approach helps alleviate the overfitting problem typically associated with a single decision tree. Random Forest's structure typically enhances model robustness by reducing variance and smoothing out anomalies present in individual trees.

However, my results showed that the random forest's accuracy remained similar to that of the single decision tree, even after adjusting several hyperparameters, such as the value of sample splits and n_estimators. The performance gains were minimal, and I suspect that the limited improvement may be due to the dataset's relatively small size, which contains only 145 instances. Random forest models generally perform better with larger datasets, as more data allows each tree to discover unique patterns and insights. In a smaller dataset like this one, the additional trees might not have sufficient new information to learn from, limiting the effectiveness of Random Forest's ensemble approach in this case.

Overall, I would prefer using decision trees over random forest and linear regression because the nature of the Target (Grades) is categorical (e.g., AA, BB, CC, Fail), which is suitable for a classification task. I can do regression if I convert grades into numerical values (e.g., 0, 1, 2,... for ordinal grades), but that's not ideal because grades are not continuous. Compared to the random forest, the decision tree leaves more room for interpretation, as you can understand how the model makes decisions based on different features, and the random forest might be better if the dataset has a relatively bigger size.

Lastly, the evaluation metrics used in this report were accuracy, precision, recall, f1-score, and confusion matrix, providing insight into how well the model performs and where there is room for improvement. The overall accuracy of the model using decision trees is 34%, which means that only one out of every three students' grades were predicted correctly. Such low accuracy suggests that the model may not be accurate in other areas of prediction. **Precision** focuses on the model's performance in terms of positive predictions, explaining what proportion of the predicted instances of a class were correct. For class 1, the precision is 0.38, meaning that only 38% of the samples predicted to be in class 1 were actually correct. **Recall** explains how many instances of a class were correctly predicted of all the actual instances of a class. For class 1, the recall is 0.69, meaning the model correctly identified 69% of the actual samples belonging to class 1. A higher **F1-score** indicates better performance. For example, class 1 has an F1-score is 0.49, showing moderate performance for this class. **Support** is the number of instances for each class in the test set. For example, my test set has 13 instances of class 1, which is reflected under the support column. I have low precisions and recalls; classes 0, 4, and 6 have precision, recall, and F1-scores of 0**.** The precision and recall for these classes are 0.0, indicating that the model either doesn't predict these classes or inaccurately predicts them.

Accuracy is the ratio of correct predictions (both true positives and true negatives) to the total number of predictions made. The overall accuracy of 34% is quite low, which means the model needs improvement. This could be due to class imbalance, insufficient data, or feature relevance. Macro averages (Precision, Recall, F1-Score) are unweighted averages of all classes' precision, recall, and F1-scores. For instance, the macro-average precision is 0.25, showing that, on average, precision is relatively low across all classes. Weighted Average accounts for each class's instances (support). A weighted average gives more weight to classes with more samples.

The **confusion matrix** at the bottom shows each class's actual vs. predicted classifications. Each row represents the true class, while each column represents the predicted class. There are two ways of interpreting the confusion matrix, 1)**Diagonal values**: These represent correct predictions (true positives). For example, there are 9 correct predictions for class 1 (row 2, column 2) and 6 correct predictions for class 2 (row 3, column 3).

2) **Off-diagonal values**: These are misclassifications. In the second row (true class 1), 3 instances of class 1 were wrongly predicted as class 0, and 1 instance was predicted as class 5. In the third row (true class 2), 1 instance of class 2 was wrongly predicted as class 1. There are several zero predictions, meaning the model did not predict certain classes at all (e.g., class 0, class 4, and class 6). The confusion matrix shows that the model performs relatively well at predicting some classes

(such as classes 1 and 5) but performs poorly at predicting other classes (such as classes 0, 4, and 6), as evidenced by the many off-diagonal values and zero predictions for some rows.

In conclusion, my focus for this dataset is to predict and understand how the students' demographic and academic factors influence a student's final grade by using decision trees and random forests in classification. I'm aware of my low accuracy in the results, but I tried to do hyperparameter tuning to improve it, and this experience of testing different parameters helped me gain a deeper understanding of how these different factors impact the accuracy of my results.

—---

Any outside resources that you use (cite your sources!

https://s3.amazonaws.com/assets.datacamp.com/email/other/ML+Cheat+Sheet_2.pdf

https://images.datacamp.com/image/upload/v1676302389/Marketing/Blog/Scikit-Learn_Cheat_Sheet.pdf

https://www.geeksforgeeks.org/data-visualization-using-matplotlib/

https://www.datacamp.com/blog/machine-learning-models-explained

https://www.datacamp.com/tutorial/tutorial-data-cleaning-tutorial

https://campus.datacamp.com/courses/cleaning-data-in-python/common-data-problems-1?ex=1

https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/classification-1?ex=1

https://archive.ics.uci.edu/dataset/856/higher+education+students+performance+evaluation

https://saturncloud.io/blog/how-to-use-pandas-with-jupyter-notebooks/

https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html

https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/classification-1?ex=7

https://github.com/mGalarnyk/Python_Tutorials/blob/master/Sklearn/CART/Visualization/DecisionTreesVisualization.ipynb