

Data mining Challenge: Report

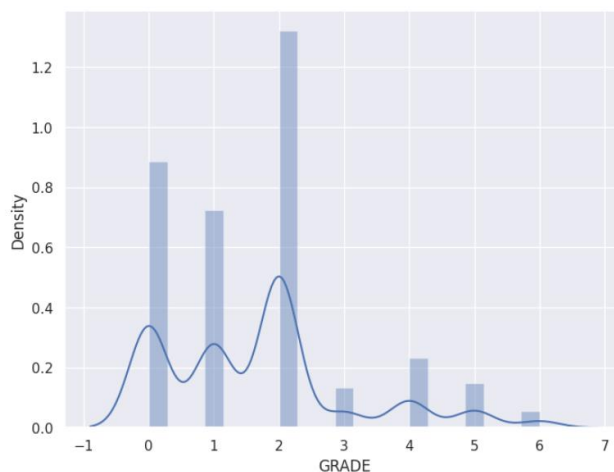
Predicting Students' Grades

In this data mining challenge, our primary objective is to predict the grades of students based on their characteristics. The training dataset provided contains the grades and characteristics of 80% of the students. Our task is to use this information to predict the grades of the remaining 20% of students, whose characteristics are given in the test dataset. The grades to be predicted range from 0 to 6, with 0 being the lowest and 6 being the highest.

To achieve this objective, I have implemented various classification machine learning (ML) methods. Additionally, I conducted thorough feature analysis and feature engineering to gain deeper insights into the provided data and improve the prediction accuracy.

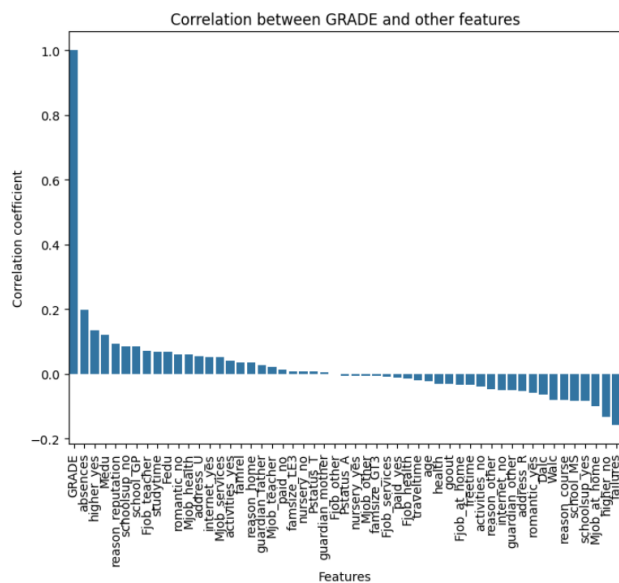
I. Feature Analysis and Future Engineering

- **Data Imbalance:** The first step was to check whether the data was balanced or imbalanced. I calculated the percentage of each grade in the training dataset. As shown in the figure below, the dominant grades were 2, 0, and 1, accounting for approximately 84% of the total grades. Initially, most models predicted only grades 2 and 0 for the test data, leading me to consider resampling. I applied oversampling to the under-represented grades and undersampling to the over-represented grades. While this approach improved performance on the training dataset, it resulted in worse performance on the test dataset. Consequently, I decided to use the original data to maintain generalization.



- **Missing values:** I checked for missing values, as they can hinder feature engineering and data visualization efforts. Fortunately, the dataset did not contain any missing values, ensuring a smooth analysis process.

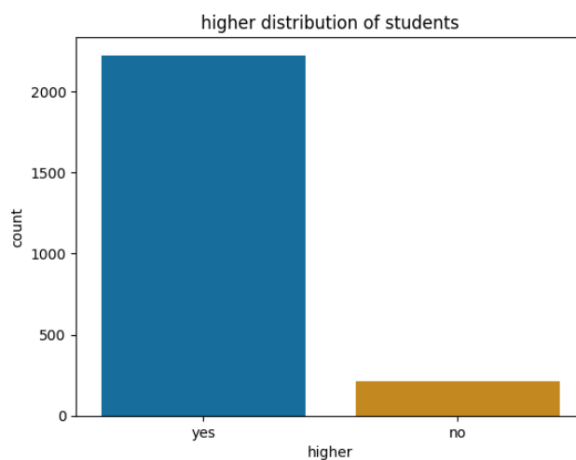
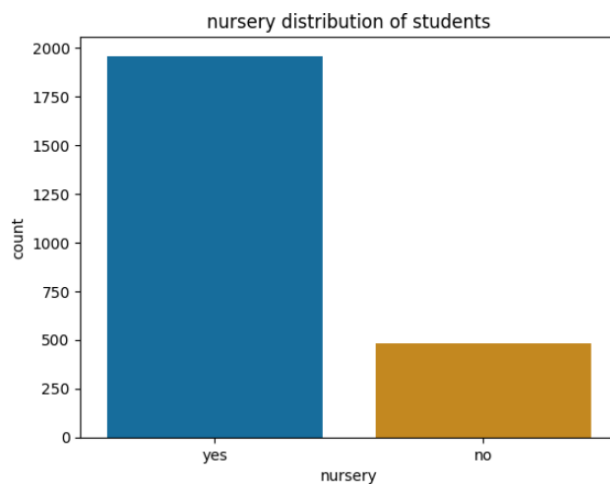
- **Data types:** I examined the data types of the features and found that some were categorical. Categorical data types can be challenging to handle with NumPy, numerical analysis, and visualization tools. Therefore, I transformed these categorical features into numerical data types using techniques like one-hot encoding. This transformation facilitated better handling and analysis of the data.
- **Different ranges of features:** The features exhibited different ranges. For example, the 'age' column ranged from 15 to 22, the 'health' column ranged from 1 to 5, and the transformed categorical features had values of 0 and 1. It was crucial to scale these variables because algorithms like KNN and SVM are sensitive to feature magnitudes and might be biased towards features with larger ranges. I used standard scaling to ensure all features contributed equally to the model.
- **Correlation between Features:** To reduce complexity and avoid overfitting, it was important to minimize the number of features. I computed the correlation between each feature and found that most features had low correlations with each other. Additionally, I calculated the correlation of each feature with the 'GRADE' feature. Some features, such as 'famsup_no', 'guardian_mother', 'guardian_father', and 'sex', showed negligible correlation with grades. On the other hand, features like 'absences', 'higher_yes', 'Medu', 'reason_reputation', 'schoolsup_no', and 'school_GP' had strong positive correlations with grades, indicating that higher values of these features were associated with better grades. Conversely, features like 'failures', 'higher_no', 'Mjob_at_home', 'schoolsup_yes', and 'school_MS' had strong negative correlations with grades. The strong positive correlation between absences and grades was counter-intuitive and surprising, as absences are typically expected to result in lower grades. This anomaly could not be easily explained.



Feature Selection: To determine the optimal set of features, I tested the performance of the models by dropping features with near-zero correlation with grades. The best performance was achieved when I dropped the 'sex' and 'famsup' features. This indicates that these features did not contribute significantly to the prediction of grades.

Assumptions and Exploratory Analysis: Initially, I hypothesized that some features might be common among all students and therefore not significant in differentiating grades. For instance, I assumed that most students might have attended nursery school, making this feature less impactful. To validate these assumptions, I plotted bar charts for some features to visualize their distribution and assess their relevance.

By carefully analyzing the data and applying feature engineering techniques, I aimed to enhance the predictive performance of the machine learning models. This process included handling data imbalance, transforming categorical data, scaling features, examining correlations, and selecting the most relevant features.



Almost all of the students in the training data set want to attend a higher education, so this feature might not be a differentiating factor.

II. Model Selection and Evaluation

To identify the most effective models for predicting student grades, I conducted a thorough evaluation process. Before applying the models to the test data, I aimed to determine which models would likely perform best and to narrow down my choices. To achieve this, I employed a cross-validation approach on the training dataset.

Cross-Validation: I divided the training data into multiple folds and used cross-validation to assess the performance of various models. This approach provided a robust estimate of each model's performance and helped mitigate the risk of overfitting.

```
,
models = [
    LinearDiscriminantAnalysis(), # LDA
    QuadraticDiscriminantAnalysis(), # QDA
    DecisionTreeRegressor(),
    RandomForestClassifier(),
    GradientBoostingClassifier(),
    AdaBoostClassifier(),
    SVC(),
    KNeighborsClassifier(),
    GaussianNB(),
    LogisticRegression(),
    XGBClassifier(),
    KMeans(n_clusters=7)
]

for model in models:
    cv_scores = cross_val_score(model, X_train, y_train, cv=5)
    print(f"{model.__class__.__name__}: Mean CV score = {cv_scores.mean()}, Std CV score = {cv_scores.std()}")
```

Initial Model Evaluation: Based on the cross-validation results, the following models demonstrated promising performance:

- K-Nearest Neighbors (KNN)
- Random Forest Classifier
- Gradient Boosting Classifier
- Support Vector Classifier (SVC)

Given their strong performance, I decided to focus on these models and proceed with hyperparameter tuning to optimize their performance further.

Hyperparameter Tuning with Optuna: To fine-tune the hyperparameters of these models, I utilized Optuna, an efficient hyperparameter optimization framework. Optuna's ability to perform a wide search of the hyperparameter space helped in finding the optimal settings for each model.

K-Nearest Neighbors (KNN): The optimal number of neighbors was found to be 104, balancing the bias-variance trade-off and enhancing the model's predictive accuracy.

Support Vector Classifier (SVC): The best hyperparameter values were: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}. These settings allowed the SVC to perform well by adjusting the regularization parameter and kernel type, enabling it to handle the complex decision boundaries in the data.

Random Forest Classifier: The optimal hyperparameters were: {'n_estimators': 213, 'max_depth': 29, 'min_samples_split': 4, 'min_samples_leaf': 12}. These parameters improved the model's ability to capture intricate patterns in the data while preventing overfitting.

Gradient Boosting Classifier: Similarly, for the Gradient Boosting Classifier, I performed hyperparameter tuning.

Feature Selection and Model Training: After tuning the hyperparameters, I revisited the feature selection process. By analyzing the feature importance scores and their correlation with the target variable, I identified and dropped features that contributed little to the model's performance. This step not only reduced the complexity of the models but also potentially enhanced their generalizability.

Model Evaluation: I evaluated the performance of the optimized models using cross-validation on the training data. This included metrics such as accuracy, precision, recall, and F1-score. Additionally, I assessed the confusion matrix to understand the distribution of prediction errors across different grade categories.

Final Model Selection: Based on the cross-validation performance and the feature selection results, I chose the models with the best performance for final testing. The selected models were then trained on the entire training dataset using the optimized hyperparameters.

Ensemble Method: After evaluating the individual models, I explored the potential of combining their predictions to further improve performance. This approach leverages the strengths of multiple models to produce a more robust and accurate prediction. I implemented a simple yet effective ensemble method by averaging the predictions of the individual models. The benefit of this approach is that different models may capture different aspects of the data. The performance of each model increased when I averaged their predictions.







III. Conclusion

The ensemble method, particularly the averaging of predictions from KNN, Random Forest, Gradient Boosting, and SVC, demonstrated better performance in predicting student grades. This approach highlights the effectiveness of combining multiple models to achieve more accurate and reliable predictions.

Melese Medhin
20210727

Public Private

The private leaderboard is calculated with approximately 50% of the test data.
This competition has completed. This leaderboard reflects the final standings.

#	Δ	Team	Members	Score	Entries	Last	Solution
1	▲ 10	Jihoo Kang		0.45901	28	8d	
2	▲ 21	CC		0.43934	38	5d	
3	▲ 1	Angela Hu		0.43606	11	5d	
4	▼ 2	Chan-Min Lee		0.42622	97	5d	
5	▲ 10	smile_one_tool		0.42622	78	5d	
6	▲ 11	mmt		0.42622	85	5d	

User name = mmt

Ranking = 6

Score = 0.42622

Entries = 85