Mini Project-2

Predicting Student's Final Grade

Introduction:

The goal of this project is to predict student's grades from a given dataset, process the data, extract features, spilt dataset for model training and evaluate their performances. Also, the project provides three important features that are used to predict the grades.

Dataset Overview:

The dataset contains anonymized records for 107 students enrolled in the course. The primary features include:

- 9 grades from quizzes and mini-projects (Week2_Quiz1, Week3_MP1, Week4_Quiz2, Week5_MP2, Week6_Quiz3, Week7_MP3).

Data Preprocessing:

After having on overview on the dataset, to process the data, finding missing values, selecting feature and splitting data to the further model training are the tasks to do.

- **Handling Missing Values:** During initial data exploration, no missing values were identified in the rows and columns.
- **Preliminary Feature Selection:** ID is irrelevant for the prediction of final grade. Also, Week8_Total is just an aggregation of all the available grades in quizzes, mini-projects, and peer reviews. So, we can safely discard the feature. We could have discarded all the individual grades and kept the overall only. However, in that case it would not be possible to measure the impact of the grade of each assessment on the final grade.
- **Categorizing Target Variable:** The target variable 'Grade' is categorized.

Data Split:

The class distribution (Figure-1) is imbalanced. So, we must use stratified sampling during split. We kept 20% data for testing and the rest of the 80% for training.

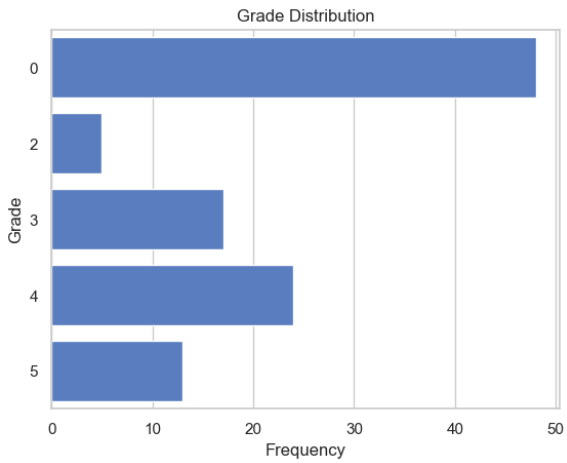

Figure-1: Grade Distribution

Mini Project-2

Predicting Student's Final Grade

Model Training:

The primary challenge of this dataset is the imbalance between the large number of features and the limited amount of data available.

As a result, more complex models are likely to suffer from overfitting. To address this issue, the use of regularization is necessary to control model complexity and improve generalization. To mitigate overfitting, I propose employing both a simple and a complex model, with regularization applied as appropriate.

At the very beginning, for the simple model, I considered two options: Naive Bayes and K-Nearest Neighbors (KNN). However, KNN is not well-suited for high-dimensional data with a small number of data points, making Naive Bayes the more appropriate choice.

For the complex model, I evaluated Logistic Regression, Decision Tree, and Support Vector Machine (SVM). While SVM is effective in handling high-dimensional data.

I selected Decision Tree due to its interpretability. Although Logistic Regression and Decision Tree both support extensive regularization techniques to prevent overfitting, the Decision Tree model offers the additional advantage of producing easily interpretable results.

In summary, I have chosen Naive Bayes as the simple model and Decision Tree as the complex model, both of which are suitable given the dataset's characteristics and the need to prevent overfitting through regularization.

Feature Importance:

To understand the driving factors behind students' performance, feature importance was analyzed using the Random Forest model. The three most important features in predicting final grades were:

Week7_MP3

Week5_MP2

Week2_Quiz1

Week5_Stat1

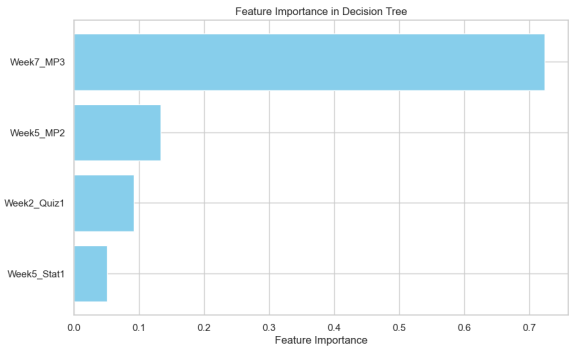

Figure-2: Important Features

Mini Project-2

Predicting Student's Final Grade

Performance Evaluation:

Since the data is imbalanced, accuracy would not be a good measure for comparing models. I could use either precision or recall.

Instead, I used F1-Score to maximize both. I employed a randomized search technique to find the best set of hyperparameters that reduce overfitting on the validation set.

The best model of Decision Tree provided an average F1-Score of 0.63±0.18.

On the other hand, the best Naive Bayes model had an average F1-Socre of 0.53±0.12.

There might be several reasons for Decision Tree to perform better than Naive Bayes. For example, Decision Trees are adept at capturing complex, non-linear relationships and feature interactions. If your dataset contains features that interact with each other in a non-trivial way, the Decision Tree could model these interactions more effectively.

In contrast, Naive Bayes assumes conditional independence between features, meaning it does not model interactions between them. If your features are not independent, Naive Bayes would likely perform worse.

From our dataset, it is obvious that the features have interactions between them. Another reason might be that it was possible to regularize decision tree by tuning different hyperparameters.

However, Naive Bayes is often a high bias model with simplistic assumptions.

As Decision Tree is giving a good F1-Score, we can have a Confusion Matrix and a Classification Report to justify.

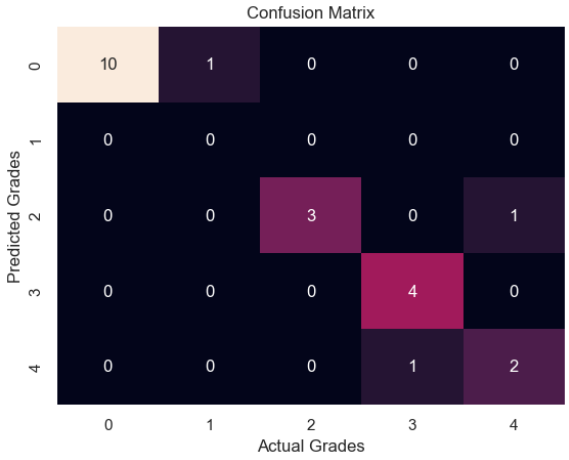

Figure-3: Confusion Matrix for Decision Tree

	precision	recall	f1-score	support
0	1.00	0.91	0.95	11
2	0.00	nan	0.00	0
3	1.00	0.75	0.86	4
4	0.80	1.00	0.89	4
5	0.67	0.67	0.67	3
accuracy			0.86	22
macro avg	0.69	0.83	0.67	22
weighted avg	0.92	0.86	0.88	22

Figure-4: Classification Report

Mini Project-2

Predicting Student's Final Grade

From Figure 3, the classification report shows Decision Tree has good accuracy and performs better.

Conclusion:

There have been a few bottlenecks as:

- The primary challenge of this dataset is the imbalance between the large number of features and the limited amount of data available. As a result, more complex models are likely to suffer from overfitting.
  To address this issue, the use of regularization is necessary to control model complexity and improve generalization. To mitigate overfitting, I propose employing both a simple (Naive Bayes) and a complex model (Decision Tree), with regularization applied as appropriate.
- As, the data is imbalanced, it was tough to measure accuracy. Precision or Recall can be used here.
  After analyzing, I used F1-Score, and it maximized both.