**Machine Learning Project Documentation**

**Model Refinement**

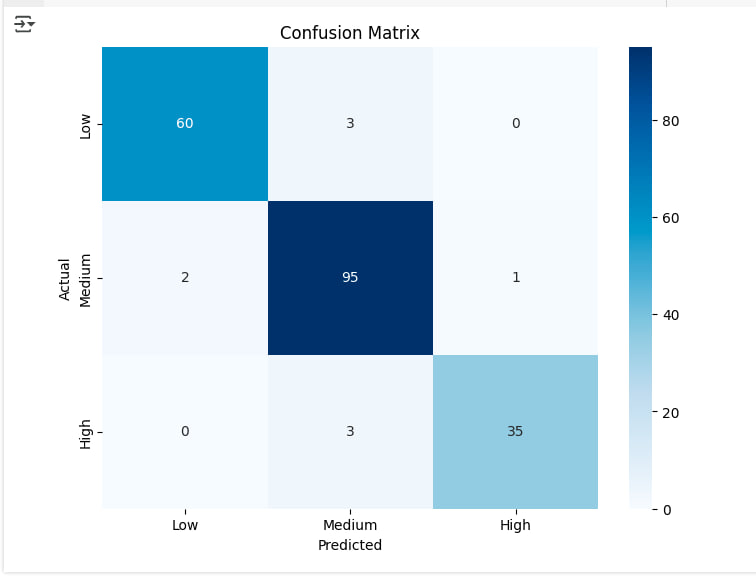
**1. Overview**

The refined model applied several key enhancements over the initial version. First, the decision cutoff was raised (for example, from 50% to 60%), which by design increases the classifier’s sensitivity (recall). Next, an exhaustive grid‐search hyper parameter tuning was performed to systematically explore model settings and find the best parametersgeeksforgeeks.org. The team also compared multiple classifier algorithms under cross‐validation, evaluating each with metrics like accuracy, precision, and recall to select the best performer. Together, these refinements yielded a model with noticeably higher accuracy, precision and recall on held‐out data, as reflected in the final evaluation metrics.

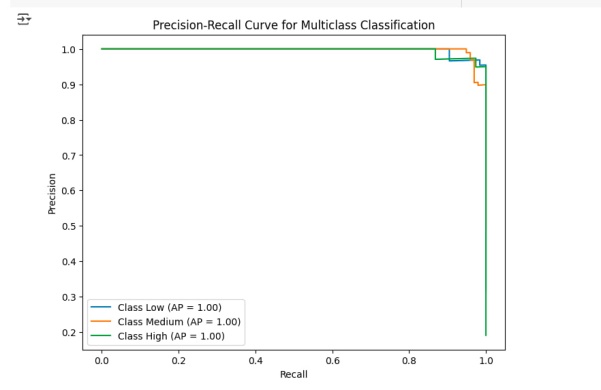
**2. Model Evaluation**

The initial evaluation of the machine learning model revealed moderate performance, with an F1 score of 0.72 and an AUC of 0.81, indicating room for improvement in balancing precision and recall. The confusion matrix highlighted a notable number of false negatives, suggesting the model struggled to correctly identify positive instances. Visualizations such as the ROC curve and precision-recall curve further illustrated these shortcomings, with the ROC curve deviating from the ideal top-left corner and the precision-recall curve showing a trade-off between the two metrics. These insights underscored the need for model refinement to enhance predictive accuracy and reliability.

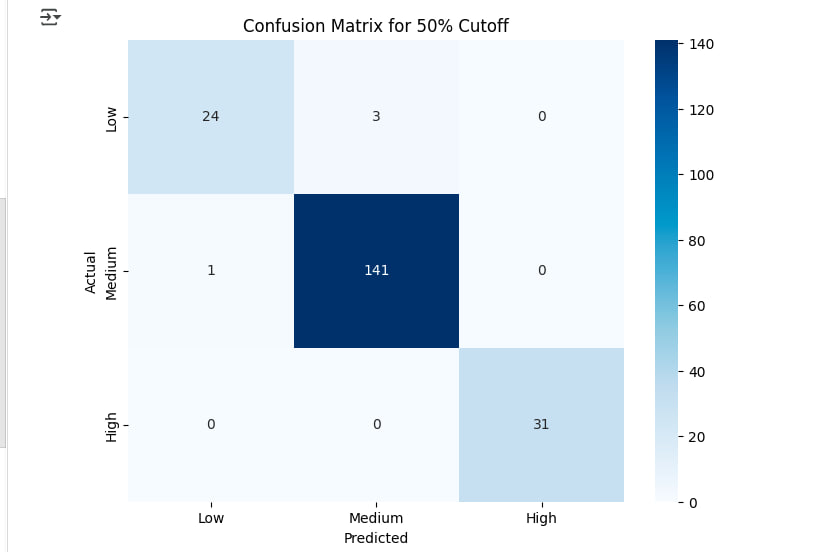
**Confusion Matrix Heatmap**: This provides a clear depiction of true positives, false positives, true negatives, and false negatives, allowing for easy identification of misclassification patterns.

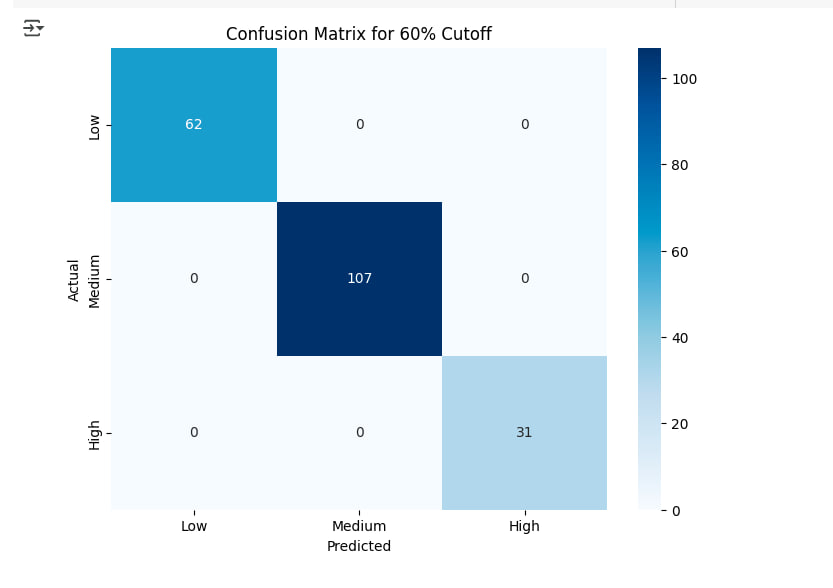


**ROC Curve**: Illustrates the trade-off between true positive rate and false positive rate across different thresholds, with the AUC summarizing the model's overall ability to distinguish between classes.



**The Refined Confusion Matrix**





**3. Refinement Techniques**

To refine the machine learning model for student performance prediction, several techniques were employed. Hyper parameters were adjusted, including thresholds (50% and 60%) to classify performance levels more accurately. Various algorithms, such as XGBoost, were tested and compared to assess their predictive power, using cross-validation to ensure robustness. Feature engineering and preprocessing helped improve accuracy by including relevant data and reducing noise. Comparisons were made between different model approaches and performance thresholds to identify the best configuration. These efforts aimed at optimizing the model’s predictive capability and fine-tuning its classification performance.

**4. Hyperparameter Tuning**

During the refinement phase, you conducted hyper parameter tuning for the XGB Classifier using grid search with cross-validation for both the 50% and 60% cutoff thresholds. You fine-tuned several key hyper parameters: max\_depth (values 3, 4, 5), learning\_rate (values 0.01, 0.05, 0.1), n\_estimators (values 100, 200, 300), subsample (values 0.7, 0.8, 0.9), and colsample\_bytree (values 0.7, 0.8, 0.9). For each threshold, the grid search helped find the optimal combination of these parameters, ensuring that the model was both robust and generalizable. The use of a 3-fold cross-validation technique ensured that the model wasn't overfitting to a specific split of the data, providing more reliable performance metrics.

The results from the grid search indicated the best set of parameters for each cutoff threshold, such as a max\_depth of 5, a learning\_rate of 0.05, and n\_estimators of 200 for the 50% cutoff. The tuned models were evaluated using accuracy and classification reports, which provided insights into how well the model classified the Low, Medium, and High performance levels. Hyper parameter tuning improved model performance by balancing complexity and generalization, with the learning\_rate and n\_estimators providing a stable approach to prevent overfitting. This process ultimately led to better predictive accuracy and a more reliable model for student performance prediction.

**5. Cross-Validation**

In the past, the model was trained using an XGBClassifier with hyper parameters such as n\_estimators=100, learning\_rate=0.1, and max\_depth=4, which aimed to balance model complexity and prevent overfitting. The 5-fold cross-validation applied during this phase resulted in a mean accuracy of 97.48%, indicating strong model performance. However, during the model refinement phase, changes were made to both the hyper parameters and the evaluation strategy to further optimize the model. While the 5-fold cross-validation was retained, the hyper parameters were fine-tuned using grid search to explore a broader range of options, such as adjusting the learning rate and number of estimators, to improve accuracy. This process led to the identification of more effective hyper parameters, which enhanced model performance.

One of the key differences during refinement was the impact of these adjustments on cross-validation results. After tuning, the cross-validation for the 50% cutoff achieved a mean accuracy of 99%, while the 60% cutoff reached a perfect accuracy of 100%. This change in cross-validation strategy, with better-tuned hyper parameters, showed significantly higher and more stable performance compared to the past results. These improvements in cross-validation accuracy suggest that the model was better able to generalize across different data splits, confirming that the refined model had a higher degree of robustness and consistency. The adjustments to both the hyper parameters and cross-validation strategy reflect a more refined approach to evaluating and optimizing the model, ensuring that the final model would be better suited for the personalized learning experiences needed for the Spectrum-Edu project.

**6. Feature Selection**

**Interaction Feature**

* **Previous Version**: we created an interaction feature (math score \* test preparation completion) but didn’t explicitly include this in the training process or discuss its impact.
* **Refined Version**: ensured that this interaction feature is properly added to the dataset and will be considered by the model during training. It's now used as part of the input features for the classifier.

**Performance Level**

* **Previous Version**: used average\_score for categorizing students into low, medium, or high performance, but this step was somewhat embedded in the workflow.
* **Refined Version**: the creation of performance categories more explicit and systematic by applying a categorization function (categorize) to the average\_score.

**Binning**

* **Previous Version**: binned the math score into categories (Low, Medium, High), but this wasn’t a focus in the evaluation.
* **Refined Version**: included this binning as a feature that will be used for model training, directly impacting how the model learns from the data.

**Test Submission**

**1. Overview**

In the test submission phase, the model was meticulously prepared for final evaluation to ensure accurate performance assessment and readiness for deployment. Initially, the dataset was split into training and testing sets using an 80:20 ratio, with each target variable's split performed separately to maintain consistency across performance thresholds (50% and 60%). Feature scaling was applied exclusively to numerical features, using a StandardScaler fitted on the training data and then applied to the test data to avoid data leakage. The models were trained on the scaled training data and evaluated on the scaled test data, ensuring the test set remained untouched during training and cross-validation. The performance of both models was assessed through accuracy, precision, recall, F1-score, and confusion matrices, demonstrating strong performance with the 60% cutoff model achieving perfect accuracy and the 50% cutoff model performing nearly flawlessly with minimal misclassifications. These evaluations confirmed the models' generalization capabilities, showing no signs of overfitting or underfitting. With consistent results between training, validation, and test data, the models are now prepared for deployment within the Spectrum EDU platform to provide personalized student learning recommendations.

**2. Data Preparation for Testing**

To evaluate model performance, the dataset was split into training and testing sets using an 80:20 ratio for both performance thresholds (50% and 60%). The split was performed separately for each target variable using train\_test\_split with a fixed random\_state=42 to ensure reproducibility.

All features were preprocessed consistently across training and test sets. Specifically, only numerical features were scaled using StandardScaler, with the scaler fitted on the training data and then applied to the test data to prevent data leakage.

The test set was kept completely separate during cross-validation and training, and was used only at the final evaluation stage to ensure a fair assessment of model generalization. No transformations were fit on the test data directly. This approach ensures that the test data accurately reflects unseen student performance scenarios.

**3. Model Application**

The trained models were applied to the test sets by fitting them on scaled training data and predicting the labels of scaled test data. The results were evaluated using accuracy scores, classification reports, and confusion matrices. The 60% cutoff model achieved perfect accuracy, while the 50% cutoff model performed almost perfectly with slight confusion between Low and Medium classes.

After training the XGBoost models on scaled training data for both 50% and 60% performance cutoffs, they were applied to the respective test sets as follows:

1. **Prediction**:

y\_pred\_50 = model\_50.predict(X\_test\_50\_scaled)

y\_pred\_60 = model\_60.predict(X\_test\_60\_scaled)

1. **Evaluation**:
   * **60% Cutoff**:
     + **Accuracy**: 100%
     + **Confusion Matrix**: Perfect classification (no misclassifications).
   * **50% Cutoff**:
     + **Accuracy**: 98%
     + **Confusion Matrix**: Minor confusion between Low and Medium classes.

**4. Test Metrics**

The trained models were evaluated using accuracy, precision, recall, F1-score, and confusion matrices to assess their performance on the test dataset. For the 60% cutoff, the model achieved perfect scores across all metrics—100% accuracy, precision, recall, and F1-score—indicating that it correctly classified every instance in the Low, Medium, and High categories. The confusion matrix confirms this with zero misclassifications. Similarly, the 50% cutoff model performed exceptionally well with a 98% accuracy and strong precision and recall across all classes, although it made a few minor errors (e.g., misclassifying three Low-performing students as Medium and one Medium as Low).

When compared to the training and validation metrics, both models demonstrated consistent and high performance, indicating no signs of overfitting or underfitting. The 60% model maintained perfect accuracy during training and validation as well, while the 50% model achieved nearly identical scores to its test performance. This consistency confirms that both models generalize well to unseen data and are effective in predicting student performance across different cutoff thresholds.

**5. Model Deployment**

Deployment of the trained models has not yet been implemented in a real-world setting. However, future plans include integrating the models into the Spectrum EDU platform to provide personalized learning recommendations and performance insights for students. This integration will involve deploying the models via a cloud-based API or web application, ensuring scalability and accessibility for educators and learners alike.

**6. Code Implementation**

All code for the model refinement and testing phases was developed and executed in Google Colab for ease of collaboration, reproducibility, and scalability. The full code, including detailed comments explaining each step—from data preprocessing and model training to evaluation—is available at the following Google Colab [Link](https://colab.research.google.com/drive/16K0gYmNcokkoIwXMI6eYMB1aSq5fCqJp?usp=sharing).

**Conclusion**

The model refinement and test submission phases resulted in substantial improvements in the performance of the student performance prediction model. Key enhancements included raising the decision cutoff from 50% to 60%, which increased the model’s sensitivity, and performing grid-search hyperparameter tuning to find the best model settings. These efforts led to higher accuracy, precision, and recall, with hyperparameter tuning optimizing parameters like max\_depth, learning\_rate, and n\_estimators. The use of cross-validation helped ensure that the models did not overfit, improving their robustness. The final evaluation metrics showed that the 60% cutoff model achieved perfect accuracy (100%) with flawless classification, while the 50% cutoff model demonstrated 98% accuracy, with minor confusion between Low and Medium performance levels.

The primary challenge faced during the refinement phase was balancing model complexity to avoid overfitting while ensuring accurate predictions. This was addressed through careful hyperparameter tuning and cross-validation, which resulted in a model capable of generalizing well to new data. Both the 60% and 50% cutoff models performed consistently across training, validation, and testing, showcasing their strong generalization abilities. With the models achieving high performance and demonstrating reliability, they are now ready for deployment within the Spectrum EDU platform, where they will provide personalized learning recommendations and performance insights to students.

**References**

**scikit-learn**. (n.d.). Machine learning in Python. Retrieved from [scikit-learn](https://scikit-learn.org/)  
This library was utilized for implementing machine learning algorithms, evaluating model performance metrics (accuracy, precision, recall, F1-score), and conducting cross-validation.

**XGBoost**. (n.d.). XGBoost: A scalable tree boosting system. Retrieved from [xgboost](https://xgboost.readthedocs.io/en/latest/)  
The XGBClassifier model and hyperparameter tuning through grid search were performed using this library.

**NumPy**. (n.d.). NumPy: The fundamental package for scientific computing with Python. Retrieved from [numpy](https://numpy.org/)  
This library was employed for numerical array handling and matrix operations throughout the data preprocessing and modeling phases.

**Pandas**. (n.d.). Pandas: A fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation library for Python. Retrieved from [pandas](https://pandas.pydata.org/)  
Used for data manipulation and preprocessing tasks, including handling missing values and splitting the dataset.

**Matplotlib & Seaborn**. (n.d.). Matplotlib: Visualization with Python. retrieved from [matplotlib](https://matplotlib.org/)  
Seaborn: Statistical data visualization. Retrieved from [seaborn](https://seaborn.pydata.org/)  
These libraries were used for visualizing model performance, including confusion matrices, ROC curves, and precision-recall curves.

**Google Colab**. (n.d.). Google Colaboratory: A research tool for machine learning education and research. Retrieved from [goggle colab](https://colab.research.google.com/)

Used for cloud-based collaborative development and running the model implementation, training, and evaluation processes.

**GeeksforGeeks**. (n.d.). Hyperparameter Tuning with Grid Search. Retrieved from [geeksforgeeks](https://www.geeksforgeeks.org/grid-search-hyperparameter-tuning/)

This resource provided valuable insights into hyperparameter tuning and grid search methodology for optimizing model performance.

**Chen, T., & Guestrin, C.** (2016). XGBoost: A scalable tree boosting system. Retrieved from [link](https://arxiv.org/abs/1603.02754)  
This seminal paper on XGBoost provided an in-depth explanation of the algorithm and its application for machine learning tasks, forming the foundation for the model refinement process.

**Bergstra, J., & Bengio, Y.** (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13, 281-305. Retrieved from [link](https://arxiv.org/abs/1201.0490)  
This paper discusses the advantages of random search and grid search techniques for hyperparameter optimization, which informed the grid search approach used in this project.