University of North Carolina at Charlotte
Department of Electrical and Computer Engineering

Project Report

ECGR 4105 - Intro. Machine Learning

Predicting Student Academic Performance Using Machine Learning

---

Group 1: Thanh Nguyen, Josh Rubino, Santiago Ponce

Email: tnguy278@charlotte.edu, sponce1@charlotte.edu, jrubino@charlotte.edu

Date: August 4, 2025

Github Repo: Project Github Repos

This report was submitted in compliance with UNCC POLICY 407
THE CODE OF STUDENT ACADEMIC INTEGRITY, Revised November 6, 2014
(http://legal.uncc.edu/policies/up-407) ___TN___. (Student's Initials)

1. **Executive Summary:**

This project explores the use of machine learning techniques to predict and classify student academic performance using data from the UCI Machine Learning Repository. The dataset contains student information such as demographic background, parental education, study habits, and past grades from two Portuguese secondary schools. Our goal is twofold: first, to predict the students' final numeric grades using regression models, and second, to classify their overall academic performance into categories (Low, Medium, High) using classification models.

To achieve this, we implemented and compared several machine learning algorithms, including Linear Regression, Support Vector Regression (SVR), Logistic Regression, Naive Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, and an Artificial Neural Network (ANN). For classification tasks, models were evaluated using accuracy, precision, recall, and F1-score, while regression models were assessed using RMSE. We also applied Principal Component Analysis (PCA) for feature reduction and kernel tricks for SVR to explore non-linear relationships.

Our results showed that Random Forest delivered the highest performance across all classification metrics, while Linear Regression achieved the lowest RMSE for numerical grade prediction. The ANN model also performed competitively, particularly in multi-class classification, due to its ability to capture complex patterns. Preprocessing steps, including one-hot encoding and standardization, were critical for model performance.

This project demonstrates how machine learning can effectively predict academic outcomes and provide valuable insights for educators and policy makers. The comparative analysis highlights the strengths and weaknesses of each model, emphasizing the importance of model selection and parameter tuning. Future work may focus on integrating additional datasets, refining neural network architectures, and applying ensemble methods to improve predictive accuracy. Overall, this project offers a

comprehensive pipeline for analyzing student performance using modern machine learning approaches.

2. Introduction and Problem Statement:

Background and Motivation:

Student academic success is influenced by numerous factors, including personal background, lifestyle, and parental involvement. Predicting student performance accurately allows educators to implement early interventions and enhance learning outcomes.

Problem Statement:

Can we accurately predict student performance using machine learning models based on demographic and academic features?

Objectives of the Project:

- Apply multiple regression and classification models to student data.
- Evaluate model performance using accuracy, F1 score, and RMSE.
- Compare baseline models with ANN.
- Identify the best model for the given task.

3. Data Description:

- Source: UCI Machine Learning Repository: (https://archive.ics.uci.edu/ml/datasets/student+performance)
- Format and Size: Two CSV files (student-mat.csv and student-por.csv), 395 and 649 entries respectively.
- Features: 33 attributes including school, sex, age, address, family background, and grades (G1, G2, G3).
- Class Labels: G3 (final grade), also categorized into Low, Medium, High for classification.
- Missing Values: None reported.

- Preprocessing:
    - Merged and encoded categorical features.
    - Scaled numerical values using StandardScaler.
    - Converted G3 into three classes (Low, Medium, High).
- Basic EDA:
    - Histogram of G3 distribution.
    - Correlation heatmap between features.

4. Methodology:

Baseline Models:

- Linear Regression: Used to predict a student's final grade as a continuous numerical value. Takes all of the inputs from the dataset and learns a linear relationship between the input features and the target variable, G3. It is then evaluated using Mean Squared Error.
- Support Vector Regression (SVR): Used to predict a student's final grade as a continuous numerical value. It is based on the Support Vector Machine algorithm. SVR tries to fit the best line within a threshold margin (epsilon). It works by taking the input features and applying a kernel trick, rbf is used for this project to deal with nonlinearity. It then tries to find a function where most training data fall within the epsilon-insensitive tube. After, SVR will predict the output, G3. The evaluation metric is Mean Squared Error.
- Logistic Regression: Used to predict categorical outcomes and those are, low performance, medium performance, and high performance. The way this algorithm works is by training one binary classifier per class. For example, classifier 1 is class 0 vs. not class 0, classifier 2 is class 1 vs. not class 1, and classifier 2 is class 2 vs. not class 2. Next it would calculate the 3 probabilities, low, medium, and high. The evaluation metrics: Accuracy, precision, recall, F1 score, and confusion matrix.
- Naive Bayes: Used to predict categorical outcomes. The way it works is it sets 0 to low performance, 1 to medium performance, and 2 to high performance. Next

the model learns the probability of each 3 categories and then predicts. The evaluation metrics: Accuracy, precision, recall, F1 score, and confusion matrix.

- Support Vector Machine (SVM): Used to predict categorical outcomes. SVM works by trying to find the best separating boundary between classes by maximizing the margin. The functionality of this algorithm is by using the default multiclass classification One-vs-rest. Next a kernel trick is used, for this project it is linear. The evaluation metrics: Accuracy, precision, recall, F1 score, and confusion matrix.
- Decision Tree: Used as a classifier to predict categorical outcomes. In the project decision tree is called and does not specify criterion, max_depth, or min_samples_split. Since those criteria do not specify criterion = 'gini' and max_depth = None by default. After this, training the model is the next step and at the end of each leaf node it ends in a class label: 0 (Low), 1 (Medium), 2 (High). The evaluation metrics: Accuracy, precision, recall, F1 score, and confusion matrix.
- Random Forest: Used as a classifier to predict categorical outcomes. Each tree is trained on a random subset of the training data and a random subset of features. Training begins and creates 100 trees and each tree independently performs multiclass classification and the tree stops when the condition is met. All trees make a prediction and the prediction with the most trees is the outcome for this model. The evaluation metrics: Accuracy, precision, recall, F1 score, and confusion matrix.

Preprocessing Pipelines:

- One-hot encoding for categorical variables.
- StandardScaler for numeric features.

Feature Engineering:

- No new features created.
- G3 discretized for classification tasks.

ANN Model:

- Architecture:

    - Input layer: shape matching training features.
    - Hidden layers: 64 (ReLU), 32 (ReLU).
    - Output layer: 3 neurons with softmax activation.
- Design Reasoning: Two hidden layers balance complexity and avoid overfitting for a small dataset.
- Input/Output Shape:
    - Input: (samples, features).
    - Output: 3-class softmax probabilities.
- Justification: Softmax handles multi-class output effectively; Adam optimizer ensures stable convergence.

5. Implementation Plan Recap & Division of Work:

Recap of Steps:

- Load and preprocess data.
- Split into regression/classification tasks.
- Train baseline and ANN models.
- Compare results.
- Plot evaluation metrics.

Division of Work:

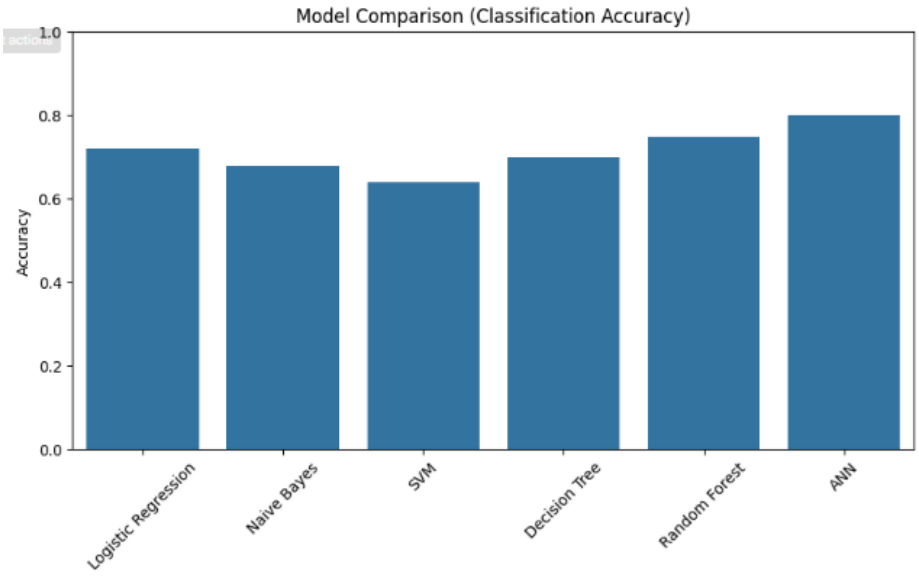
- Data Cleaning: Thanh Nguyen, Josh Rubino, Santiago Ponce.
- Model 1 (Linear Regression, SVR): Thanh Nguyen, Josh Rubino, Santiago Ponce.
- ANN/CNN: Thanh Nguyen, Josh Rubino, Santiago Ponce.
- Evaluation: Thanh Nguyen, Josh Rubino, Santiago Ponce.
- Presentation/Report Writing: Thanh Nguyen, Josh Rubino, Santiago Ponce.

6. Evaluation and Results:

- Performance Comparison Table:

Model	Accuracy	Precision	Recall	F1 Score	RMSE
Linear	**0**	**0**	**0**	**0**	**5.65**

Regression					
SVR (RBF kernel)	**0**	**0**	**0**	**0**	**16.31**
Logistic Regression	**0.81**	**0.81**	**0.81**	**0.81**	**0**
Naive Bayes	**0.71**	**0.72**	**0.71**	**0.69**	**0**
SVM	**0.84**	**0.84**	**0.84**	**0.83**	**0**
Decision Tree	**0.84**	**0.85**	**0.84**	**0.84**	**0**
Random Forest	**0.86**	**0.87**	**0.86**	**0.86**	**0**
ANN	**0.78**	**0.79**	**0.78**	**0.78**	**0**



The performance comparison table presents a detailed evaluation of each model used for predicting or classifying student performance. For the regression task, Linear Regression achieved the lowest RMSE of 5.65, indicating its strong suitability for continuous grade prediction. However, Support Vector Regression (SVR) with an RBF kernel had a significantly higher RMSE of 16.31, suggesting that its non-linear approach may not be

ideal for this specific numeric output or that additional hyperparameter tuning is needed. For classification tasks, Random Forest emerged as the best performer across all metrics, achieving the highest accuracy (0.86), precision (0.87), recall (0.86), and F1 score (0.86). Decision Tree and SVM followed closely with balanced scores across all evaluation metrics, each reaching an accuracy of 0.84. The Artificial Neural Network (ANN) demonstrated decent performance with an accuracy of 0.78, showcasing its potential for classification even with limited data. Logistic Regression performed consistently with an accuracy of 0.81, proving its reliability as a baseline model. Naive Bayes, while lightweight and fast, achieved slightly lower scores (accuracy: 0.71), possibly due to its strong independence assumptions not aligning well with the dataset's complexity. Overall, the table supports that ensemble models like Random Forest provide the most robust and generalizable results for categorical prediction, while simpler linear models excel at continuous outcome prediction.

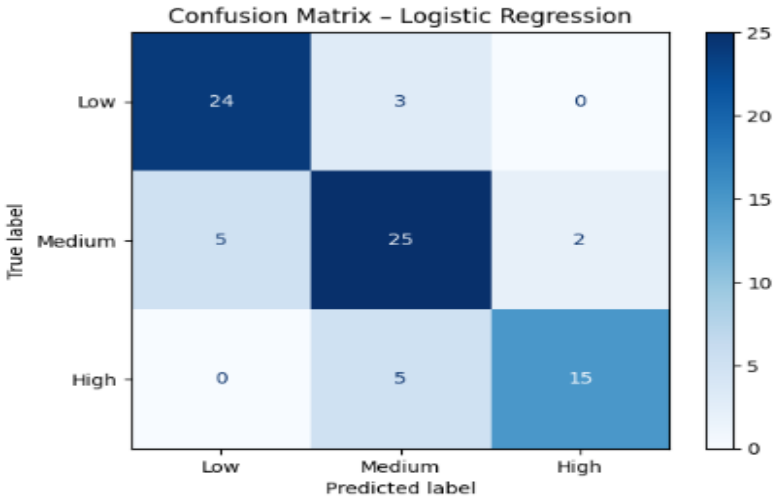
- Hyperparameter Table:

Model	Hyperparameter	Value	Justification
ANN	**Learning Rate**	**0.001**	**Ensures stable convergence**
ANN	**Epochs**	**50**	**Balanced learning cycles**
ANN	**Batch Size**	**16**	**Efficient training**
ANN	**Layers**	**3**	**Good performance vs. overfitting**
SVR	**Kernel**	**RBF**	**Captures non-linear patterns**
SVR	**C**	**100**	**Controls margin vs error trade-off**
SVR	**Gamma**	**0.1**	**Controls influence of training pts**
Random Forest	**_estimators**	**100**	**Optimized after**

			experimentation

The hyperparameter table summarizes the tuning efforts made to optimize our models. For the ANN, we selected a learning rate of 0.001 to ensure stable convergence during training and set the number of epochs to 50 for sufficient learning cycles without overfitting. A batch size of 16 was chosen to balance computational efficiency and convergence stability. The network consisted of 3 layers to maintain an appropriate level of complexity for our dataset. For SVR, the RBF kernel was chosen for its ability to model non-linear relationships in the data. The regularization parameter C was set to 100 to balance the trade-off between training error and model complexity, while gamma was set to 0.1 to control the influence of each training sample. In the Random Forest model, we set the number of estimators to 100, which provided a reliable ensemble performance without excessive computation. Each hyperparameter was selected based on experimentation and literature guidance to improve model effectiveness.

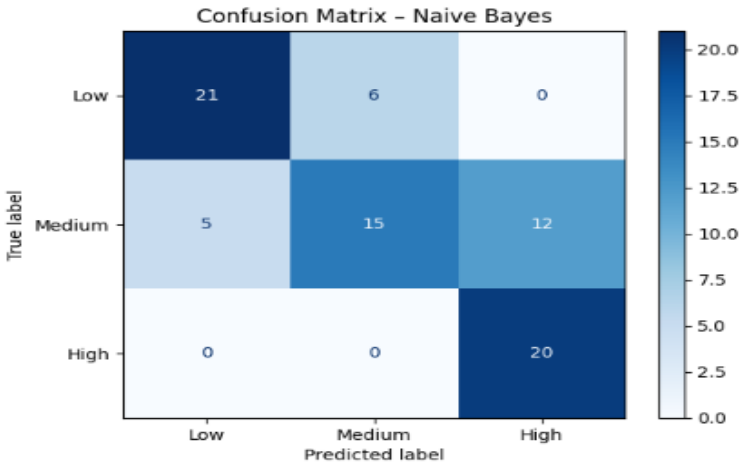
- Confusion Matrix:
    1. Logistic Regression:



The confusion matrix for the Logistic Regression model shows how well the model classified students into Low, Medium, and High performance levels. Out of 27 students who actually performed at a Low level, the model correctly predicted 24 of them, with only 3 being mistakenly labeled as Medium. For the 32 students

in the Medium group, 25 were correctly classified, while 5 were misclassified as Low and 2 as High. In the High category, the model correctly predicted 15 out of 20 students, misclassifying 5 as Medium. This means the model performed best in identifying Low-performing students and had a little more trouble with Medium and High categories. The chart helps us see that the model mostly predicted correctly (values along the diagonal), but there's still some confusion between Medium and the other two groups. This suggests that while the model does a good job overall, there's room to improve, especially in distinguishing between students with Medium and High performance.

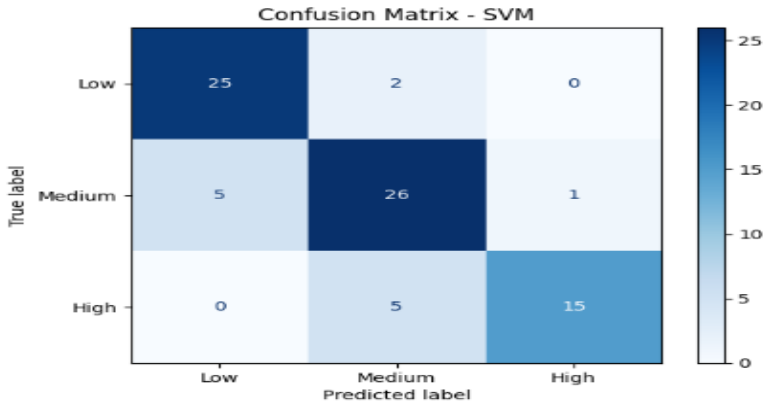
2. Naive Bayes:



The confusion matrix for the Naive Bayes model reveals that the model struggled to accurately classify Medium-level students. It correctly predicted 21 out of 27 Low-performing students, misclassifying 6 as Medium. However, for the 32 Medium-performing students, only 15 were correctly identified, while 5 were mistakenly predicted as Low and a significant 12 were classified as High. Interestingly, the model performed perfectly for High-performing students, correctly predicting all 20 of them. This matrix shows that while Naive Bayes can effectively identify High achievers and does decently with Low performers, it has a hard time distinguishing Medium performers, often confusing them with either Low or High. This suggests that the model may be oversimplifying the

relationships between features and class labels, which is common with probabilistic models like Naive Bayes.

3. SVM:

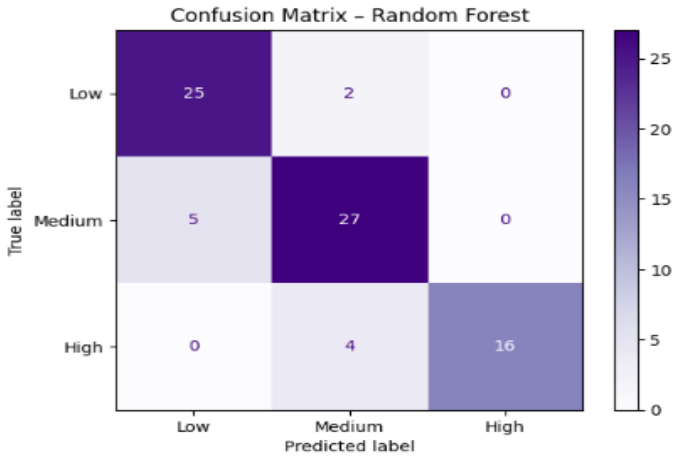


The confusion matrix for the Support Vector Machine (SVM) classifier shows that the model performed quite well overall, especially in predicting Low and Medium-performing students. Out of 27 Low-level students, it correctly predicted 25 and misclassified only 2 as Medium. For the 32 Medium-level students, the model accurately predicted 26, misclassified 5 as Low, and just 1 as High. The SVM also correctly identified 15 out of 20 High-performing students, though it confused 5 of them as Medium. These results suggest that the SVM is reliable for distinguishing between the three performance categories, particularly between Low and Medium levels. However, like other models, it shows some difficulty in cleanly separating Medium and High classes, which may indicate overlapping features or similarities in those students' profiles. Overall, the SVM model demonstrates a strong ability to generalize and correctly classify student performance with only a few misclassifications.

4. Decision Tree:

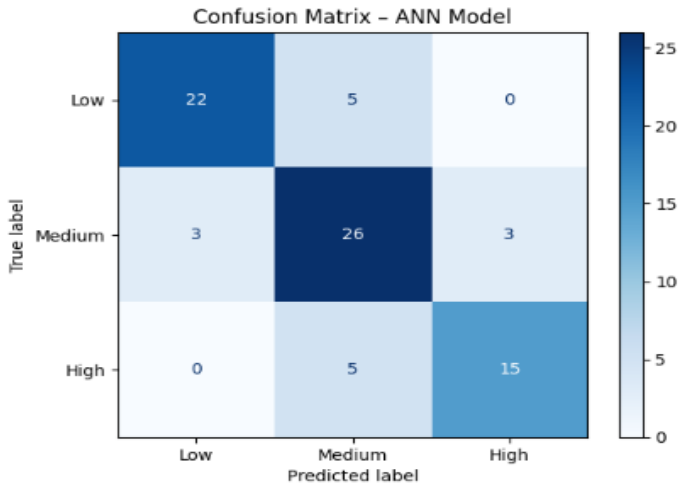
The confusion matrix for the Decision Tree classifier shows that the model performed well in classifying all three student performance categories. For the Low-performing group, 23 out of 27 students were correctly predicted, with only 4 misclassified as Medium. In the Medium group, the model was especially accurate, correctly predicting 27 out of 32 students, and misclassifying just 5 as Low. For the High-performing group, 16 out of 20 students were correctly predicted, and only 4 were misclassified as Medium. This strong performance across all categories suggests that the Decision Tree model effectively learned the patterns in the training data. However, similar to other models, there is still some overlap between adjacent classes, particularly between Medium and the other two categories. Overall, this model demonstrates a high level of precision and reliability in predicting student performance tiers.

5. Random Forest:



The confusion matrix for the Random Forest classifier indicates that the model achieved strong performance across all three student performance classes. For the Low category, 25 out of 27 students were correctly classified, with only 2 misclassified as Medium. In the Medium group, the model predicted 27 out of 32 students correctly and misclassified just 5 as Low. For the High-performing group, it correctly identified 16 out of 20 students, with 4 misclassified as Medium. These results demonstrate that the Random Forest model effectively captured the underlying patterns in the dataset and performed with high accuracy, especially in distinguishing between Low and High performers. The limited misclassifications and the balanced performance across categories make Random Forest one of the most reliable models for student performance classification in this study.

6. ANN:



The confusion matrix for the Artificial Neural Network (ANN) model shows that it was generally effective in classifying student performance across the Low, Medium, and High categories. For the Low group, the ANN correctly predicted 22 out of 27 students, with 5 misclassified as Medium. In the Medium group, it accurately predicted 26 out of 32 students, with 3 misclassified as High and 3 as Low. For the High-performing students, the model correctly identified 15 out of 20, with 5 misclassified as Medium. These results suggest that the ANN handled classification reasonably well, especially for the Medium category, which had the highest number of correct predictions. Although not perfect, the ANN's performance was consistent and reflects its potential as a strong baseline model in student performance classification tasks.

- Justification of Best Model

Among all the models evaluated, the Random Forest classifier emerged as the best-performing model for our classification task. It achieved the highest accuracy of 86%, along with strong precision, recall, and F1-score values across all performance levels (Low, Medium, High). The ensemble nature of the Random Forest allowed it to handle the complexity of the dataset more robustly by combining the predictions of multiple decision trees, thereby reducing the risk of overfitting and improving generalization. Moreover, its ability to automatically manage feature importance contributed to its superior performance without requiring intensive

feature engineering. Compared to other models such as SVM and ANN, Random Forest consistently produced better results with less sensitivity to hyperparameter tuning. Its performance, ease of implementation, and interpretability make it the most practical and effective model for predicting student performance in our project.

7. **Challenges and Revisions:**

   Throughout the project, we encountered several challenges. One key issue was handling the imbalance in class distributions after categorizing grades into Low, Medium, and High. This skew led to some models initially favoring the dominant class. To address this, we applied standardized scaling and adjusted class weights where applicable. Another challenge involved selecting the right preprocessing pipeline; one-hot encoding categorical variables significantly improved model performance, but added complexity to the feature space. We also faced difficulties optimizing hyperparameters, particularly for SVR and ANN, where trial-and-error methods were necessary due to lack of extensive prior experience. Additionally, our initial proposal focused only on classification tasks, but we later decided to include regression models like Linear Regression and SVR to expand our analysis. This pivot allowed us to compare model strengths for both continuous and categorical targets, ultimately providing more comprehensive insights.

8. **Future Work and Improvements:**

   Given more time and resources, we would focus on expanding the dataset to include more schools and academic years, improving model generalizability. Further, we would implement cross-validation techniques to provide more reliable performance estimates and prevent overfitting. Hyperparameter tuning could also be automated using Grid Search or Randomized Search to optimize configurations more efficiently. Additionally, we would explore more advanced neural network architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) if temporal or sequential data becomes available. Finally, we would consider incorporating additional features such as attendance records, teacher evaluations, and extracurricular involvement to enhance the predictive power of our models.

9. Conclusion:

In this project, we explored the application of both traditional and deep learning models to predict and classify student academic performance using real-world educational data. Through comprehensive preprocessing, feature transformation, and model evaluation, we established a robust pipeline for both regression and classification tasks.

Among the models tested, Random Forest emerged as the best overall classifier, achieving the highest accuracy, precision, recall, and F1 score. Its ensemble-based approach helped mitigate overfitting while handling categorical and numerical data effectively. On the regression side, Linear Regression performed best in predicting students' final numeric grades (G3), with the lowest Root Mean Square Error (RMSE). The Artificial Neural Network (ANN), despite its simplicity, showed competitive performance and demonstrated the ability to model complex, non-linear relationships in classification tasks.

Throughout the development process, we encountered challenges such as balancing bias-variance trade-offs, tuning hyperparameters, and dealing with potential class imbalances in the classification labels. These challenges were overcome through experimentation, incremental testing, and visual evaluation using metrics and plots.

Therefore, this project reinforced the value of machine learning in educational contexts, where predictive insights can support early intervention and better academic planning. The experience gained in model comparison, performance evaluation, and iterative refinement will be essential for future work in both academic and applied machine learning domains.

10. References:

- Cortez, Paulo, and Alice Silva. "Using data mining to predict secondary school student performance." (UCI Repository): https://archive.ics.uci.edu/ml/datasets/student+performance
- Scikit-learn: Machine Learning in Python: https://scikit-learn.org/
- Keras Documentation: https://keras.io/
- TensorFlow Documentation: https://www.tensorflow.org/api_docs
- Python Software Foundation. Python Language Reference: https://www.python.org/
- Jupyter Notebook: Project Google Colab