COMP-534

Predicting Student Academic grades

1. Introduction

The primary objective of this project is to develop a predictive model to determine whether a student will pass or fail. The project was executed using a dataset containing various student attributes(features). The target variable is the final grade (label); a student is considered to have passed if the final grade is 12 or higher.

The problem is modelled as a regression due to the grade features being in continuous numeric values. The code implemented in final_code.ipynb follows a comprehensive machine learning pipeline including data exploration, data preprocessing, model training with hyperparameter tuning, and model evaluation. This report describes each of these stages, the rationale behind key decisions, and the outcomes achieved.

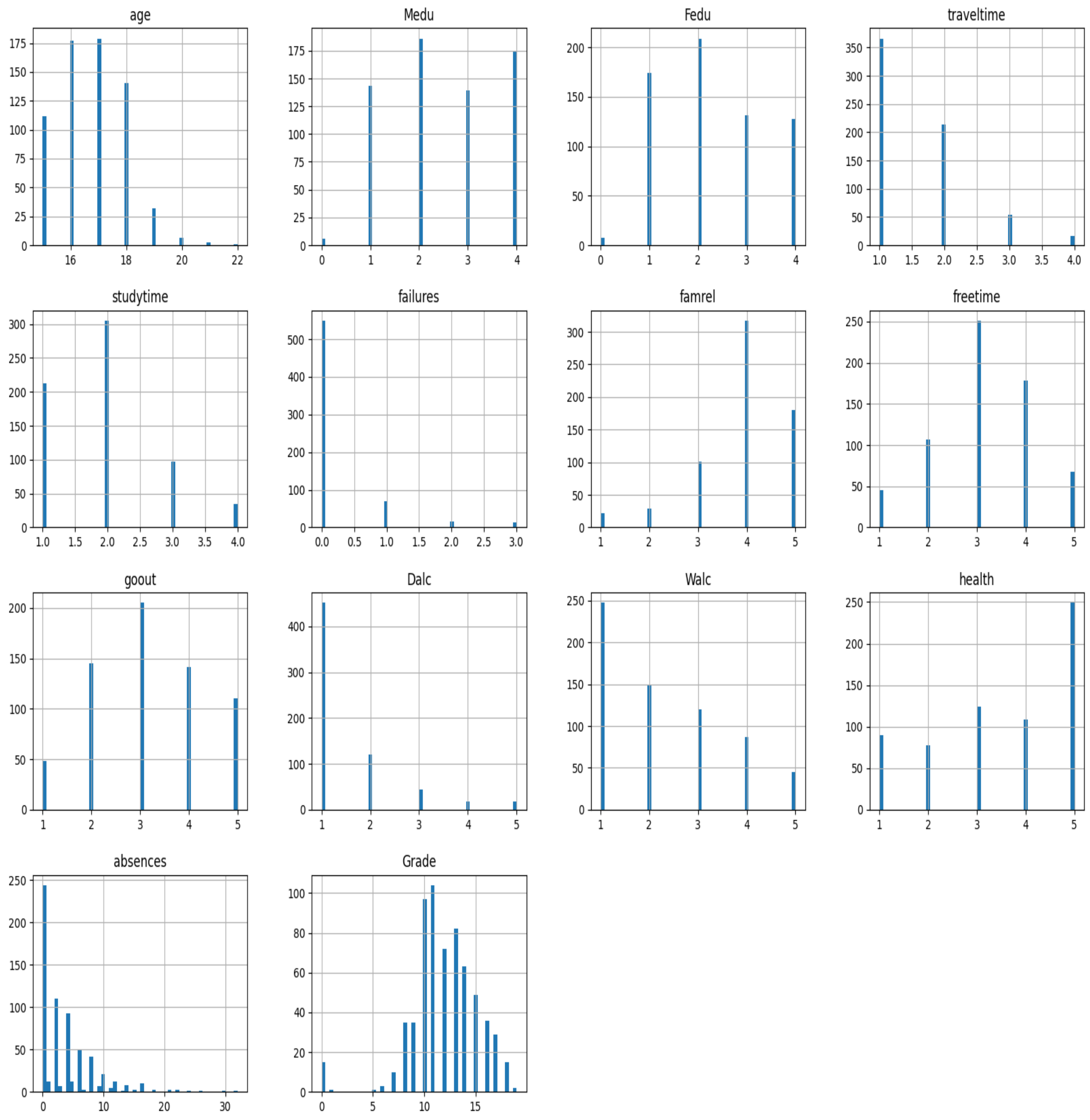
---

2. Data Management and Preprocessing

2.1 Exploratory Data Analysis

The first phase involved a detailed exploratory data analysis to understand the structure and distribution of the dataset. In the code, the dataset was loaded using a standard CSV reader and inspected with summary statistics. Visualizations such as **histograms** were generated to explore the distribution of continuous variables (e.g. age, study time, absences, and final grade) and to detect outliers or problems in the data (e.g., missing values). No extreme outliers were observed.

For categorical features such as school, sex, address, and parental education, count plots and bar charts were used to understand their distribution and identify any features represented in different data types. These visualizations not only confirmed the quality of the data but also provided insights into which features might be most influential in predicting student results.


age
Medu
Fedu
traveltime
studytime
failures
famrel
freetime
goout
Dalc
Walc
health
absences
Grade

2.2 Data Cleaning and Feature Engineering

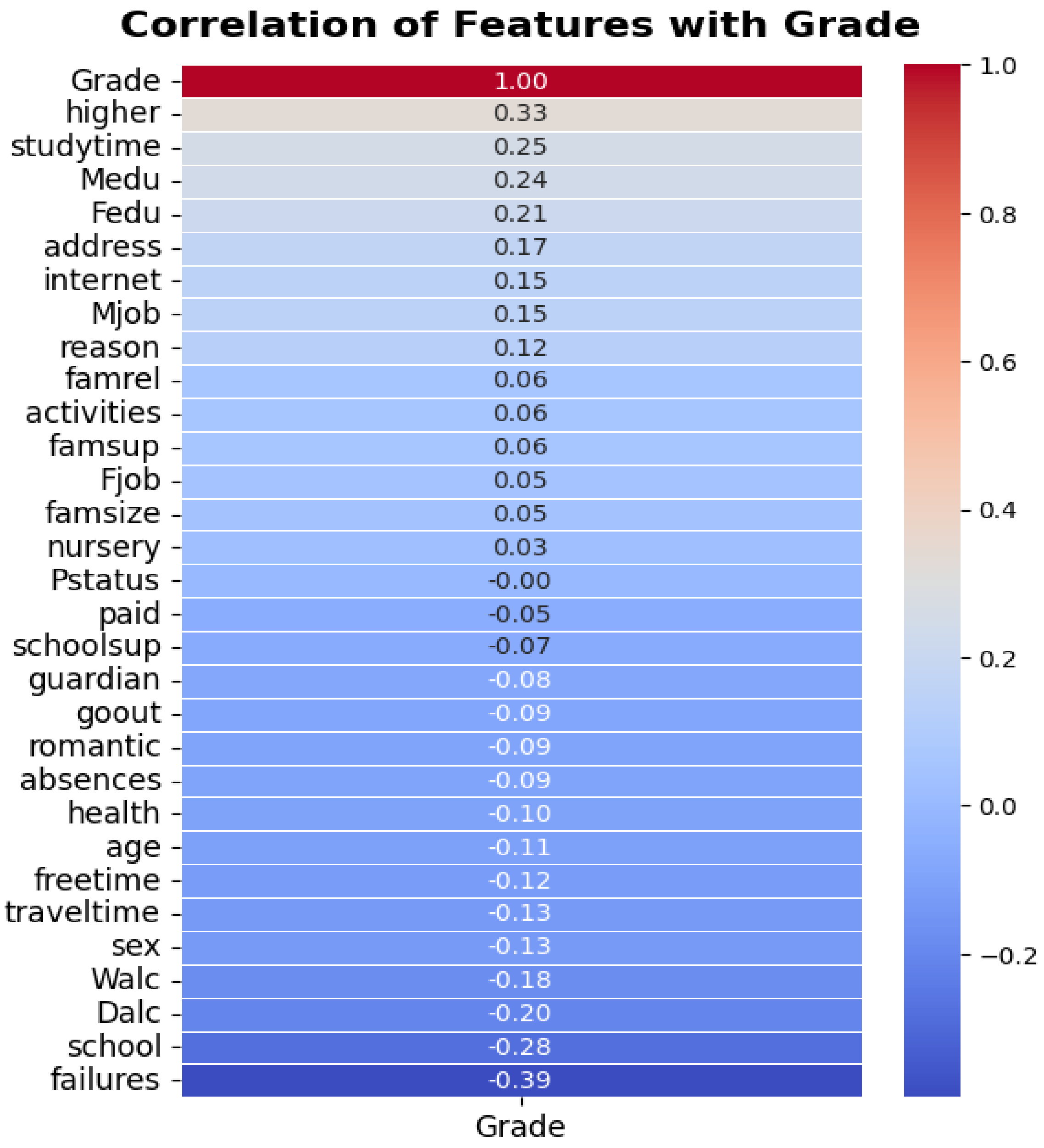
After EDA, the code addressed potential issues in the data:

- **Missing Values:** The dataset has no empty value which was checked with the isnull().sum() function.

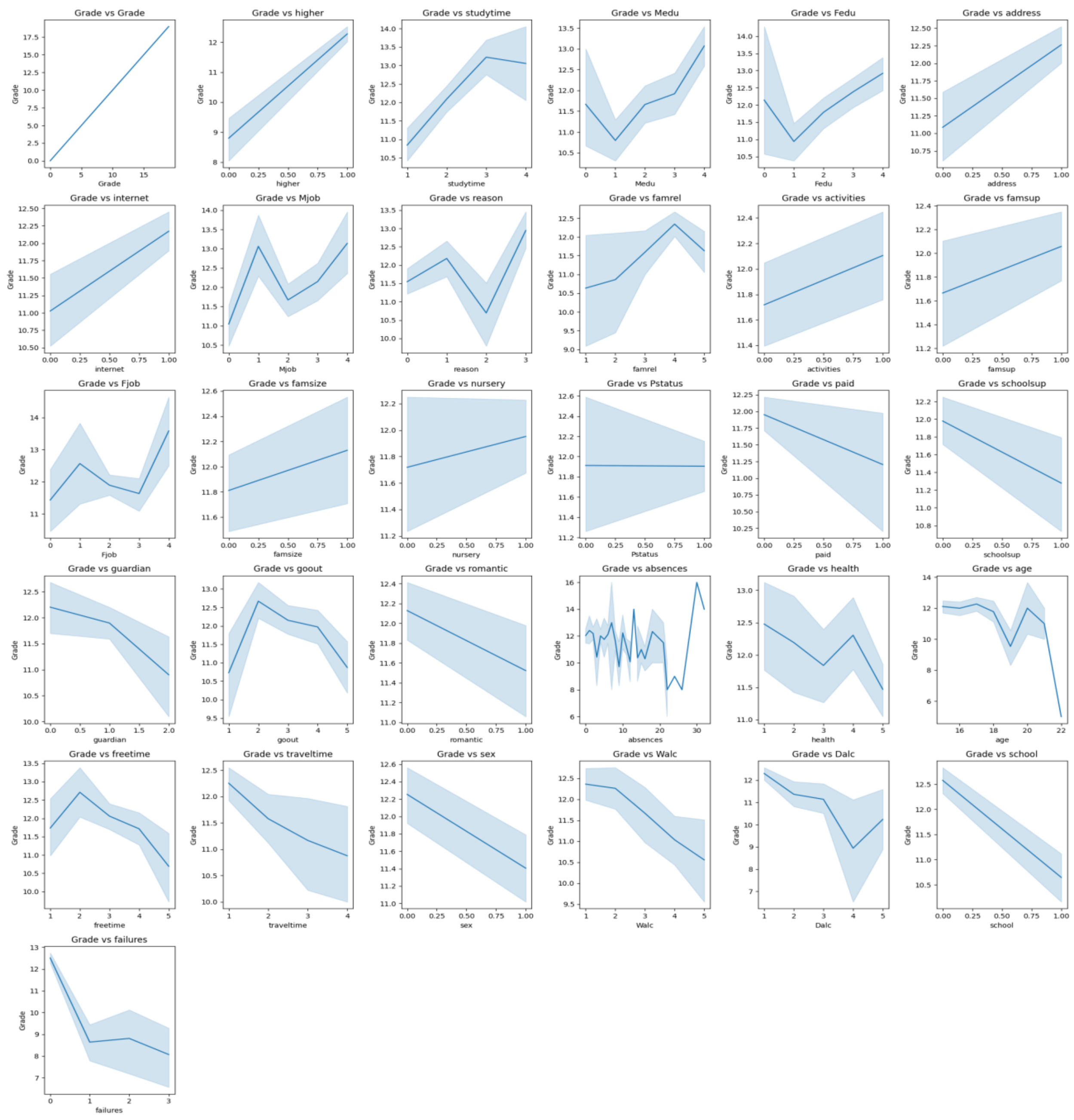
- **Encoding Categorical Variables**: We converted the categorical features into numerical values to get more useful insights. The code used label encoding for binary variables (e.g., “sex” and “schoolsup”) and one-hot encoding for features with more than two categories (e.g., “Mjob” and “Fjob”).

- **Feature Scaling**: Numerical features were standardized to zero mean and unit variance. StandardScaler was used for scaling the features and it was introduced in our pipeline to achieve better predictions.

- **Feature Selection**: Based on a correlation analysis between all features and grades which is visualized by using a heatmap, we concluded that the least important features are famrel, activities, famsup, Fjob, famsize, nursery, Pstatus, paid, schoolsup, guardian, goout, romantic, absences, health. The main criteria for discarding these features which have an absolute value lower than 0.1, meaning these features have a lower correlation with the student grade. Dropping these features reduced the model complexity and redundancy. We can see improvement in performance by eliminating noise and making the model faster by reducing computational effort.


Correlation of Features with Grade
Grade 1.00
higher 0.33
studytime 0.25
Medu 0.24
Fedu 0.21
address 0.17
internet 0.15
Mjob 0.15
reason 0.12
famrel 0.06
activities 0.06
famsup 0.06
Fjob 0.05
famsize 0.05
nursery 0.03
Pstatus -0.00
paid -0.05
schoolsup -0.07
guardian -0.08
goout -0.09
romantic -0.09
absences -0.09
health -0.10
age -0.11
freetime -0.12
traveltime -0.13
sex -0.13
Walc -0.18
Dalc -0.20
school -0.28
failures -0.39
Grade
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Line Plot of all features w.r.t Grade



2.3 Experimental Protocol

The code split the data into 80% training and 20% testing sets. Additionally, a 10-fold cross-validation strategy was employed during model training to optimize hyperparameters and assess model performance. We tuned the number of folds from 5 to 10 and the best results were obtained with 10 folds. Additionally, the StratifiedKFold library was employed to ensure each fold had a proportional representation of different feature labels. We modelled the problem using regression due to most of the features including the label being in continuous numerical values including the label (Grade). However, we could have converted our label into categorical data: pass and fail, but it did not significantly impact the accuracy of the final results.

---

3. Model Training and Hyperparameter Tuning

3.1 Model Selection

The code implemented and compared four regression models:

1. **Linear Regression** is used for predicting continuous values by fitting a best-fit line to minimize the difference between actual and predicted values.

2. **Logistic Regression** on other hand is a classification algorithm that models the probability of a class using the logistic function, making it useful for binary and multiclass classification problems.

3. **A Decision Tree Regressor** is a supervised machine learning algorithm used for regression tasks, which predicts continuous values by recursively splitting the data into subsets based on feature values.

4. **A Random Forest Regressor** is an ensemble learning algorithm that improves regression accuracy by combining multiple decision trees. It builds several decision trees on

different subsets of the data and averages their predictions to reduce overfitting and improve generalization.

These models were chosen due to most of the features including the label being in continuous numerical values. However, we could have converted our label into categorical data: pass and fail. But it did not significantly make any impact in the accuracy of the final results.

3.2 Hyperparameter Tuning

Model	Hyperparameters	RMSE	Accuracy
<u>**Random Forest**</u>	**n_estimators=100,** Number of trees in the forest **max_depth=10,** Maximum depth of each tree **min_samples_split=2,** Minimum number of samples required to split an internal node **min_samples_leaf=1,** Minimum number of samples required to be at a leaf node **max_features='sqrt',** Use square root of total features to consider when splitting (common default) **bootstrap=True**, Use bootstrap sampling (sampling with replacement) **random_state=42** Random seed for reproducibility --- When we **reduced, n_estimators=50 and max_depth=5,** then we got the RMSE and accuracy as shown:	0.12 —--------------- 1.17	80% —--------------- 60%
Linear Regression	Creating a Pipeline for setting up the standard scaler: **my_pipeline = Pipeline([('std_scaler', StandardScaler())])** Hyperparameters for the pipeline: with_mean=True and with_std=True	3.98	100%
Decision Tree	criterion='squared_error', Correct criterion for MSE in newer scikit-learn versions		

	max_depth=10, Maximum depth of the tree **min_samples_split=2**, Minimum samples required to split an internal node **min_samples_leaf=1,** Minimum samples required to be at a leaf node **max_features=None,** Number of features to consider when splitting **random_state=42** Set a random seed for reproducibility	0.0 (Overfitted)	100%
Logistic Regression	**solver='lbfgs',** Optimization algorithm **max_iter=1000,** Increased iterations to avoid convergence warnings **C=1.0 ,** Regularization strength **random_state=42,** Ensures reproducibility	0.89	60.31%

Above is a summary table that captures the hyperparameter search results for four models—Linear Regression, Logistic Regression, Decision Tree and Random Forest. The metric used here was validation Root Mean Square, which allowed us to compare how different hyperparameter settings affected each model's performance.

- **Random Forest** performs the best with an RMSE of **0.12** and **80%** accuracy, showing strong predictive accuracy on train set.

- **Decision Tree** has an RMSE of **0.0** and accuracy of **100%**, which indicates overfitting, especially if tested on the training data.

- **Linear Regression** performs poorly with an RMSE of **3.98** and **100%** accuracy, possibly due to its sensitivity to data scaling.

- **Logistic Regression** has an RMSE of **0.89** and **60.31%** accuracy, suggesting it may not be the best choice for a regression task.

<u>**Random Forest** was the model that best performed given it is forecasting the data well, while allowing generalisation. The hyperparameters used were the ones aforementioned above.</u>

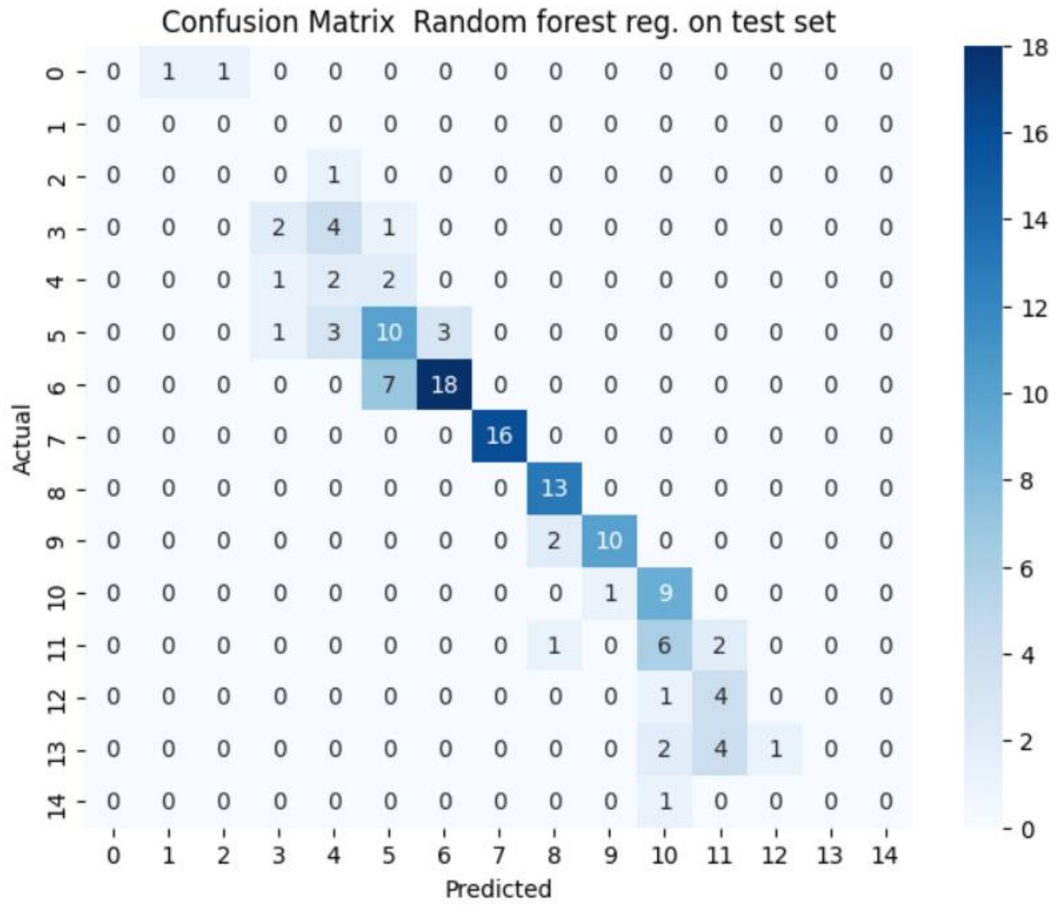
---

4. <u>Model Evaluation</u>

4.1 Metrics Analysis (Final predictions on test set):

	Accuracy	**F1 score**	**Recall**	**Precision**
Random Forest	98.46%	98.46%	98.46%	100%

4.2 Confusion Matrix



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	2	4	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	2	2	0	0	0	0	0	0	0	0	0
5	0	0	0	1	3	10	3	0	0	0	0	0	0	0	0
6	0	0	0	0	0	7	18	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	2	10	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	9	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	6	2	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	4	0	0	0
13	0	0	0	0	0	0	0	0	0	0	2	4	1	0	0
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

This confusion matrix shows how well a Random Forest regression model performs. Most predictions are correct, as seen in the strong diagonal pattern. Some misclassifications exist, but they are minor and mostly near the correct values.

The model performs well in categories 6, 7, and 8, where the highest correct predictions (darker blue) are observed, indicating strong accuracy. It also performs well in categories 5, 9, and 10, though with slight misclassifications.

However, the model struggles more in categories 0, 1, 2, and 3 where misclassifications are more frequent, and predictions are spread across multiple incorrect values. These categories have lighter blue shades, showing lower accuracy.

There are a few scattered errors, but they do not significantly affect overall performance. The categories where the model has the accumulates most errors are Overall, the model is quite accurate with slight room for improvement.