Report on ML Models on Student Data

Understanding Factors Influencing Student Performance and Predict Student Grades for Early Intervention

Our aim is to understand the factors influencing student performance. The goal is to identify patterns, correlations, and insights that can inform predictive modeling for final grades.
This report outlines the steps taken, from Exploratory Data Analysis (EDA) to model selection, tuning, and evaluation. Additionally, we will find insights into feature engineering, database setup, and a Python function for adding new students.

.

Dataset Overview:

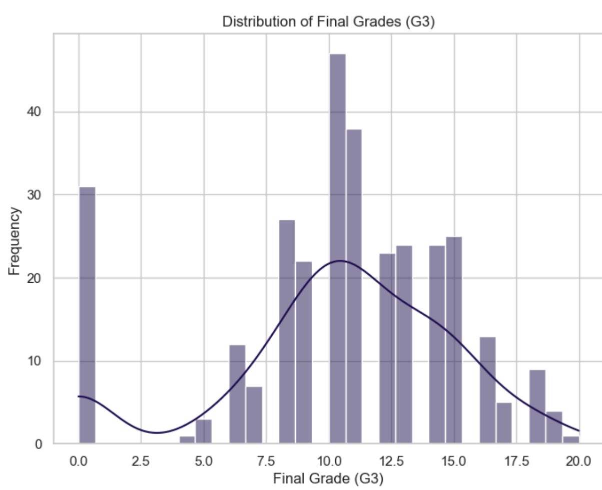
No of records in the dataset: 316
No of student record columns: 33
Student Demography: 30
Student grades G1,G2,G3: 3
Grade Range : 0-20

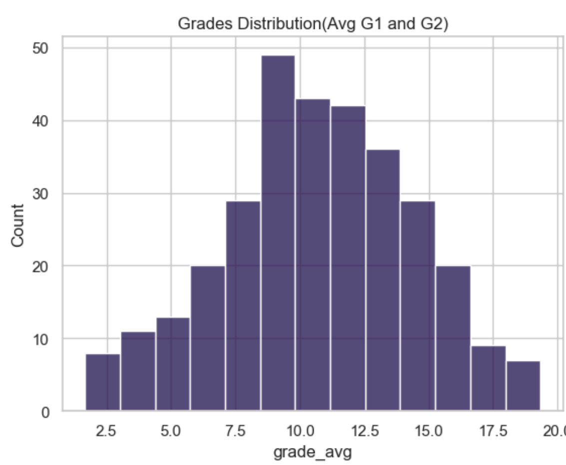
Exploratory Data Analysis (EDA):

Discrete Variables Count: 15
"age" _________ 15__22
"Medu" _________ 0__4
"Fedu" _________ 0__4
"traveltime" _________ 1__4
"studytime" _________ 1__4
"failures" _________ 0__3
"famrel" _________ 1__5
"freetime" _________ 1__5
"goout" _________ 1__5
"Dalc" _________ 1__5
"Walc" _________ 1__5
"health" _________ 1__5
"G1" _________ 5__19
"G2" _________ 0__19
"G3" _________ 0__20

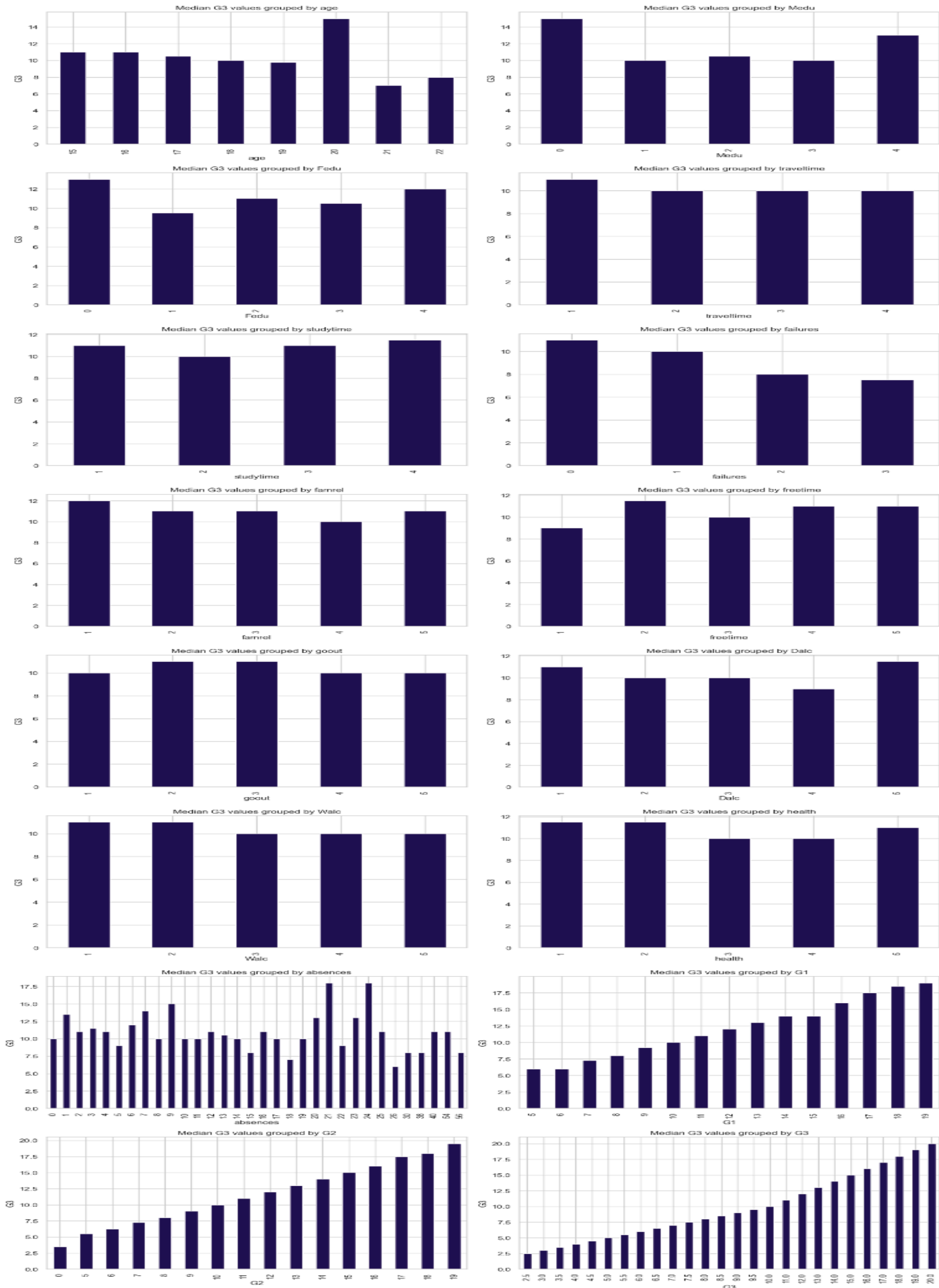
Number of categorical variables: 17
"school" _________ 2
"sex" _________ 2
"address" _________ 2
"famsize" _________ 2
"Pstatus" _________ 2
"Mjob" _________ 5
"Fjob" _________ 5
"reason" _________ 4
"guardian" _________ 3
"schoolsup" _________ 2
"famsup" _________ 2
"paid" _________ 2
"activities" _________ 2
"nursery" _________ 2
"higher" _________ 2
"internet" _________ 2
"romantic" _________ 2

Distribution of grades:

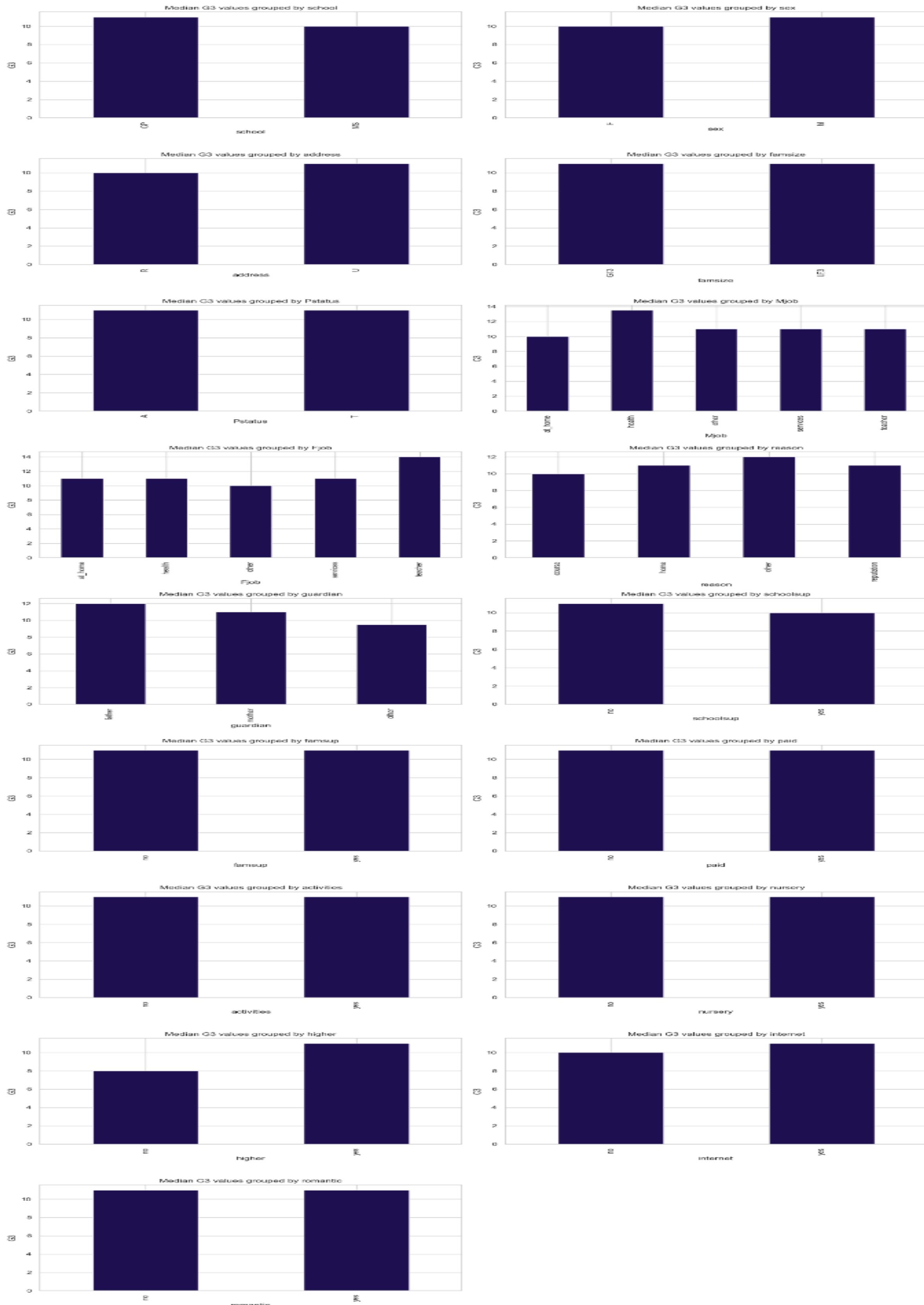




Numerical Features vs Target Grade:

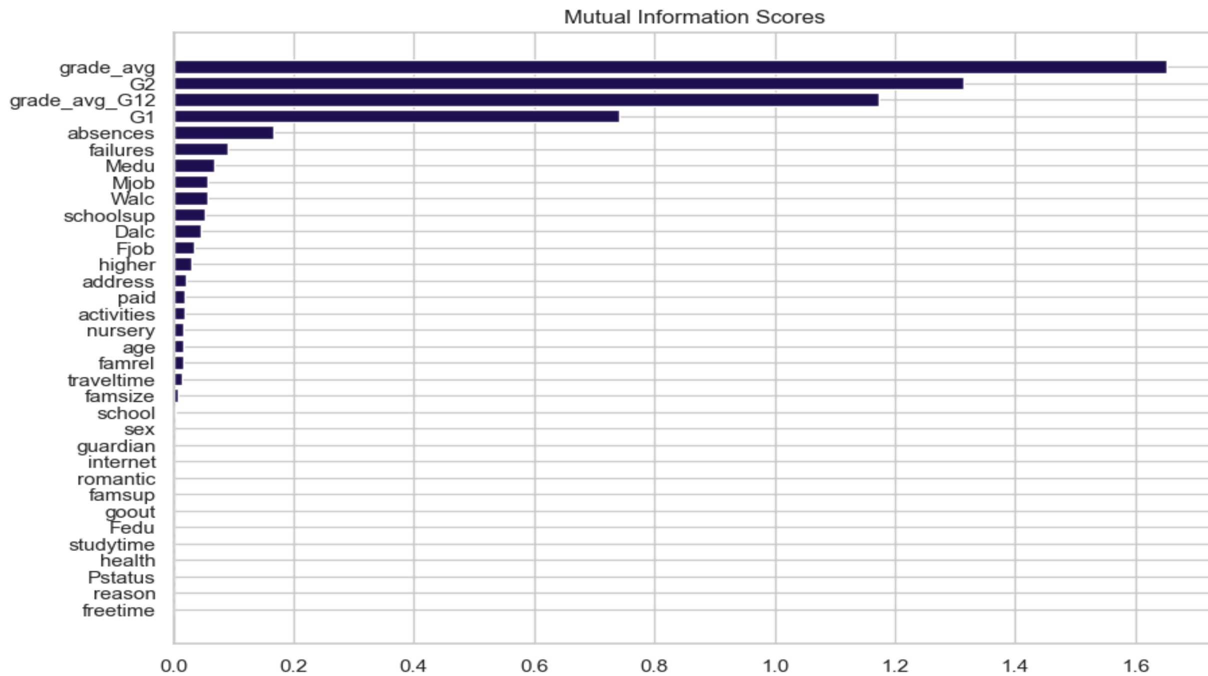


Categorical features vs Target Grade



Mutual Information scores:

G1 and G2 have a strong relationship with G3. But the absences and failures also indicate some relationship.



Here's a correlation map showing some of the features with high MI scores:

Feature encoding:

OrdinalEncoder for Binary Features:

OrdinalEncoder is used for binary features, which means these features have only two categories. This encoder assigns numerical labels to these categories.

OneHotEncoder for Categorical Features:

Explanation: OneHotEncoder is applied to categorical features, indicating that these features have more than two categories. It creates binary columns for each category, representing the presence or absence of that category.

SimpleImputer for Numerical Features:

Explanation: SimpleImputer is used to handle missing values in numerical features. This imputer replaces missing values with a specified strategy, such as the mean, median, or a constant.

SelectKBest is a feature selection method in scikit-learn that scores features using a statistical test and selects the top k features based on these scores. We have used the K values in the range 3-36 which includes all the features and finds the top 3 k values used in each model.

Models Used with Params:

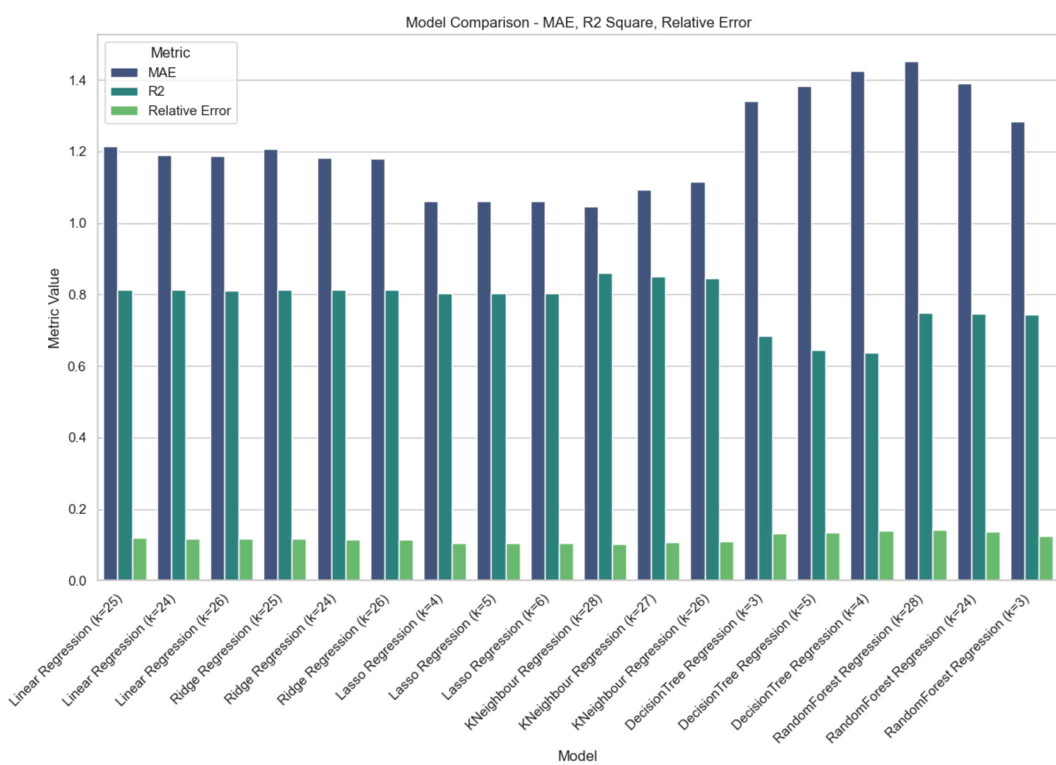
```
   ('Linear Regression', LinearRegression()),
    ('Ridge Regression', Ridge(solver='sag', alpha=0.5,
max_iter=10000, random_state=40)),
    ('Lasso Regression', Lasso(random_state=40,max_iter=10000)),
    ('KNeighbour Regression', KNeighborsRegressor(n_neighbors=3)),
    ('DecisionTree Regression',
DecisionTreeRegressor(random_state=40)),
    ('RandomForest Regression', RandomForestRegressor(n_estimators=10,
random_state=40))
```

The results we got:

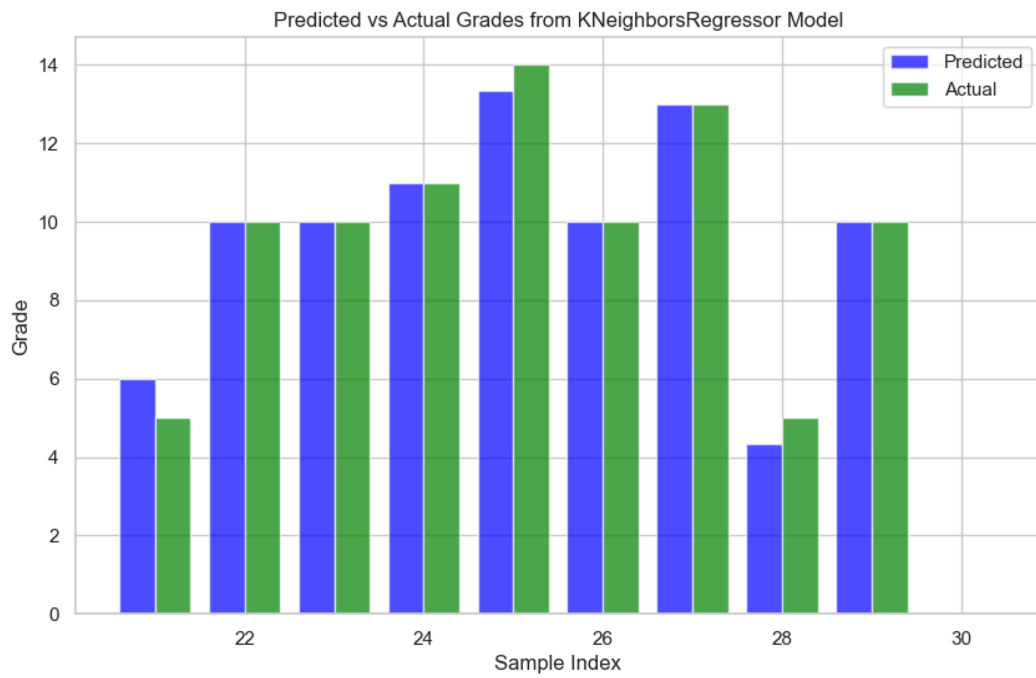
We used metrics like MAE, RMSE, R2 and Relative errors to gauge our models.

	Model	MAE	MSE	RMSE	Cross Validation	R2	Relative Error
0	Linear Regression (k=25)	1.215428	3.224133	1.795587	-5.013093	0.813006	0.118759
1	Linear Regression (k=24)	1.190540	3.232203	1.797833	-4.970016	0.812538	0.116328
2	Linear Regression (k=26)	1.187025	3.252390	1.803438	-5.120451	0.811368	0.115984
3	Ridge Regression (k=25)	1.207288	3.202707	1.789611	-4.983514	0.814249	0.117964
4	Ridge Regression (k=24)	1.182317	3.210704	1.791844	-4.941860	0.813785	0.115524
5	Ridge Regression (k=26)	1.180140	3.235244	1.798678	-5.084883	0.812362	0.115311
6	Lasso Regression (k=4)	1.060212	3.384877	1.839804	-4.352850	0.803684	0.103593
7	Lasso Regression (k=5)	1.060212	3.384877	1.839804	-4.352850	0.803684	0.103593
8	Lasso Regression (k=6)	1.060212	3.384877	1.839804	-4.352850	0.803684	0.103593
9	KNeighbour Regression (k=28)	1.046875	2.394097	1.547287	-5.800239	0.861147	0.102290
10	KNeighbour Regression (k=27)	1.093750	2.583333	1.607275	-6.028410	0.850172	0.106870
11	KNeighbour Regression (k=26)	1.114583	2.649306	1.627669	-6.033043	0.846345	0.108906
12	DecisionTree Regression (k=3)	1.340513	5.431201	2.330494	-6.114020	0.685001	0.130981
13	DecisionTree Regression (k=5)	1.382812	6.145399	2.478992	-7.375995	0.643579	0.135115
14	DecisionTree Regression (k=4)	1.424479	6.242622	2.498524	-7.335845	0.637940	0.139186
15	RandomForest Regression (k=28)	1.453125	4.331875	2.081316	-4.984257	0.748759	0.141985
16	RandomForest Regression (k=24)	1.390625	4.393125	2.095978	-4.911338	0.745207	0.135878
17	RandomForest Regression (k=3)	1.282948	4.431463	2.105104	-5.691188	0.742984	0.125357

Visual of how our models fared:



Let’s look at our best model’s prediction using KNeighbours regression model:

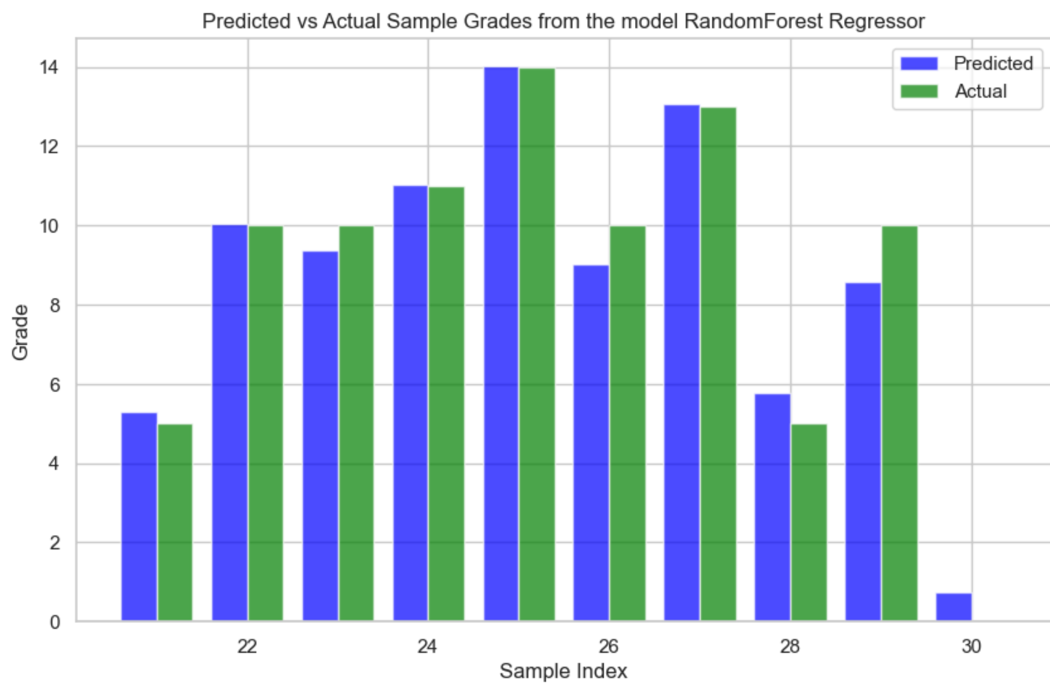


Not bad at all HUH!

We also tried Randomized search cv to tune the randomForestRegressor model and the best model used 35 features to predict. This model did well as well.

```
▾ RandomForestRegressor
RandomForestRegressor(max_features=35, n_estimators=71, random_state=40)
```

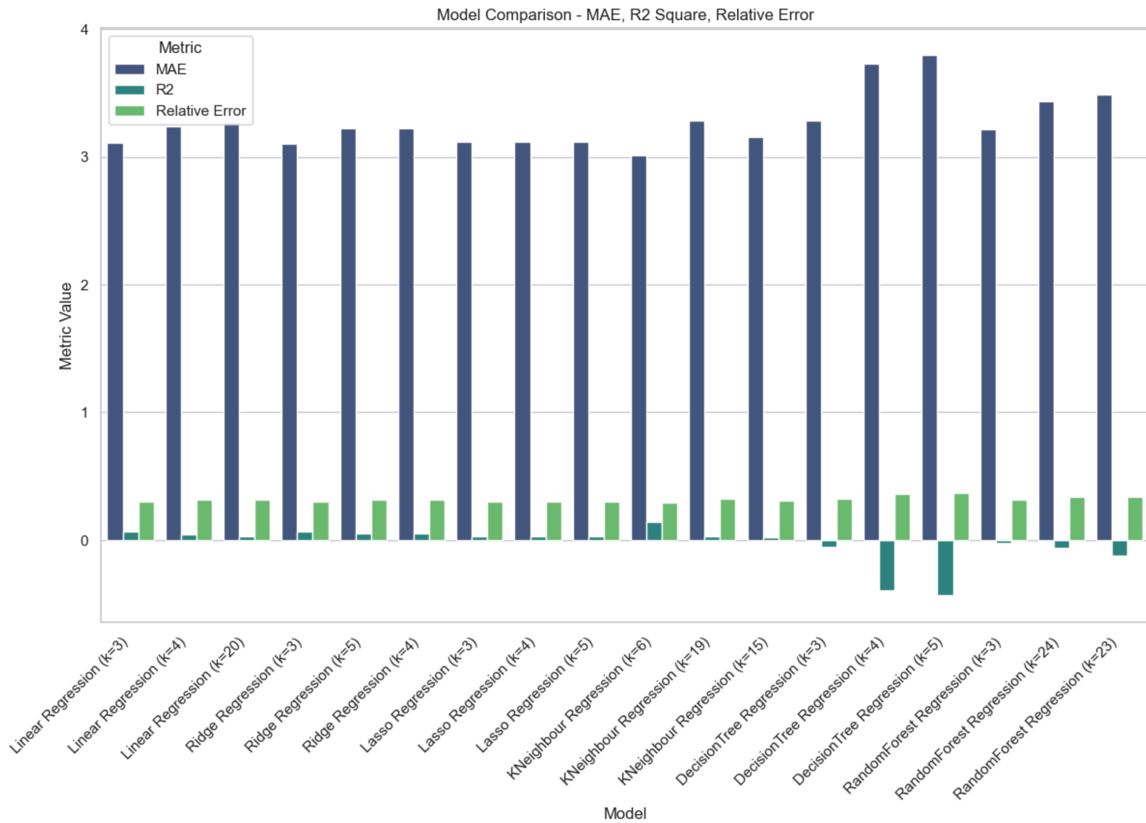
Let's look at a sample of test prediction:



For the bonus part where we predict G3 without using the Grades 1 and 2, we reused the same models with a little bit of reengineered features like

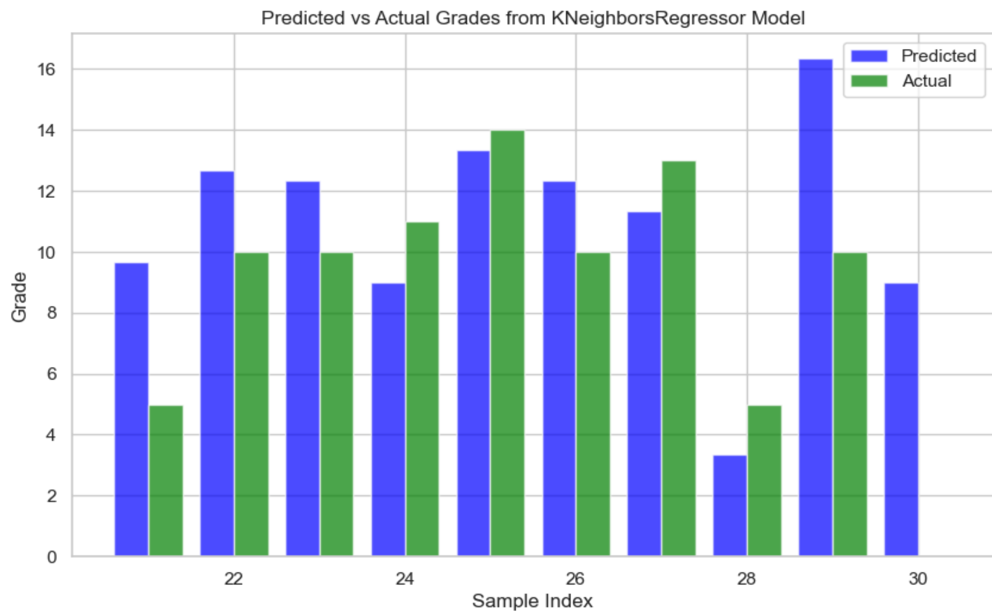
```
df['Pedu'] = (df['Medu'] + df['Fedu'])/2

# non study time

df['non_studytime'] = (10 - df[['studytime']])

# effort index based on these features combined

df['effort_index'] = df[['non_studytime','freetime','failures','absences','goout','Walc',]].sum(axis=1)

df['Talc'] = df['Dalc']+ df['Walc']
```

Let's take a look at how our models did.

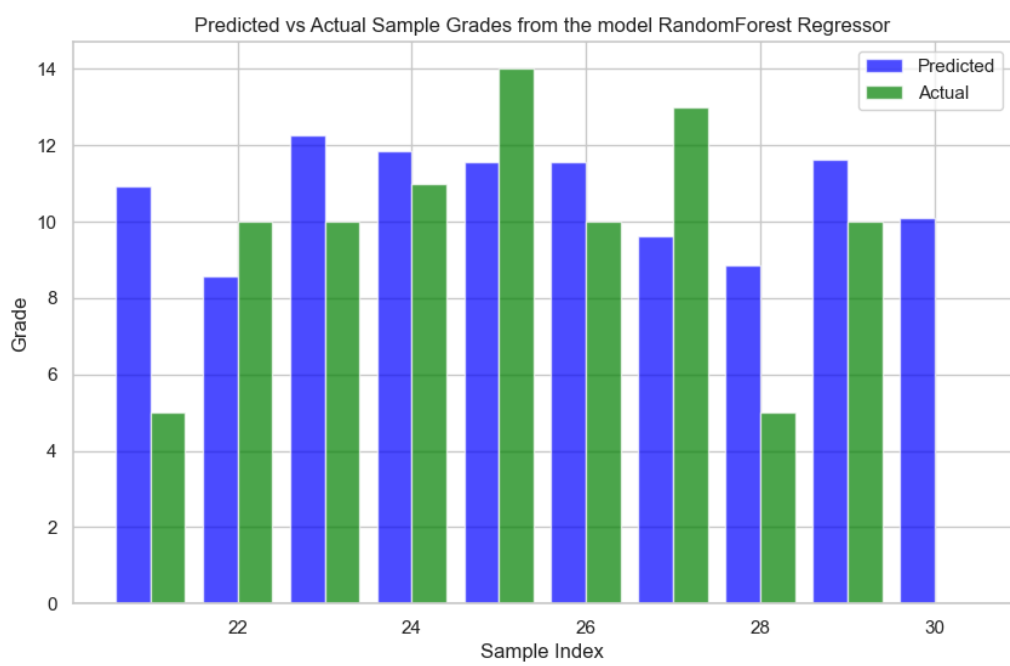


Models did worse compared to models with grade columns :)

But it’s not so bad when you look at the visuals!!!



Again with hypertuning and Random Forest Regression our result looks like below:



It's definitely not a great prediction but we did the best we could without using the previous grades.

We definitely recommend using Random Forest Regression and KNeighbours regression.