Names: Shayan Asif, Rafi Te

Linear Regression Analysis Report

For this project, we chose a dataset called Student Performance Dataset from the UCI Machine Learning (https://archive.ics.uci.edu/dataset/320/student+performance). The data includes demographic data (gender, family background, parental education, etc.), social data (internet access, romantic relationships, time spent with friends, etc.), and academic data (past grades, study time, failures, etc.). The key aim of this project is to make predictions on the final grade (G3) of the students based on these features. Academic performance prediction is not only applicable in measuring the performance of students but also can assist the teacher, schools and policymakers in developing specific interventions to address students at risk of underperforming.

Main Goal: Target variable is G3, which is the final grade.

Data:

- There are 395 students in the math dataset and 33 attributes / columns.
- We will be choosing the attributes that have the strongest correlation to the final grade. These will be the **features.**

Two regression methods were used to do predictive modeling:

1. Ordinary Least Squares (OLS) Regression with the use of the statsmodels library.
2. The SGDRegressor library of Scikit-learn was used to perform Stochastic Gradient Descent (SGD) Regression.

2.1 - Pre-processing

We loaded the dataset into a Pandas DataFrame using a public Github link.

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

5 rows × 33 columns

There were no missing values in any column and no duplicate values as well.

```
df.isna().sum()
```

	0
school	0
sex	0
age	0
address	0
famsize	0
Pstatus	0
Medu	0
Fedu	0
Mjob	0
Fjob	0
reason	0
guardian	0
traveltime	0

```
df.duplicated().sum()
```

```
np.int64(0)
```

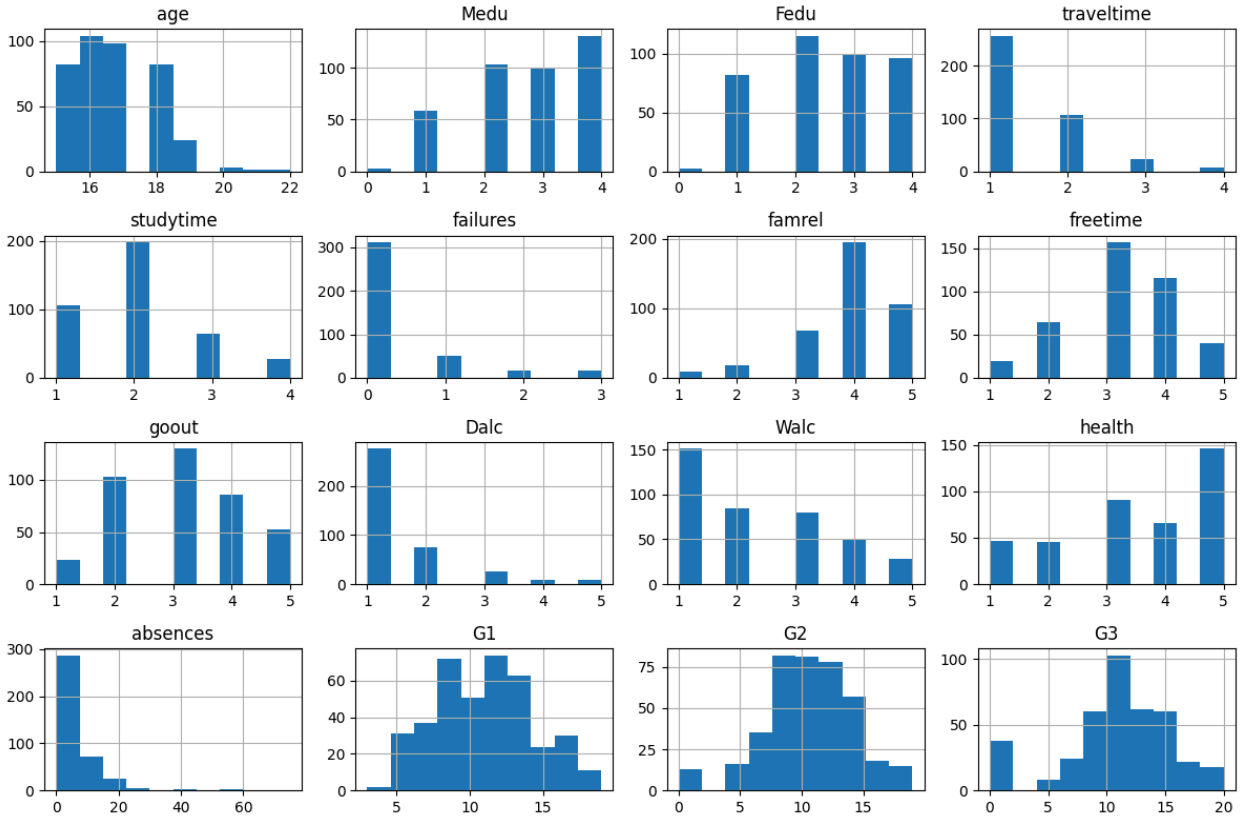
Pd.get dummies was used to create the dummy variables, including school, sex, address, Mjob, and Fjob. This was done to convert categorical to numeric values.

```
categorical_cols = [
    "school", "sex", "address", "famsize", "Pstatus",
    "Mjob", "Fjob", "reason", "guardian",
    "schoolsup", "famsup", "paid", "activities",
    "nursery", "higher", "internet", "romantic"
]

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=False)
df_encoded = df_encoded.astype(int)
```

Normalization:

We checked the distributions of the data. As you can see, G1, G2, G3 are somewhat normally distributed. Absences is skewed to the left. Some attributes have plain bars because they may be more discrete and don't have enough unique values. Even still, you can see that Medu, Fedu, goout, and freetime are somewhat normally distributed whereas as health, traveltime, failures, dalc, walc, are skewed either to the right or to the left.



Check Correlation and Choose Important Attributes:

For this part of the assignment, we checked the correlation for every attribute with the target variable G3 (final grade).

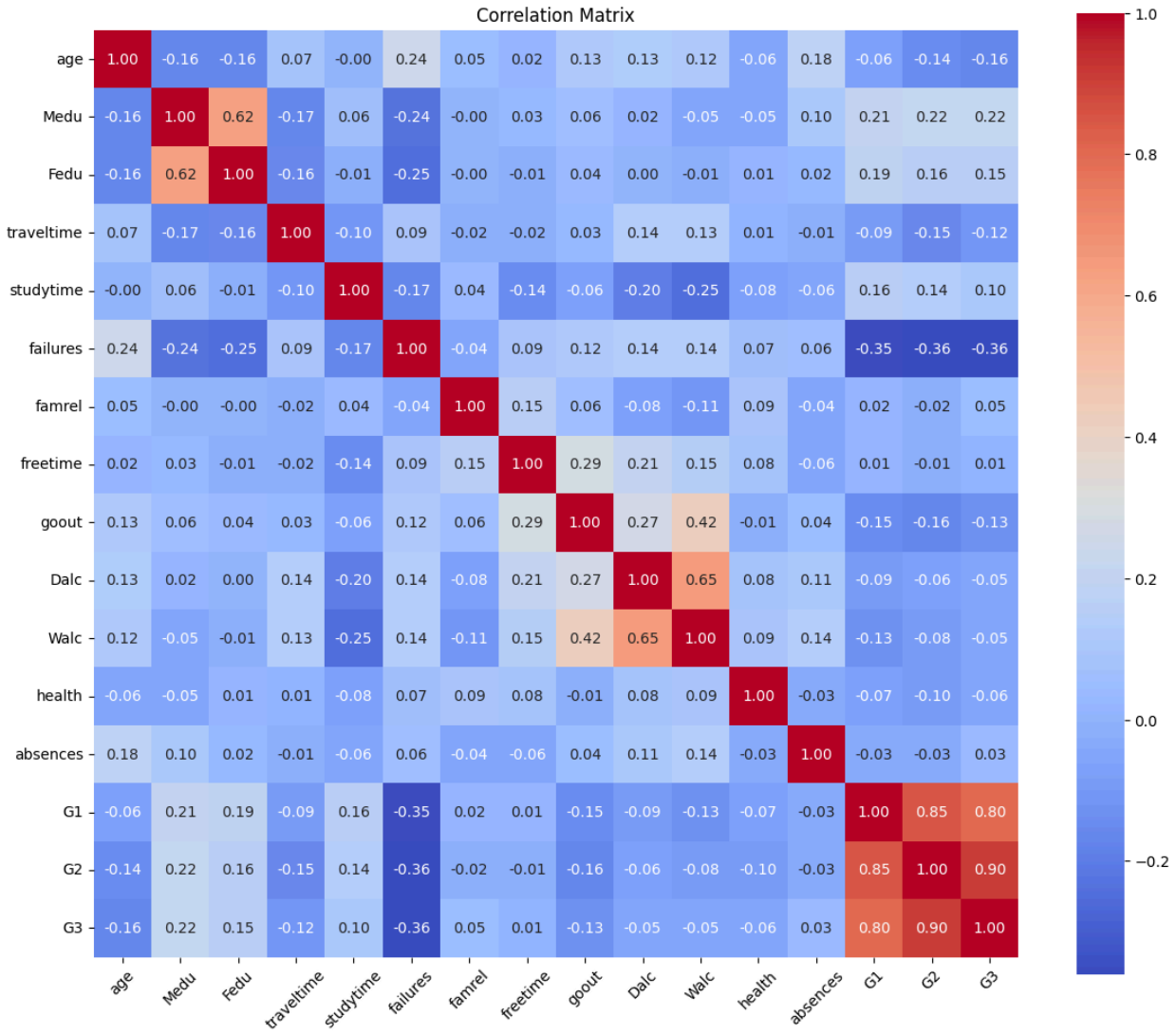
Correlation with Target Variable

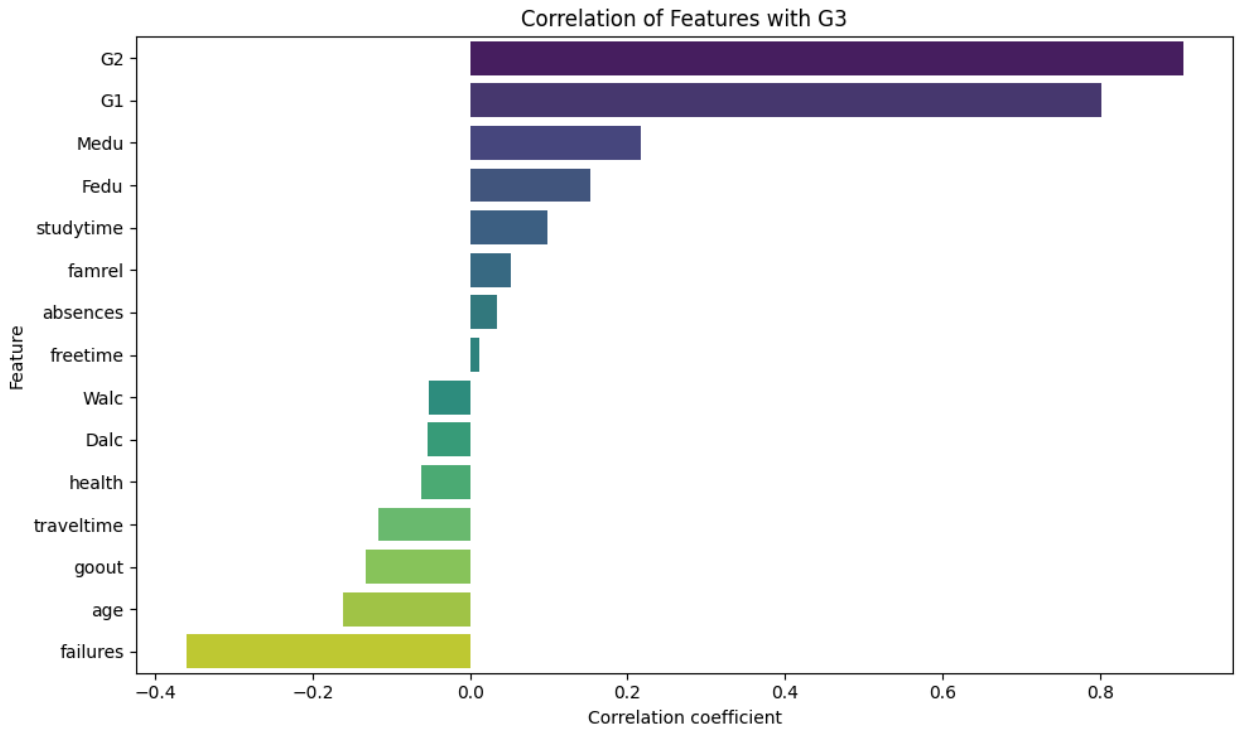
	G3
G3	1.000000
G2	0.904868
G1	0.801468
Medu	0.217147
Fedu	0.152457
studytime	0.097820
famrel	0.051363
absences	0.034247
freetime	0.011307
Walc	-0.051939
Dalc	-0.054660
health	-0.061335
traveltime	-0.117142
goout	-0.132791
age	-0.161579
failures	-0.360415

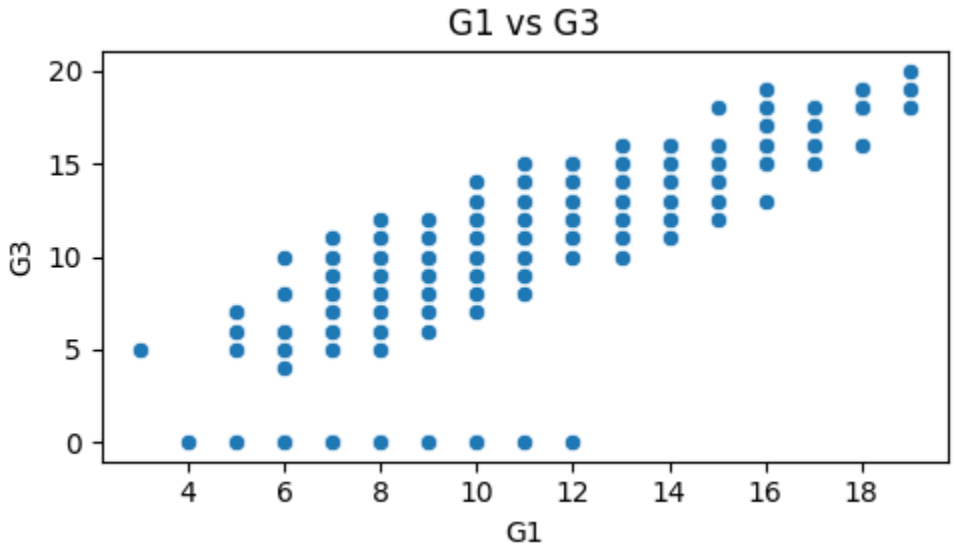
dtype: float64

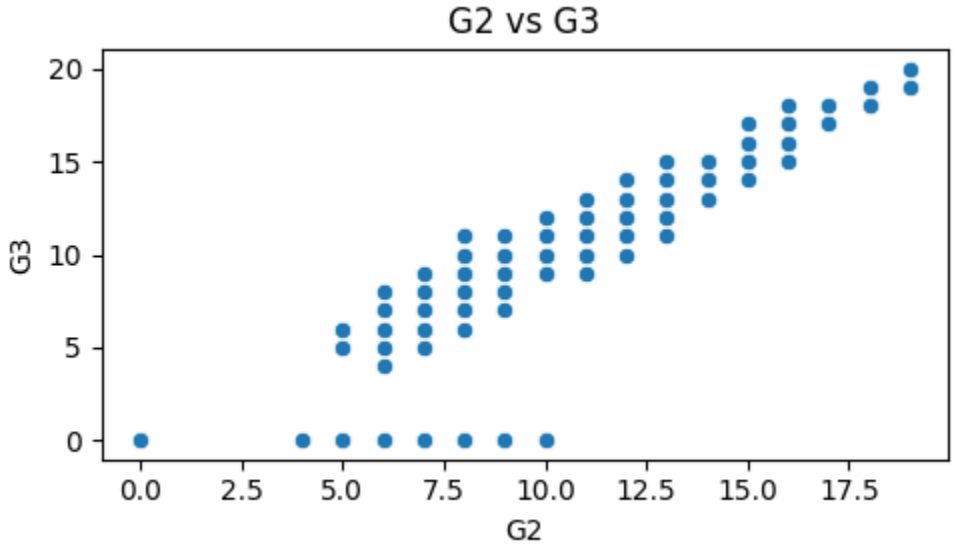
As you can see, G2, G1 have strong positive correlations with G3. There are other attributes with medium correlations such as Medu and Fedu. Failures and age have medium correlations as well, but in a negative way. These are the features we chose due to their strongest correlation.

Here are some plots that showcase this:


Correlation Matrix
age 1.00 -0.16 -0.16 0.07 -0.00 0.24 0.05 0.02 0.13 0.13 0.12 -0.06 0.18 -0.06 -0.14 -0.16
Medu -0.16 1.00 0.62 -0.17 0.06 -0.24 -0.00 0.03 0.06 0.02 -0.05 -0.05 0.10 0.21 0.22 0.22
Fedu -0.16 0.62 1.00 -0.16 -0.01 -0.25 -0.00 -0.01 0.04 0.00 -0.01 0.01 0.02 0.19 0.16 0.15
traveltime 0.07 -0.17 -0.16 1.00 -0.10 0.09 -0.02 -0.02 0.03 0.14 0.13 0.01 -0.01 -0.09 -0.15 -0.12
studytime -0.00 0.06 -0.01 -0.10 1.00 -0.17 0.04 -0.14 -0.06 -0.20 -0.25 -0.08 -0.06 0.16 0.14 0.10
failures 0.24 -0.24 -0.25 0.09 -0.17 1.00 -0.04 0.09 0.12 0.14 0.14 0.07 0.06 -0.35 -0.36 -0.36
famrel 0.05 -0.00 -0.00 -0.02 0.04 -0.04 1.00 0.15 0.06 -0.08 -0.11 0.09 -0.04 0.02 -0.02 0.05
freetime 0.02 0.03 -0.01 -0.02 -0.14 0.09 0.15 1.00 0.29 0.21 0.15 0.08 -0.06 0.01 -0.01 0.01
goout 0.13 0.06 0.04 0.03 -0.06 0.12 0.06 0.29 1.00 0.27 0.42 -0.01 0.04 -0.15 -0.16 -0.13
Dalc 0.13 0.02 0.00 0.14 -0.20 0.14 -0.08 0.21 0.27 1.00 0.65 0.08 0.11 -0.09 -0.06 -0.05
Walc 0.12 -0.05 -0.01 0.13 -0.25 0.14 -0.11 0.15 0.42 0.65 1.00 0.09 0.14 -0.13 -0.08 -0.05
health -0.06 -0.05 0.01 0.01 -0.08 0.07 0.09 0.08 -0.01 0.08 0.09 1.00 -0.03 -0.07 -0.10 -0.06
absences 0.18 0.10 0.02 -0.01 -0.06 0.06 -0.04 -0.06 0.04 0.11 0.14 -0.03 1.00 -0.03 -0.03 0.03
G1 -0.06 0.21 0.19 -0.09 0.16 -0.35 0.02 0.01 -0.15 -0.09 -0.13 -0.07 -0.03 1.00 0.85 0.80
G2 -0.14 0.22 0.16 -0.15 0.14 -0.36 -0.02 -0.01 -0.16 -0.06 -0.08 -0.10 -0.03 0.85 1.00 0.90
G3 -0.16 0.22 0.15 -0.12 0.10 -0.36 0.05 0.01 -0.13 -0.05 -0.05 -0.06 0.03 0.80 0.90 1.00
age Medu Fedu traveltime studytime failures famrel freetime goout Dalc Walc health absences G1 G2 G3
1.0 0.8 0.6 0.4 0.2 0.0 −0.2


Correlation of Features with G3
G2
G1
Medu
Fedu
studytime
famrel
absences
freetime
Walc
Dalc
health
traveltime
goout
age
failures
Feature
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Correlation coefficient


G1 vs G3
G3
20
15
10
5
0
4
6
8
10
12
14
16
18
G1



SGD Model:

The standardized features (StandardScaler) were used in an SGDRegressor. Some of the hyperparameters that were tuned include learning rate (alpha), penalty (l2 vs l1), and the number of iterations (max_iter).

- Performance metrics:
  - R^2 on training set.
  - R^2 on test set.
  - Mean Absolute Error (MAE) as well as Mean Squared Error (MSE).

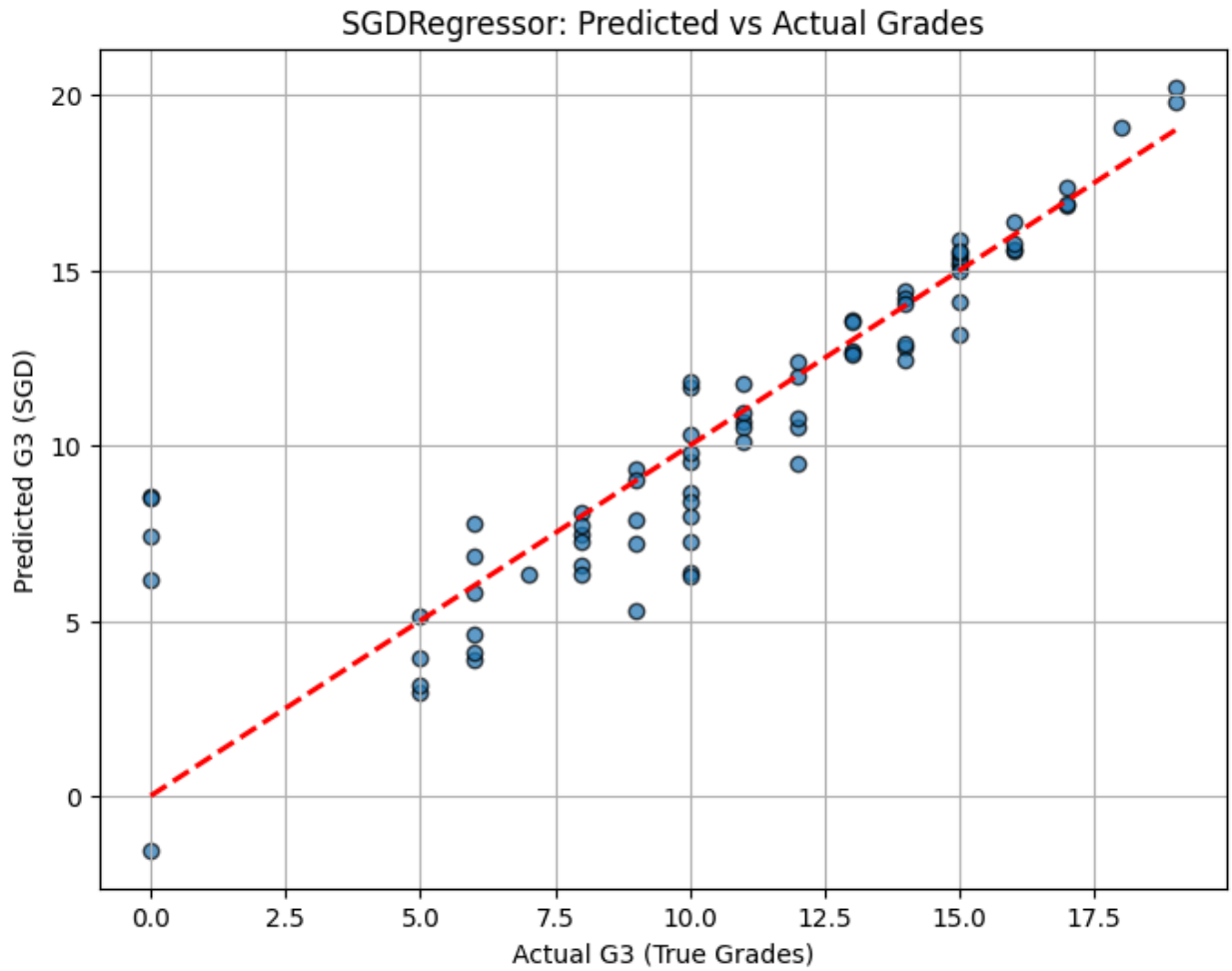
```python
# Create and train the model
sgd_model = SGDRegressor(
    loss='squared_error',   # Ordinary Least Squares loss
    penalty='l2',           # Ridge regularization (you can also try 'l1', 'elasticnet')
    alpha=0.001,            # Regularization strength
    max_iter=1000,          # Number of iterations
    learning_rate='invscaling',  # Can try 'optimal', 'adaptive', etc.
    eta0=0.01,              # Initial learning rate
    random_state=42
)

sgd_model.fit(X_train_scaled, y_train)

y_train_pred = sgd_model.predict(X_train_scaled)
y_test_pred = sgd_model.predict(X_test_scaled)

print("Training R²:", r2_score(y_train, y_train_pred))
print("Test R²:", r2_score(y_test, y_test_pred))
print("Train MSE:", mean_squared_error(y_train, y_train_pred))
print("Test MSE:", mean_squared_error(y_test, y_test_pred))
```

```
Training R²: 0.834976827282255
Test R²: 0.7745024805652284
Train MSE: 3.466231954853607
Test MSE: 4.623837294841627
```



This is the data from our initial parameters. Overall a test r2 of .77 is good but it can be better.

Tuning hyper-parameters:

We tested different hyper-parameters and these were the results:

	loss	penalty	alpha	learning_rate	eta0	max_iter	train_r2	test_r2	train_mse	test_mse
4	huber	l2	0.001	adaptive	0.010	2000	0.817918	0.786472	3.824537	4.378409
2	squared_error	l1	0.001	adaptive	0.010	1500	0.835306	0.775331	3.459314	4.606854
1	squared_error	l2	0.010	invscaling	0.010	1000	0.834935	0.775105	3.467112	4.611486
0	squared_error	l2	0.001	invscaling	0.010	1000	0.834977	0.774502	3.466232	4.623837
3	squared_error	elasticnet	0.001	optimal	0.001	2000	0.760457	0.652215	5.031482	7.131345

.786 was our best score, but we wrote some code that can get us an even better score by testing all the possible combinations:

```
# Try all combinations
for loss, penalty, alpha, eta0, lr in product(losses, penalties, alphas, etas, learning_rates):
    model = SGDRegressor(
        loss=loss,
        penalty=penalty,
        alpha=alpha,
        eta0=eta0,
        learning_rate=lr,
        max_iter=2000,
        tol=1e-3,
        random_state=42
    )
    model.fit(X_train_scaled, y_train)
    y_test_pred = model.predict(X_test_scaled)
    r2 = r2_score(y_test, y_test_pred)

    if r2 > best_score:
        best_score = r2
        best_params = (loss, penalty, alpha, eta0, lr)

print("Best Test R²:", best_score)
print("Best Hyperparameters:", best_params)
```

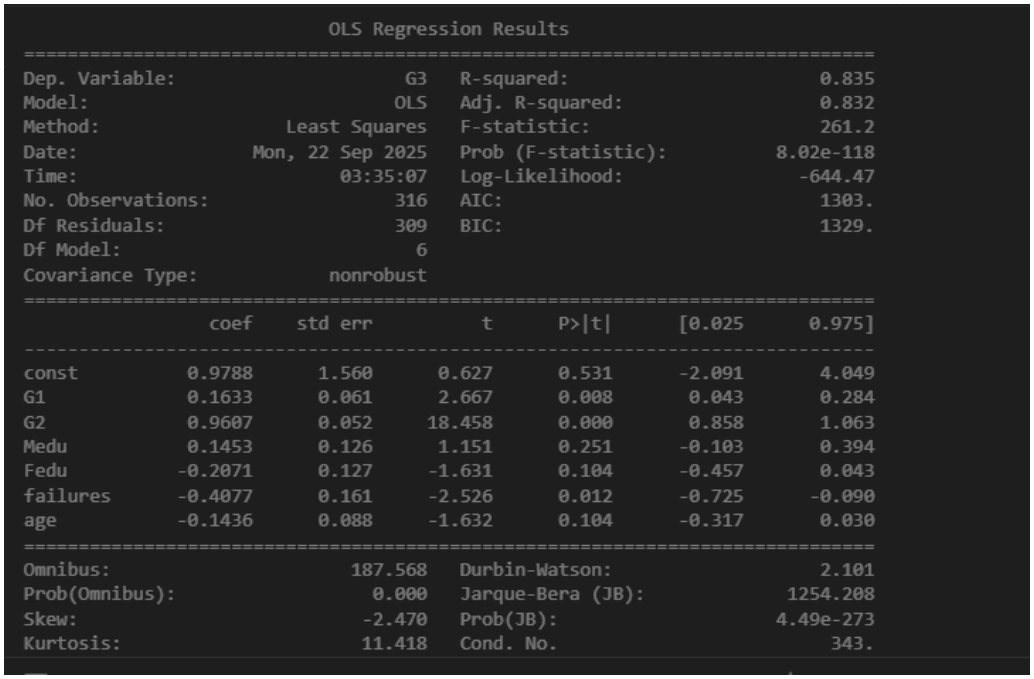
```
Best Test R²: 0.7920685308482791
Best Hyperparameters: ('huber', 'l1', 0.01, 0.01, 'adaptive')
```

Ultimately, these were the hyperparameters which gave us the best score of .792.

OLS Model:

G3 was the dependent variable and an OLS regression model was constructed. The outputs of the given model contained coefficients, standard errors, t-statistic, p-values, and R2 values.

- Key findings:
  - G1 and G2 proved to be the most critical predictors of G3.
  - There were not so critical demographic characteristics (e.g., sex, address).
  - The model had a high R 2 that indicated a high level of explanatory power.

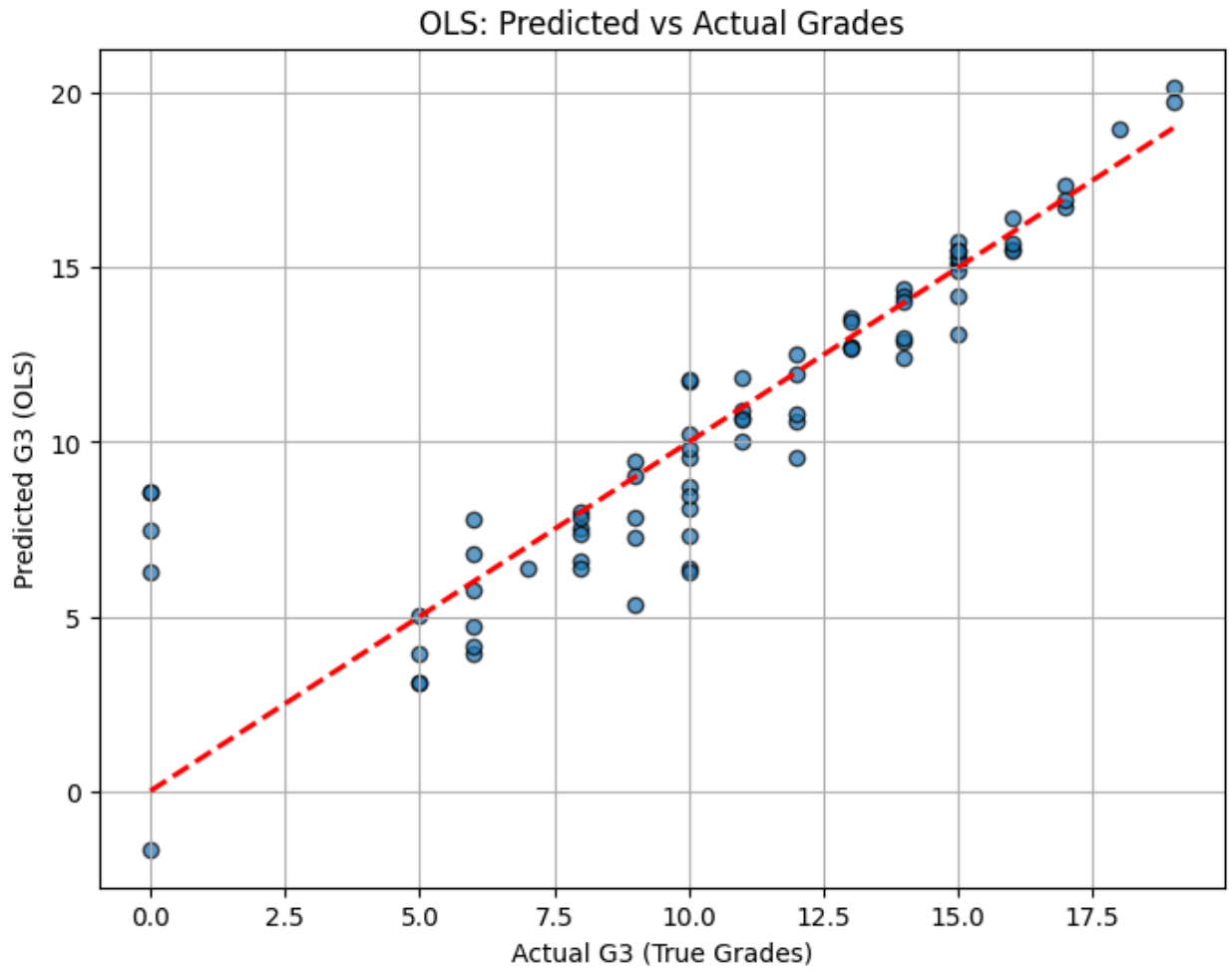


OLS Regression Results

Dep. Variable:	G3	R-squared:	0.835
Model:	OLS	Adj. R-squared:	0.832
Method:	Least Squares	F-statistic:	261.2
Date:	Mon, 22 Sep 2025	Prob (F-statistic):	8.02e-118
Time:	03:35:07	Log-Likelihood:	-644.47
No. Observations:	316	AIC:	1303.
Df Residuals:	309	BIC:	1329.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.9788	1.560	0.627	0.531	-2.091	4.049
G1	0.1633	0.061	2.667	0.008	0.043	0.284
G2	0.9607	0.052	18.458	0.000	0.858	1.063
Medu	0.1453	0.126	1.151	0.251	-0.103	0.394
Fedu	-0.2071	0.127	-1.631	0.104	-0.457	0.043
failures	-0.4077	0.161	-2.526	0.012	-0.725	-0.090
age	-0.1436	0.088	-1.632	0.104	-0.317	0.030

Omnibus:	187.568	Durbin-Watson:	2.101
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1254.208
Skew:	-2.470	Prob(JB):	4.49e-273
Kurtosis:	11.418	Cond. No.	343.



Conclusion:

We have managed to use Ordinary Least Squares (OLS) regression and Stochastic Gradient Descent (SGD) regression to Student Performance dataset in this project to predict final grade (G3). We were able to find that past academic performance (G1 and G2) had the highest predictive value with respect to final grades followed by the demographic and lifestyle factors. The OLS model was more interpretable and had a clear understanding of the significance of the features, but the SGD model was more flexible and efficient in case of large-scale learning. On the whole, both models have done quite a good job, with high R-squared scores, and a combination of them illustrates the significance of integrating machine learning solutions that can be scaled with statistical interpretability in educational analytics.