Student Performance Prediction

Team - 9

Sai Krishna Nandyala - 1002144528

&

Vinay Kumar Algam - 1002176591

Contents

Problem Statement

In the context of this project, our primary objective is to develop a predictive model capable of accurately forecasting the final grades of students attending two distinct schools. Leveraging the various features provided in the dataset, our aim is to create a robust and reliable system that can contribute to understanding the factors influencing students' academic performance. This predictive model will not only offer insights into the relationships between different features and final grades but also has the potential to assist educators, administrators, and policymakers in making informed decisions to enhance the overall educational outcomes in these schools.

Data Set Introduction

The data set used is collected from https://www.kaggle.com/datasets/devansodariya/student-performance-data/data

There are 395 instances and 33 columns in the dataset.

Student Performance Data was obtained in a survey of student's math course in secondary school.

```
df=pd.read_csv("student_data.csv")
df.head()
```

:

	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

Data Description

The dataset consists of 32 dependent variables and 1 target variable.

Out of the 32 dependent variables, 17 are of string format and 15 are of integer data-type.

The target variable i.e; Final Grade(G3) is integer.

Feature Description

1. **school** - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2. **sex** - student's sex (binary: 'F' - female or 'M' - male)
3. **age** - student's age (numeric: from 15 to 22)
4. **address** - student's home address type (binary: 'U' - urban or 'R' - rural)
5. **famsize** - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6. **Pstatus** - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7. **Medu** - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
8. **Fedu** - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

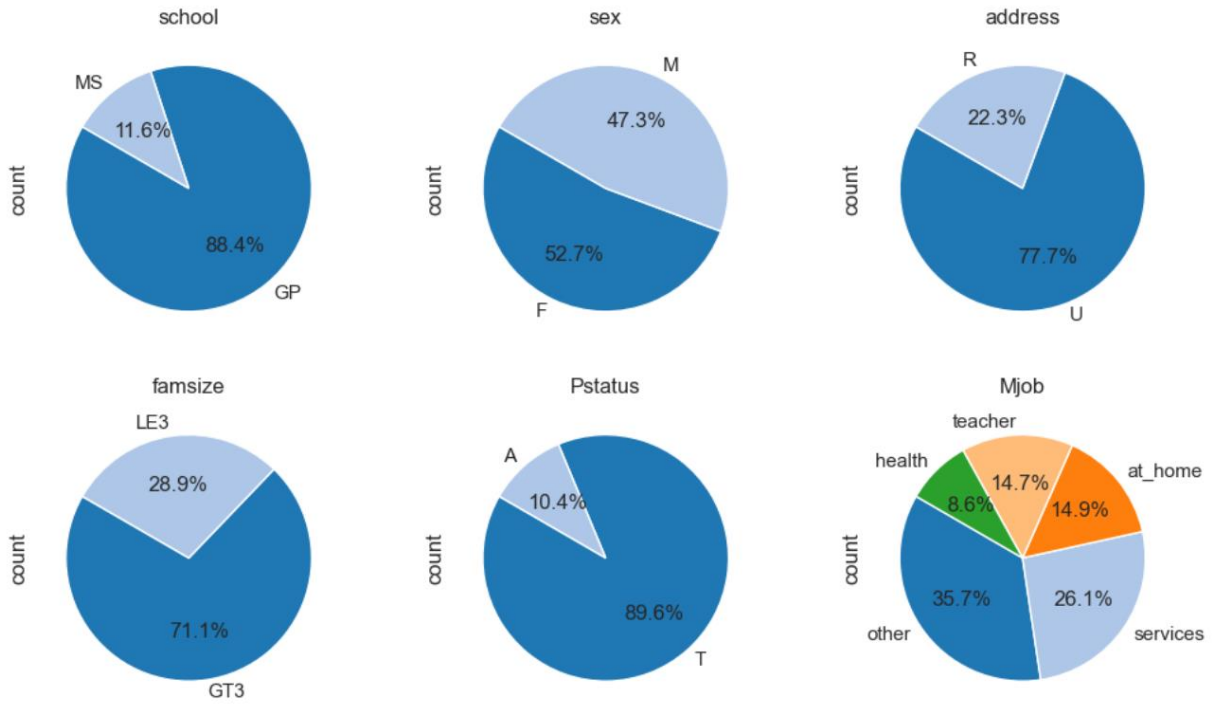
9. **Mjob** - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10. **Fjob** - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11. **reason** - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12. **guardian** - student's guardian (nominal: 'mother', 'father' or 'other')
13. **traveltime** - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14. **studytime** - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. **failures** - number of past class failures (numeric: n if 1<=n<3, else 4)
16. **schoolsup** - extra educational support (binary: yes or no)
17. **famsup** - family educational support (binary: yes or no)
18. **paid** - extra paid classes within the course subject (Portuguese) (binary: yes or no)
19. **activities** - extra-curricular activities (binary: yes or no)
20. **nursery** - attended nursery school (binary: yes or no)
21. **higher** - wants to take higher education (binary: yes or no)
22. **internet** - Internet access at home (binary: yes or no)
23. **romantic** - with a romantic relationship (binary: yes or no)
24. **famrel** - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25. **freetime** - free time after school (numeric: from 1 - very low to 5 - very high)
26. **goout** - going out with friends (numeric: from 1 - very low to 5 - very high)
27. **Dalc** - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. **Walc** - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29. **health** - current health status (numeric: from 1 - very bad to 5 - very good)
30. **absences** - number of school absences (numeric: from 0 to 93)
31. **G1** - first period grade (numeric: from 0 to 20)
32. **G2** - second period grade (numeric: from 0 to 20)
33. **G3** - final grade (numeric: from 0 to 20, output target)

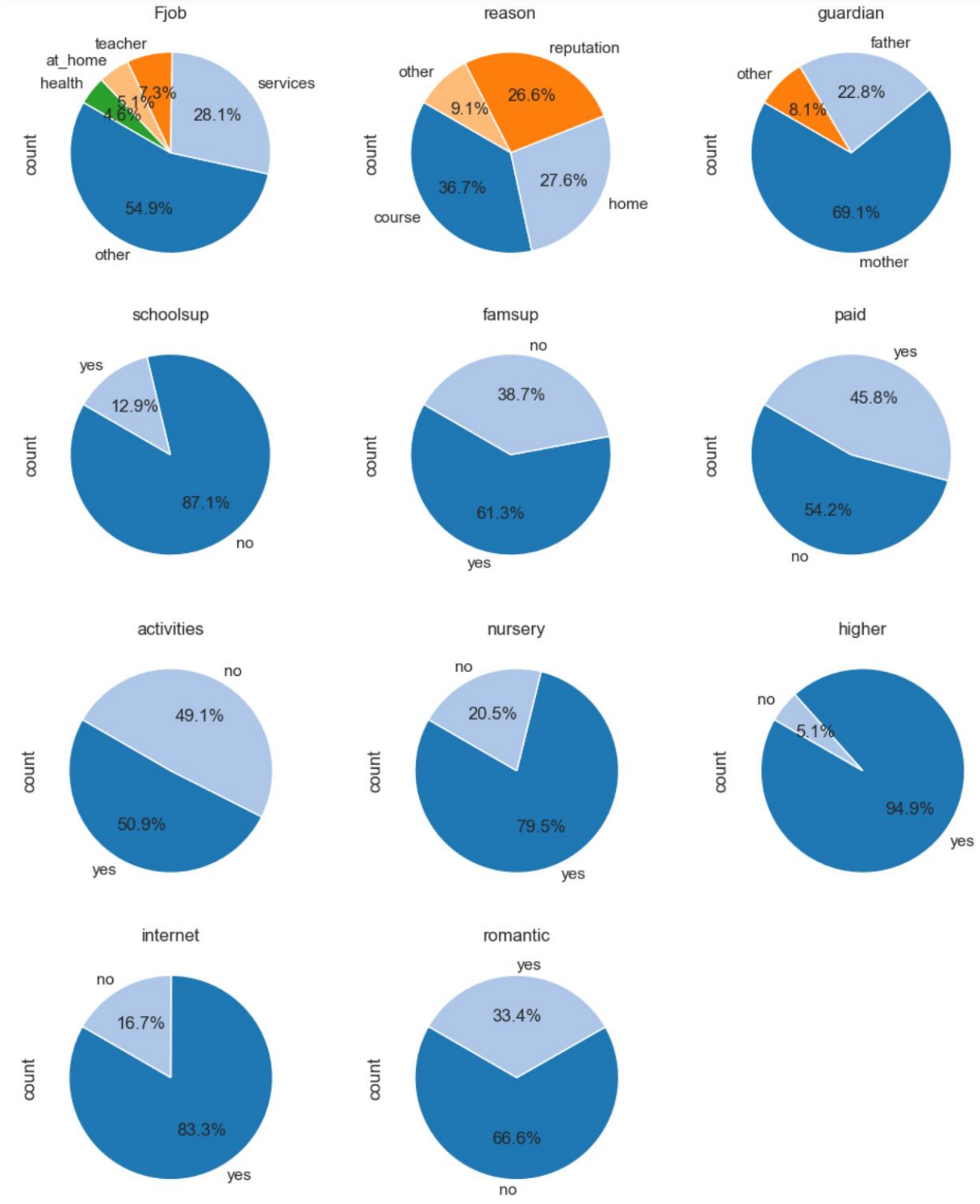
String Data Distribution

- Understanding the distribution of categorical features can help determine how important they are. A few categories might have a big effect on the target variable, which would affect feature choice and model interpretation.
- Model performance may be impacted by imbalances in categorical data. For instance, during model training and evaluation, a binary classification problem with a highly imbalanced target variable might need extra consideration.
- Outliers are categories that are uncommon or unique. Determining whether to combine uncommon categories, establish a new category, or use alternative tactics requires an understanding of the distribution.

We used pie charts and bar plots to analyze categorical data.

```
plt.figure(figsize=(12, 22))
sns.set_theme(style='darkgrid',palette='tab20')
for i, col in enumerate(string_col, start=1):
    plt.subplot(6, 3, i)
    df[col].value_counts().plot.pie(autopct='%1.1f%%',startangle=150)
    plt.title(f'{col}')
plt.show()
```




Fjob
teacher
at_home
health
services
7.3%
5.1%
4.6%
28.1%
54.9%
other
count
reason
reputation
other
26.6%
9.1%
27.6%
36.7%
home
course
count
guardian
father
other
22.8%
8.1%
69.1%
mother
count
schoolsup
yes
12.9%
87.1%
no
count
famsup
no
38.7%
61.3%
yes
count
paid
yes
45.8%
54.2%
no
count
activities
no
49.1%
50.9%
yes
count
nursery
no
20.5%
79.5%
yes
count
higher
no
5.1%
94.9%
yes
count
internet
no
16.7%
83.3%
yes
count
romantic
yes
33.4%
66.6%
no
count

Numerical Data Distribution

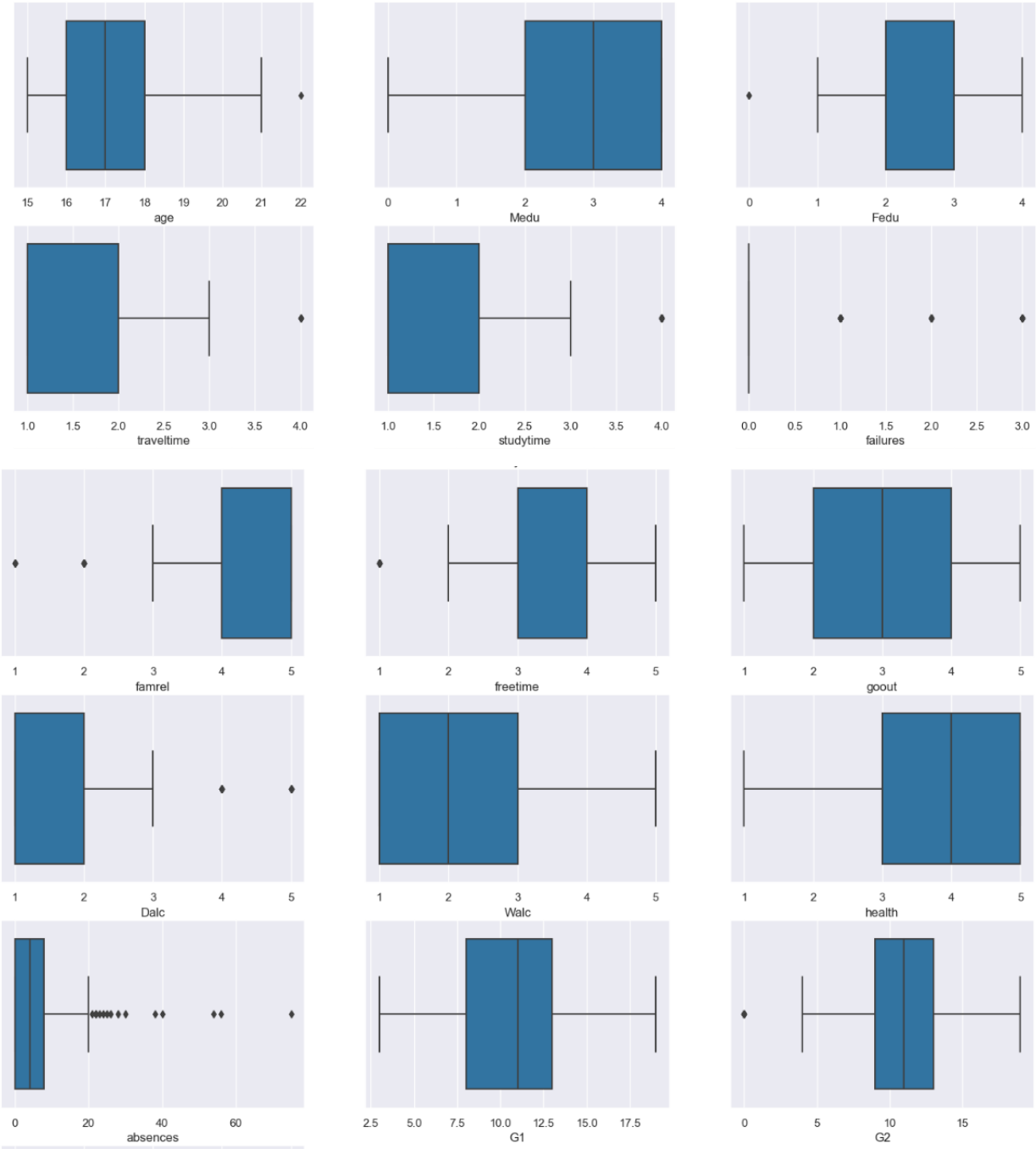
- Distributions of numerical data aid in locating outliers that could interfere with model training. Techniques like winsorization, truncation, or removal are available to deal with outliers.
- Certain data distributions are assumed by certain machine learning algorithms. The selection of appropriate models is guided by an understanding of the characteristics of the numerical data. For example, linear models might need scaled features, but decision trees are less sensitive to feature scaling. We used bar plots, boxplots and density functions to visualize numerical data.

Below is the summary of numerical data.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	395.0	16.696203	1.276043	15.0	16.0	17.0	18.0	22.0
Medu	395.0	2.749367	1.094735	0.0	2.0	3.0	4.0	4.0
Fedu	395.0	2.521519	1.088201	0.0	2.0	2.0	3.0	4.0
traveltime	395.0	1.448101	0.697505	1.0	1.0	1.0	2.0	4.0
studytime	395.0	2.035443	0.839240	1.0	1.0	2.0	2.0	4.0
failures	395.0	0.334177	0.743651	0.0	0.0	0.0	0.0	3.0
famrel	395.0	3.944304	0.896659	1.0	4.0	4.0	5.0	5.0
freetime	395.0	3.235443	0.998862	1.0	3.0	3.0	4.0	5.0
goout	395.0	3.108861	1.113278	1.0	2.0	3.0	4.0	5.0
Dalc	395.0	1.481013	0.890741	1.0	1.0	1.0	2.0	5.0
Walc	395.0	2.291139	1.287897	1.0	1.0	2.0	3.0	5.0
health	395.0	3.554430	1.390303	1.0	3.0	4.0	5.0	5.0
absences	395.0	5.708861	8.003096	0.0	0.0	4.0	8.0	75.0
G1	395.0	10.908861	3.319195	3.0	8.0	11.0	13.0	19.0
G2	395.0	10.713924	3.761505	0.0	9.0	11.0	13.0	19.0
G3	395.0	10.415190	4.581443	0.0	8.0	11.0	14.0	20.0

```
plt.figure(figsize=(18,23))
for i,col in enumerate(numerical_col, start=1):
    plt.subplot(6, 3, i)
    sns.boxplot(x=df[col])
plt.show()
```



Exploratory Data Analysis

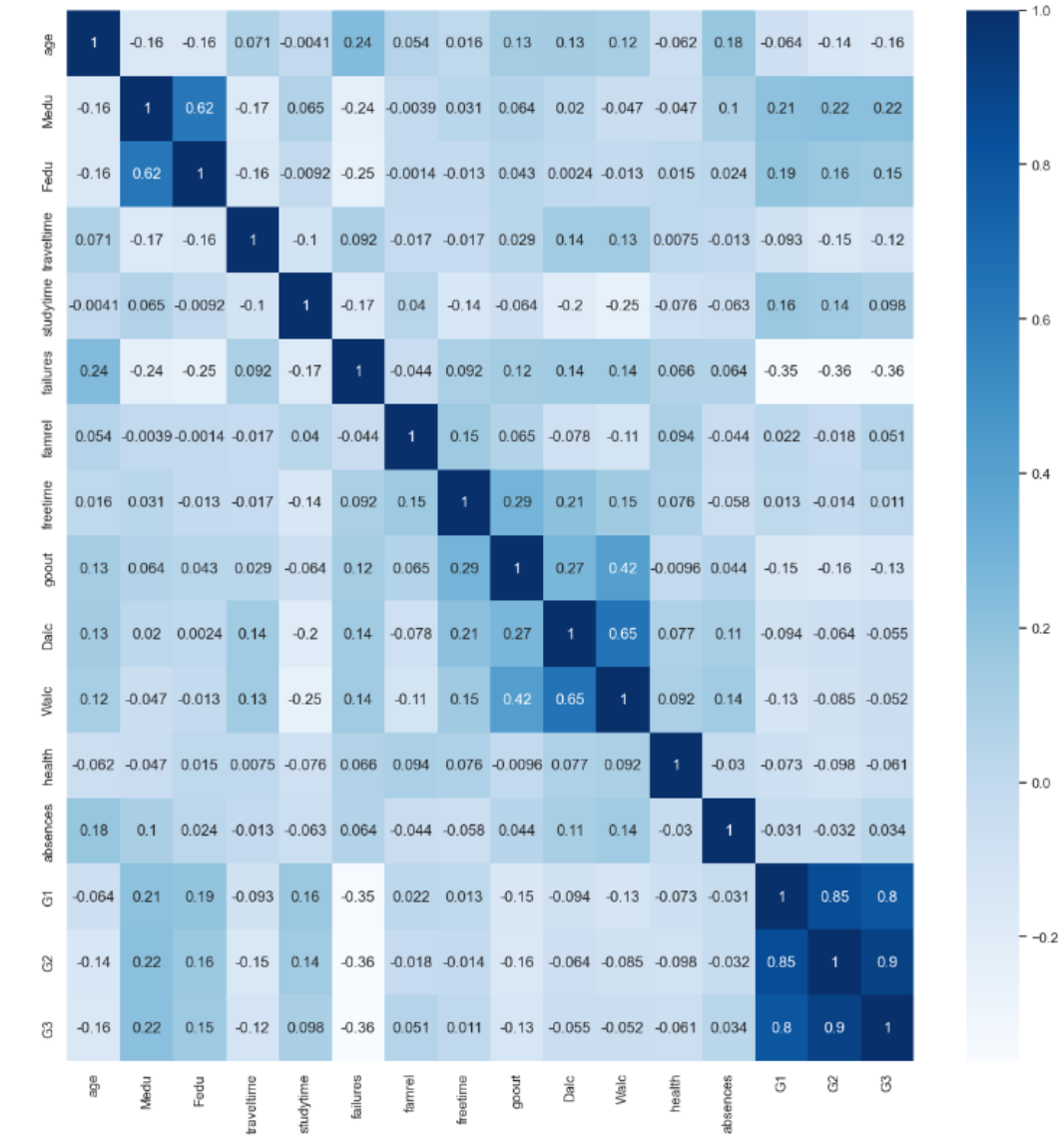
There are no null values in this data set.

Correlation Heat Map

We are plotting a correlation heat map using the numerical features in the dataset.

```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(new_df[['age', 'Medu', 'Fedu', 'traveltime', 'studytime',
                    'failures', 'famrel', 'freetime', 'goout', 'Dalc',
                    'Walc', 'health', 'absences', 'G1', 'G2', 'G3']].corr(),annot=True,ax=ax,cmap="Blues")
```

.]: <Axes: >



We can observe that G1 vs G3 and G2 vs G3 have the darkest squares which implies that they have the highest correlation to G3.

- G1 and G3 has correlation value of 0.8
- G2 and G3 has correlation value of 0.9

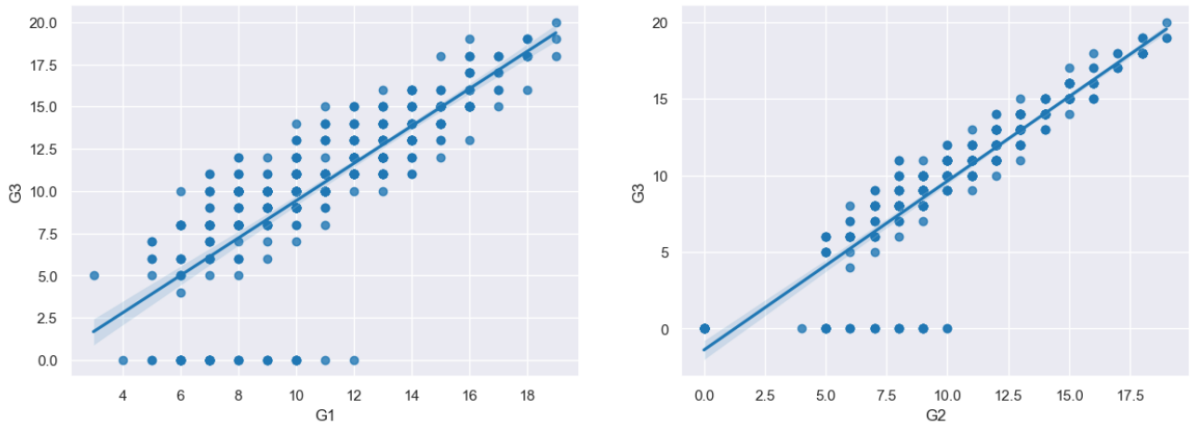
G1 and G2 vs Final Grade(G3)

Regplot is a data visualization tool created especially for visualizing relationships between variables, and Seaborn is a data visualization library built on top of Matplotlib.

We have utilized the parameters x = G1 and y = G3, which indicate which columns in the dataframe belong on the X and Y axes, respectively. In this instance, it suggests that the x-axis should be used to plot the values in the "G1" column and the y-axis to plot the values in the "G3" column. Comparable procedures have been carried out for G2 and G3. The "G1" and "G2" variables' linear relationship with "G3" is shown graphically by the regplot function, which also fits a regression line to the data in addition to producing a scatter plot. A positive correlation is indicated by an upward slope of the line. The inclusion of the scatter plot points enables a visual examination of the distribution of the data points surrounding the regression line.

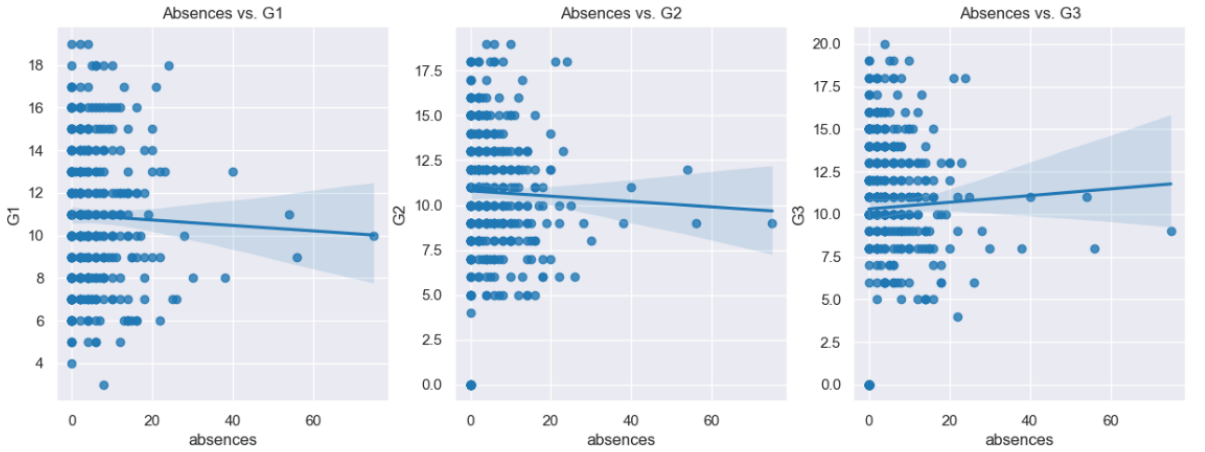
First period grade and Second period grade seem to have very high correlation with Final Grade(G3).

```
grades=['G1','G2']
fig, axes = plt.subplots(1, 2,figsize=(15,5))
for i in range(2):
    sns.regplot(ax=axes[i],x=grades[i], y="G3", data=df)

plt.show();
```



Absences vs Final Grade(G3)

```
grades=['G1','G2','G3']
fig, axes = plt.subplots(1, 3,figsize=(15,5))
for i in range(3):
    sns.regplot(ax=axes[i],x="absences", y=grades[i], data=df).set(title=f'Absences vs. {grades[i]}');

plt.show();
```



There seems to be a very low correlation between Absences and G3 and this is interesting because one might think with more absences the student performance decreases.

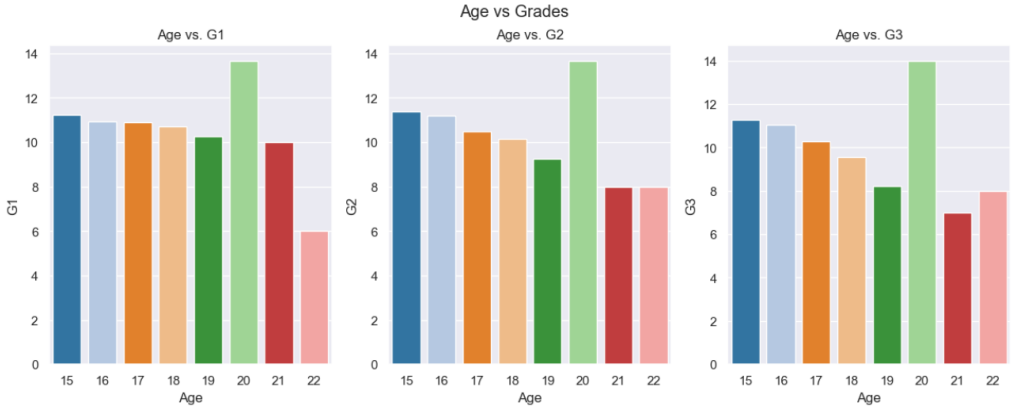
This low correlation might be because absent students revise the material missed, effectively accounting for their absence.

Age vs Grades

Grades seem to be indirectly proportional to age. In the below bar plots average grade shows a downward trend to age but there is a peak at age 20.

- Students at age 20 have the highest average grade.

```
grades=['G1','G2','G3']
fig, axes = plt.subplots(1, 3,figsize=(15,5))
fig.suptitle('Age vs Grades')
for i in range(3):
    sns.barplot(ax=axes[i], data=age_grade, x='age', y=grades[i]).set(xlabel='Age',
                                                                      ylabel=grades[i],
                                                                      title=f'Age vs. {grades[i]}');

plt.show();
```



General Health vs Final Grade

We tried to create a new column called General Health where the calculated scores would be scored.
The above formula used to compute the General health score. It is a weighted average of four different columns:

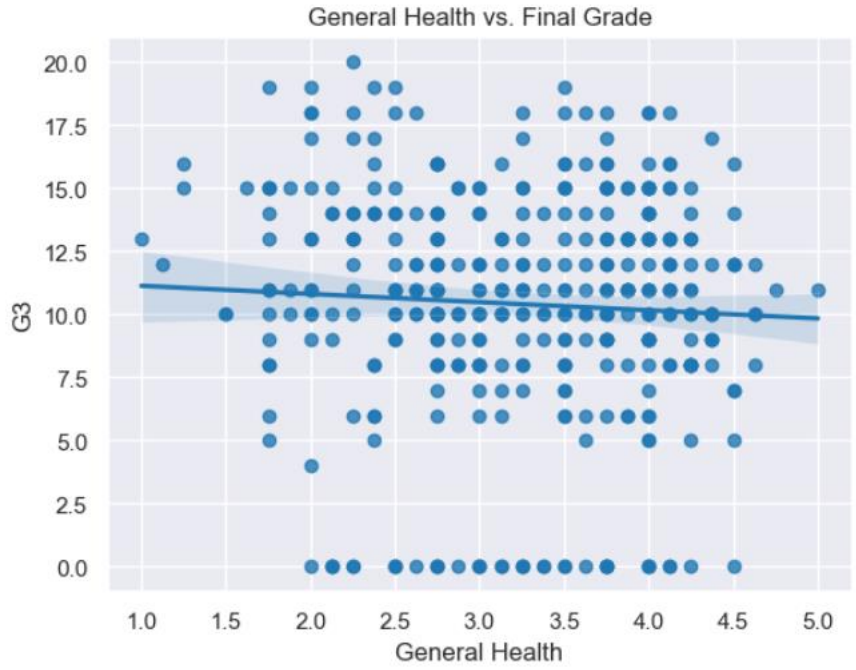
- 0.5 * df['Dalc']: This is half the value of the "Dalc" column.
- 0.5 * df['Walc']: This is half the value of the "Walc" column.
- 2 * df['health']: This is twice the value of the "health" column.
- df['famrel']: This is the value of the "famrel" column.

All these values are summed up, and then divided by 4 to obtain the average. The weights assigned to "Dalc" and "Walc" are 0.5 each, "health" is weighted 2, and "famrel" has a weight of 1. These weights influence the contribution of each factor to the General health score. In summary, the code is creating a new column in the DataFrame, "General Health," and populating it with a calculated score based on a weighted average of selected columns related to daily and weekend alcohol consumption, personal health, and family relationships.

```
df['General Health'] = (0.5 * df['Dalc'] + 0.5 * df['Walc'] + 2 * df['health'] + df['famrel']) / 4
```

```
sns.regplot(x='General Health', y='G3', data=df).set(title = 'General Health vs. Final Grade')
```

```
[Text(0.5, 1.0, 'General Health vs. Final Grade')]
```



A higher General health value typically translates into a lower final grade. On the other hand, a lower total health score typically translates into a higher final grade.
The correlation between G3 and General Health is low, as indicated by the slope of the best fit line being slightly off zero.

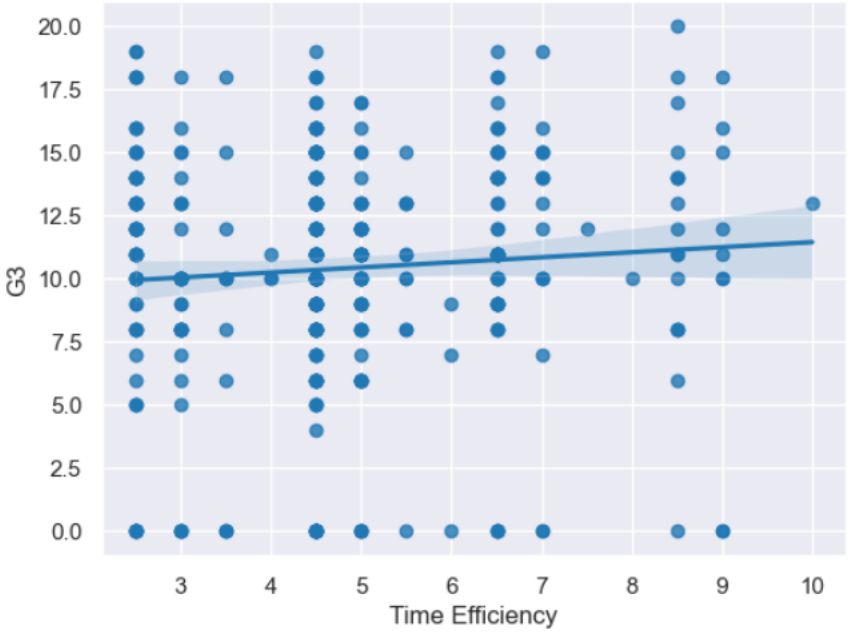
Time Efficiency vs Final Grade

We created a new column in the DataFrame called 'Time Efficiency.' The values in this column are computed using a formula where:

- 0.5 * df['traveltime']: Half the value of the 'traveltime' column.
- 2 * df['studytime']: Twice the value of the 'studytime' column.

```
df['Time Efficiency'] = 0.5 * df['traveltime'] + 2 * df['studytime']
sns.regplot(x='Time Efficiency', y='G3', data=df)
```

```
]: <Axes: xlabel='Time Efficiency', ylabel='G3'>
```



Encoding

Machine learning models generally work with numerical data, so categorical data needs to be encoded or transformed. The string data distribution aids in selecting the best encoding method (e.g., one-hot encoding, dummy encoding).

For our project we used dummy encoding as shown below.

```
new_df = pd.get_dummies(df, columns=['school','sex','address','famsize','Pstatus',
                                     'Mjob','Fjob','reason','guardian','schoolsup',
                                     'famsup','paid','activities','nursery','higher',
                                     'internet','romantic'])
```

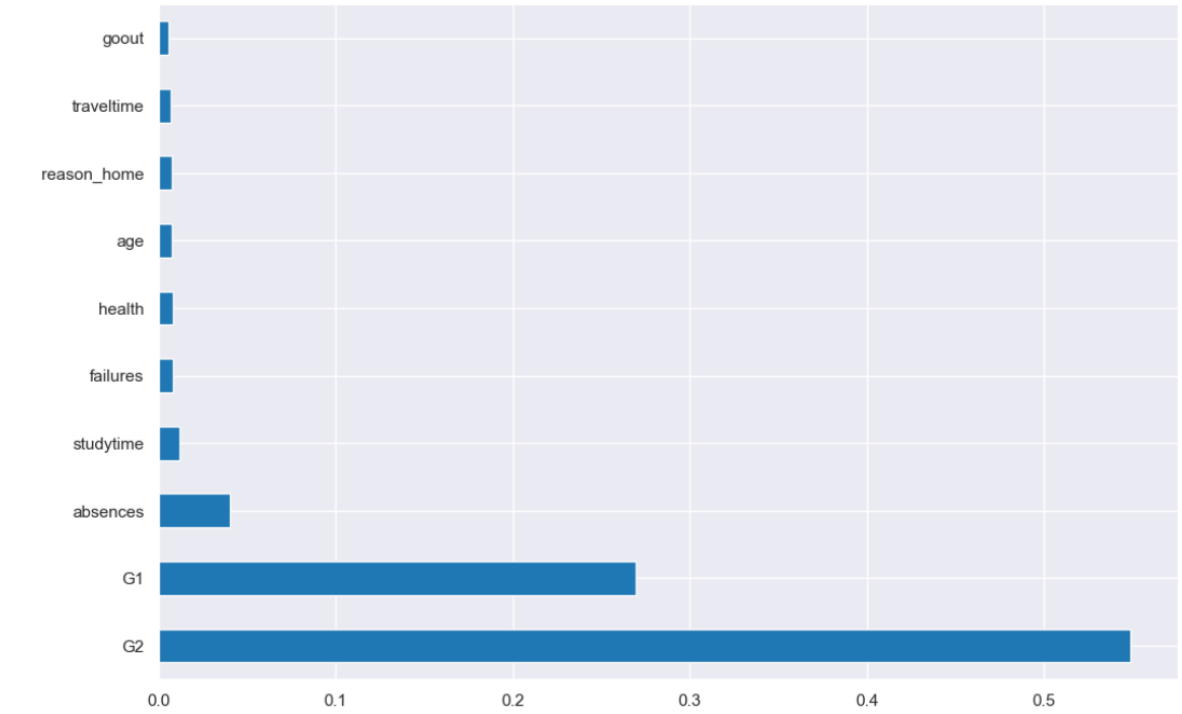
Feature Selection

We performed the feature importance analysis using the ExtraTreesRegressor, which is typically used for regression tasks, using the code mentioned below. An instance of the ExtraTreesRegressor model was initialized in this.

Using an ensemble approach, this algorithm generates a set of decision trees and produces the average prediction of each tree for regression tasks. Next, we attempt to fit the ExtraTreesRegressor model to the training data (X features and Y target variable) using the function "selection.fit(X, y)". In order to make predictions, the model must identify patterns and relationships in the data.

To summarize, the code snippet below uses the given data (X and y) to train an ExtraTreesRegressor model. A horizontal bar chart is then used to display the feature importance of the top 10 features. Understanding which features contribute most to the model's predictions can be gained from this analysis.

```
selection = ExtraTreesRegressor()
selection.fit(X, y)
plt.figure(figsize = (12,8))
f_importances = pd.Series(selection.feature_importances_, index=X.columns)
f_importances.nlargest(10).plot(kind='barh')
plt.show()
```



The top 3 features with high feature scores are G2, G1 and absences.

Using Correlation matrix and feature selection, to decrease the complexity of the models we chose 3 features namely G1, G2 and absences.

Data Scaling

Certain machine learning models exhibit sensitivity with respect to the numerical feature scale. When deciding which methods to use to bring features to a similar scale, such as normalization or standardization, distributions can be helpful.

Although G1 and G2 have the same units as G3, absences have different units which is number of days. So, scaling has to be done to eliminate bias in the model.

To scale the data we used StandardScaler()

```
pipeline = Pipeline([('std_scalar', StandardScaler())])
X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

Model Training and Testing

For this project, we chose three regression models.

Linear Regression

- Using the code above, we're attempting to create an instance of the LinearRegression class to determine how one or more independent variables and a dependent variable are related.
- We are training the data, or X_train as features and y_train as the target variable, using our linear regression model.
- Once more, we attempted to forecast the test set (X_test) and the training set (X_train) using test_pred and train_pred.
- Subsequently, we made an attempt to obtain the assessment metrics for both the training and test sets. This uses the "print_evaluate" function, which computes and prints a variety of evaluation metrics based on the true values (y_test and y_train) and the predicted values (test_pred and train_pred).
- These measurements could be R-squared (R2), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).
- Finally, we are creating a Data Frame to store the evaluation results of the model.

Polynomial Regression (Degree=2)

- PolynomialFeatures(degree=2) generates a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. In this case, we are transforming the original features into quadratic features.
- fit_transform(X_train) and transform(X_test) are used to fit the training data and then transform it into quadratic features. The same transformation is applied to the test data.
- Finally, we are updating the results Data Frame with evaluation results of the model.

Random Forest Regression

- RandomForestRegressor (n_estimators=1000) creates a Random Forest Regressor model with 1000 trees.
- fit(X_train, y_train) trains the model using the training data and the corresponding labels.
- predict(X_test) and predict(X_train) are used to make predictions on the test and train data respectively.
- Finally, we are updating the results Data Frame with evaluation results of the model.

Results and Conclusion

	Model	MAE	MSE	RMSE	R2 Square
0	Linear Regression	1.236817	5.255073	2.292395	0.774776
1	Polynomail Regression	1.279323	5.064792	2.250509	0.782931
2	Random Forest Regressor	1.050551	3.391012	1.841470	0.854667

The image displays a table with results for three different regression models:

Linear Regression, Polynomial Regression, and Random Forest Regressor.

These models have been evaluated based on four metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2 Square).

Based on the provided data:

- **Random Forest Regressor** appears to be the best performing model among the three. It has the lowest MAE (1.046769), MSE (3.368477), and RMSE (1.835341), which indicates that on average, its predictions are closest to the actual values. Furthermore, it has the highest R2 score (0.855633), suggesting that it explains a greater proportion of the variance in the dependent variable than the other models.
- **Linear Regression** has moderate performance with MAE, MSE, RMSE, and R2 values of 1.236817, 5.255073, 2.292395, and 0.774776 respectively. It performs better than Polynomial Regression in terms of these metrics except for RMSE, where it is slightly higher.
- **Polynomial Regression** has the highest errors with MAE of 1.279323 and MSE of 5.064792 but a slightly lower RMSE (2.250509) than Linear Regression. Its R2 score is 0.782931, which is lower than that of the Random Forest Regressor but close to Linear Regression.

In conclusion, the Random Forest Regressor model is the most accurate and reliable for predicting the dataset's outcomes based on the provided metrics. It would be recommended for use in practical applications requiring predictive modeling from the given options.