STUDENT PERFORMANCE IN SECONDARY EDUCATION PREDICTION

Group-10
Bhavans Vivekananda College

Presented by :
Syed Danish | Vaishali Vadde | Pravallika | Shravya

Abstract

The project focuses on developing a model that accurately predicts student performance in secondary education, leading to more effective teaching strategies and improved academic outcomes.

The study explores machine learning algorithms like Linear Regression, Decision Tree, Random Forest, K-Nearest Neighbours, Support Vector Machines, Bagging, and Boosting to predict student performance based on various academic and socio-demographic factors.

Objective

To identify the most suitable machine learning model to predict student performance in secondary education, enabling educators to tailor interventions and enhance academic success.

Content

Introduction

Education is fundamental to societal development, shaping the future workforce and driving long-term growth, much like how vital industries, such as steel, contribute to economic progress.

Recent technological advancements, including automation and data-driven insights, are revolutionizing education by enabling more personalized, efficient teaching strategies and improving overall student outcomes.

Machine learning is a transformative tool in data science Machine learning helps predict student performance and improve educational outcomes.

By analyzing academic and socio-demographic data, institutions can address student underperformance with targeted interventions.

The goal of developing predictive models is to enhance educational practices, making the learning environment more efficient and effective for students and educators alike.

Data preprocessing

Data

Dataset : Our Dataset consists of 33 variables and 649 records

Source : https://archive.ics.uci.edu/dataset/320/student+performance

Variables :

	Categorical Variables	Continuous Variables
0	school	age
1	sex	Medu
2	address	Fedu
3	famsize	traveltime
4	Pstatus	studytime
5	Mjob	failures
6	Fjob	freetime
7	reason	goout
8	guardian	Dalc
9	schoolsup	Walc
10	famsup	health
11	paid	absences
12	activities	G1
13	nursery	G2
14	higher	G3
15	internet	None
16	romantic	None

1	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob
2	GP	F	18	U	GT3	A	4	4	at_home	teache
3	GP	F	17	U	GT3	T	1	1	at_home	other
4	GP	F	15	U	LE3	T	1	1	at_home	other
5	GP	F	15	U	GT3	T	4	2	health	service
6	GP	F	16	U	GT3	T	3	3	other	other
7	GP	M	16	U	LE3	T	4	3	services	other
8	GP	M	16	U	LE3	T	2	2	other	other
9	GP	F	17	U	GT3	A	4	4	other	teache
10	GP	M	15	U	LE3	A	3	2	services	other
11	GP	M	15	U	GT3	T	3	4	other	other
12	GP	F	15	U	GT3	T	4	4	teacher	health
13	GP	F	15	U	GT3	T	2	1	services	other
14	GP	M	15	U	LE3	T	4	4	health	service
15	GP	M	15	U	GT3	T	4	3	teacher	other
16	GP	M	15	U	GT3	A	2	2	other	other
17	GP	F	16	U	GT3	T	4	4	health	other
18	GP	F	16	U	GT3	T	4	4	services	service
19	GP	F	16	U	GT3	T	3	3	other	other
20	GP	M	17	U	GT3	T	3	2	services	service
21	GP	M	16	U	LE3	T	4	3	health	other

Data Cleaning

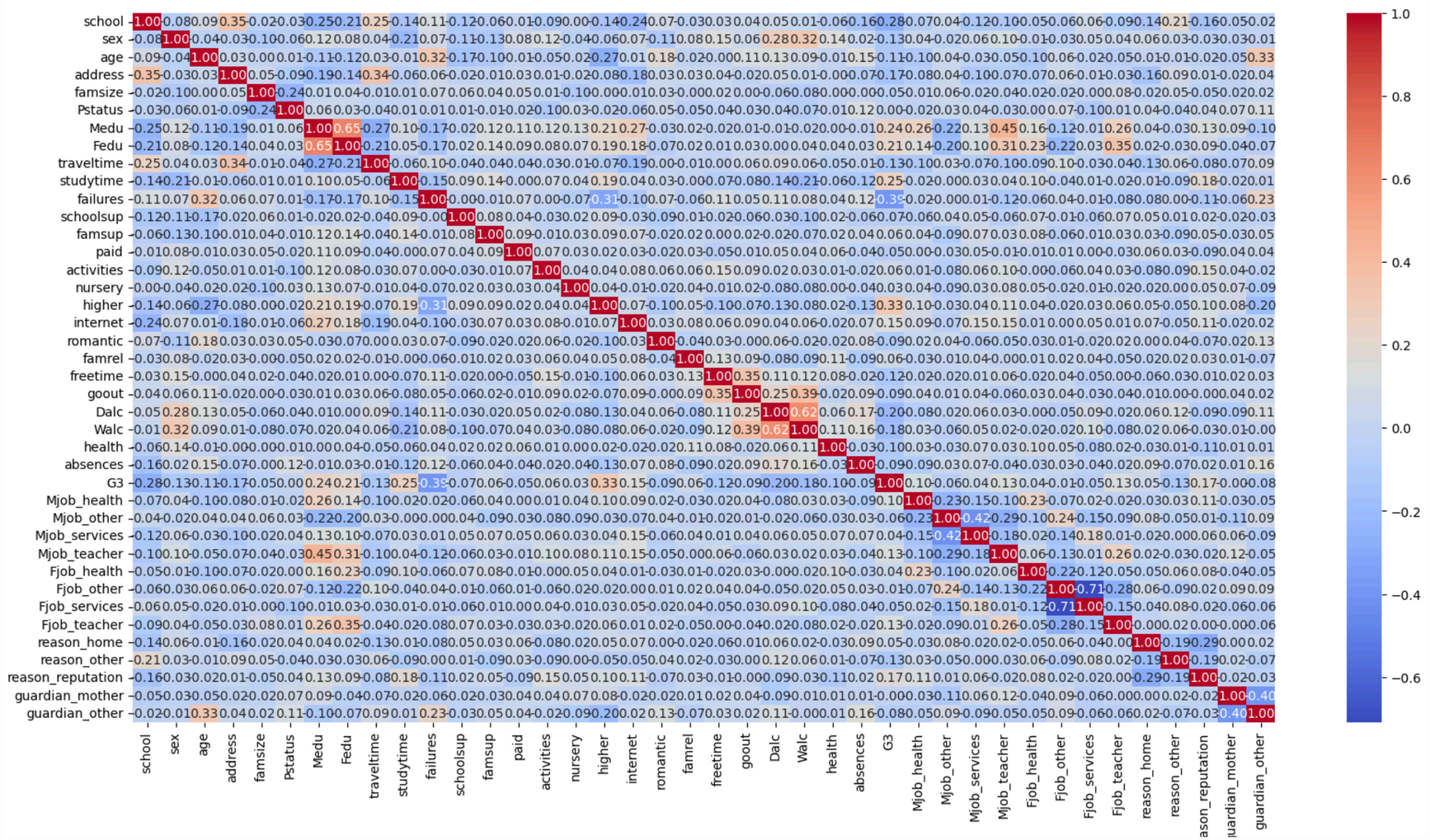
Data Cleaning :

Data cleaning is the process of identifying and correcting or removing inaccurate, incomplete, or irrelevant data from a dataset .The process performed for data cleaning are as follows:

- Identify missing values and decide on strategies . After searching for missing values we got to know that there are no missing values in our data
- Removed two variavles G1 and G2
- Dummified variables like occupation etc.

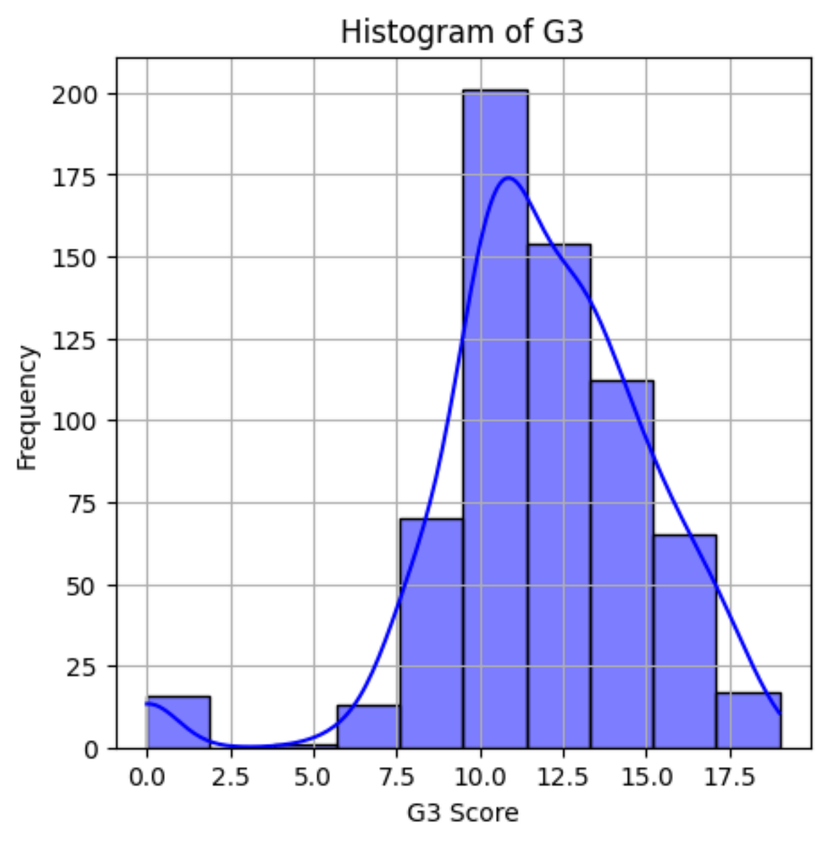
EDA - Exploratory Data Analysis

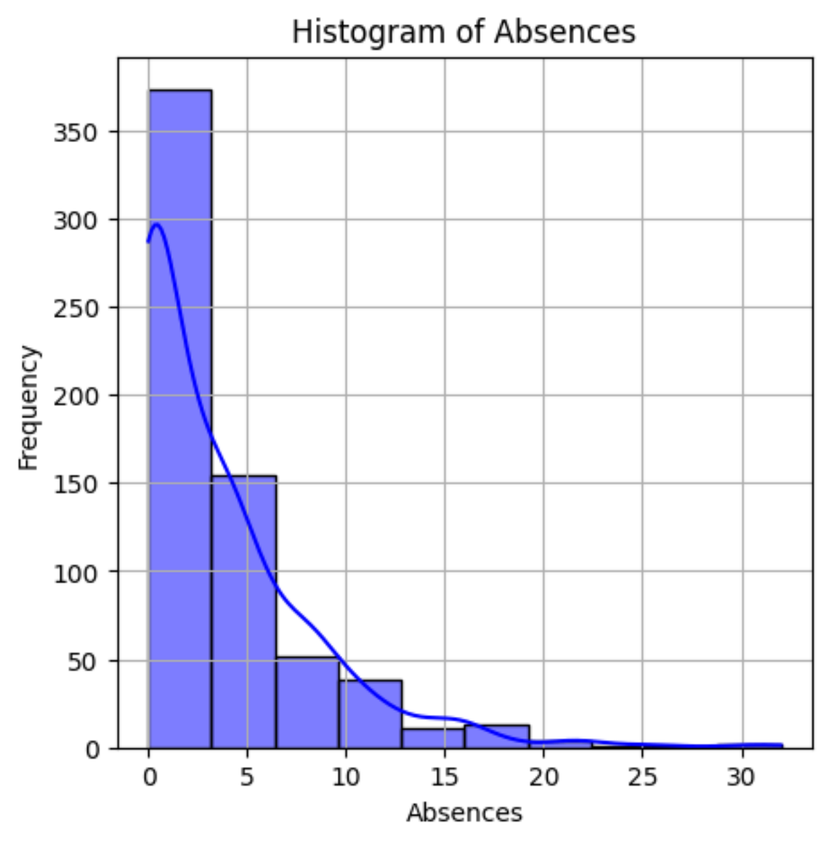
CORELATION HEATMAP



This visualizes the relationships between numerical features. It highlights correlations, helping identify features that might be redundant or strongly associated with churn

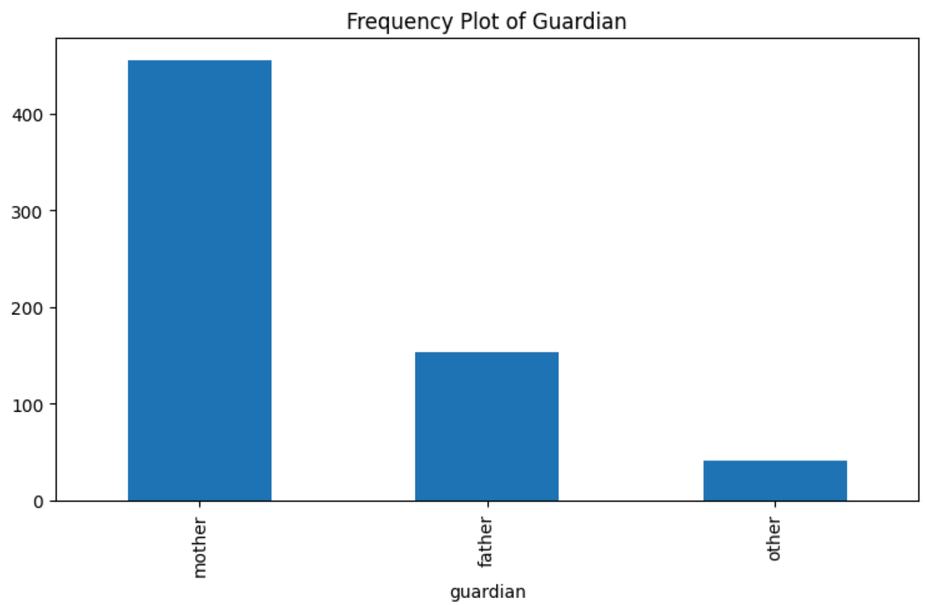
HISTOGRAM

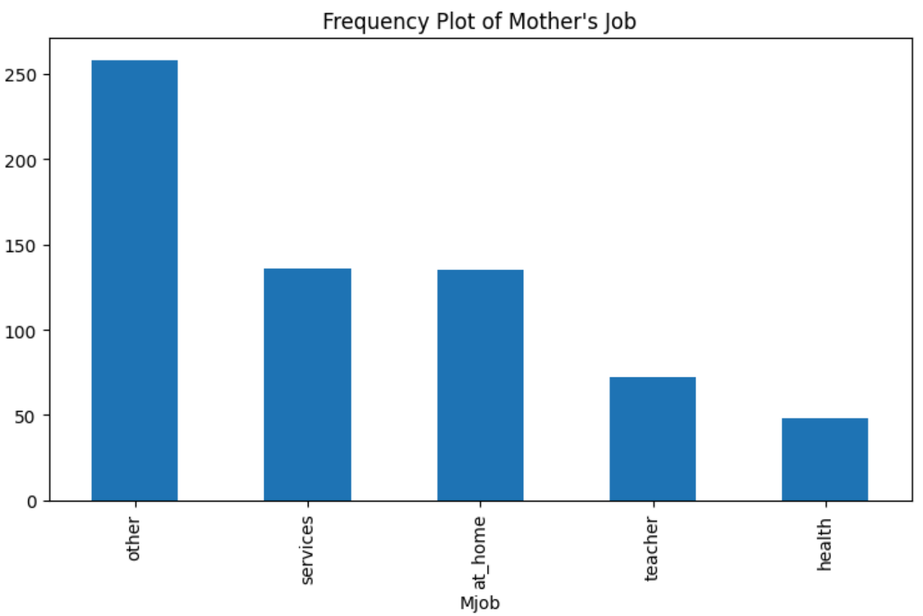


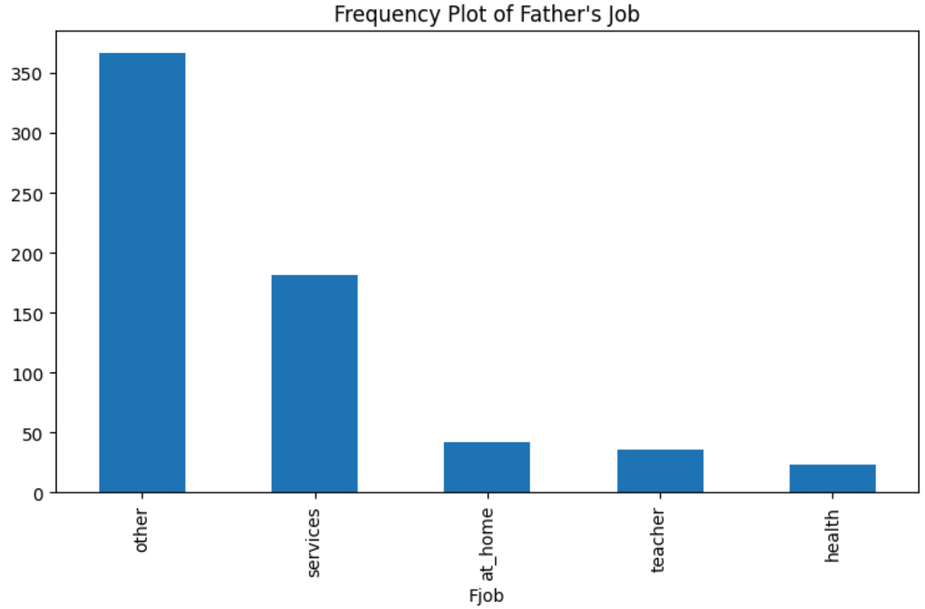


Histograms show the distribution of variables, helping us understand the frequency of different values for each feature.

COUNT PLOT

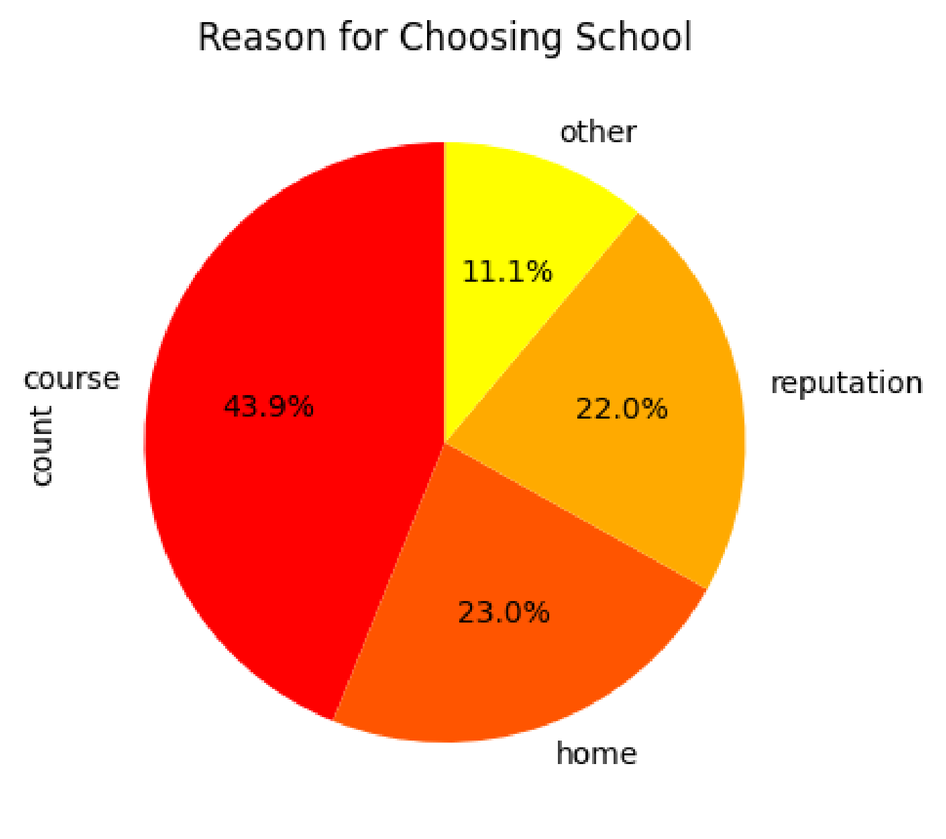


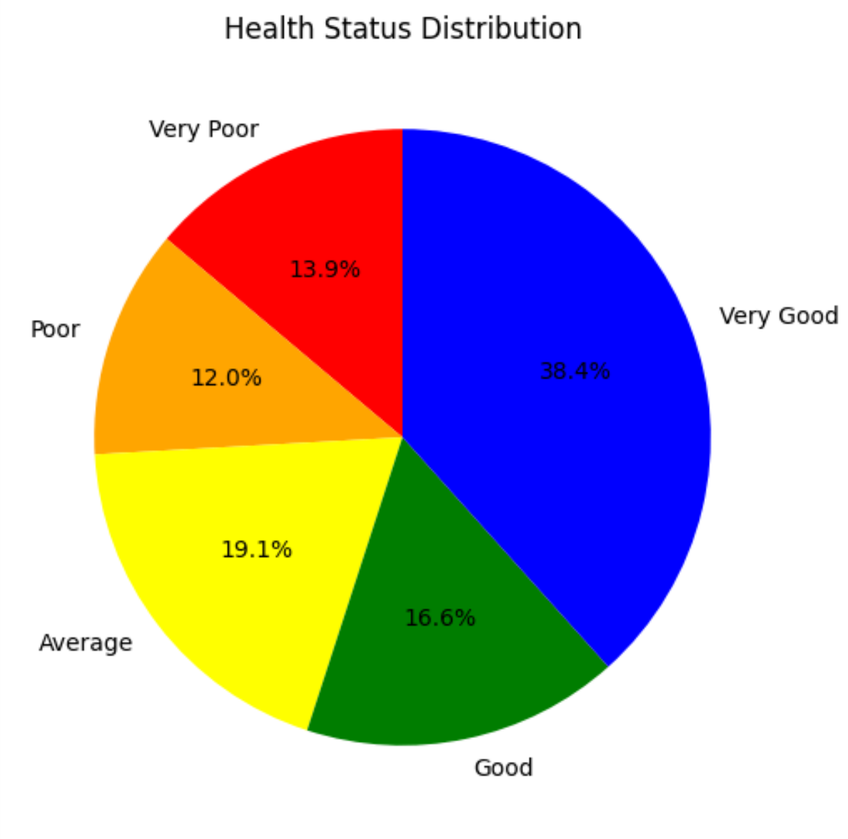




Count plots give an idea of the categorical variables and their distribution, showing, for example, the balance between churned and retained customers.

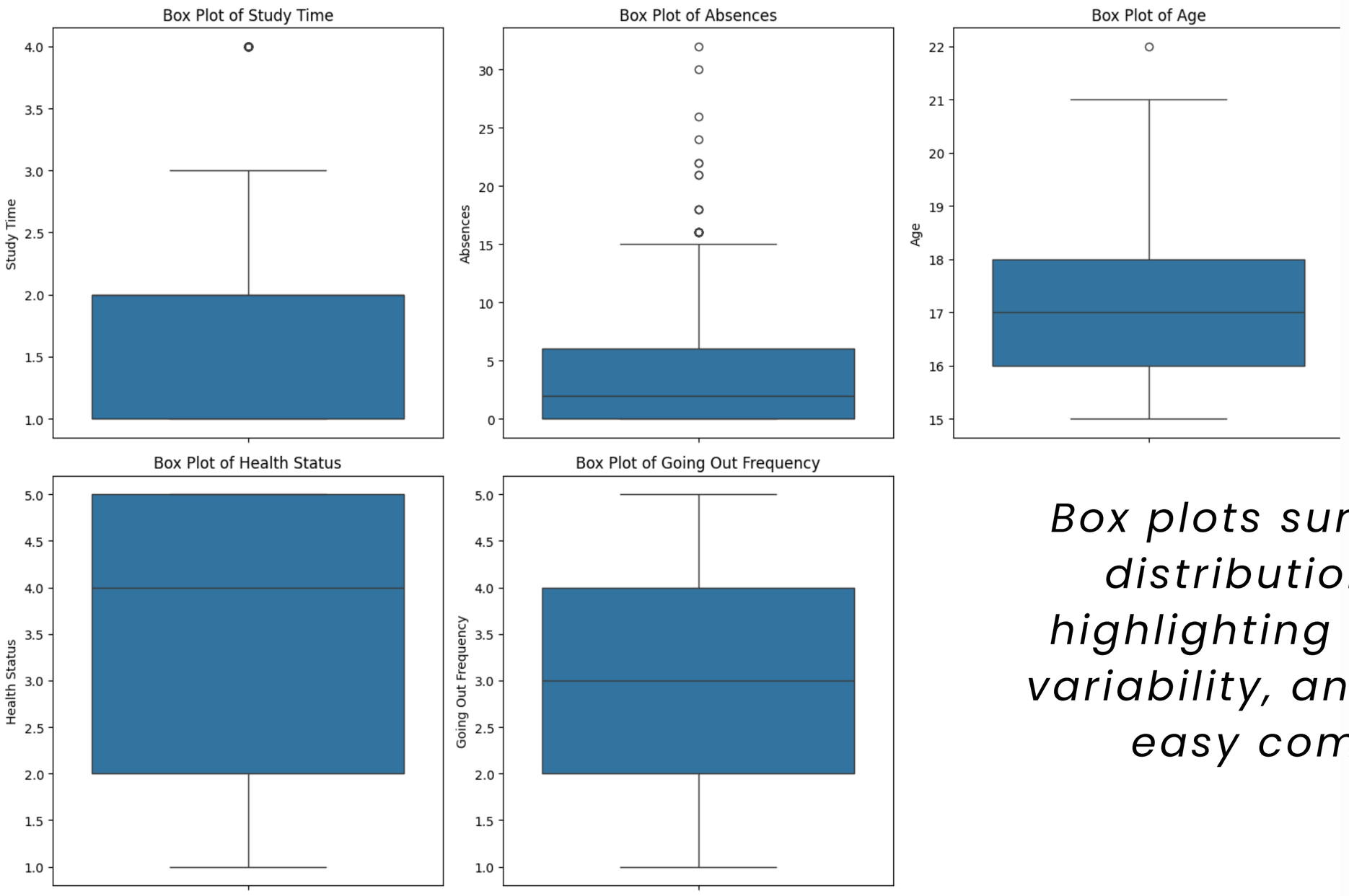
PIE PLOT





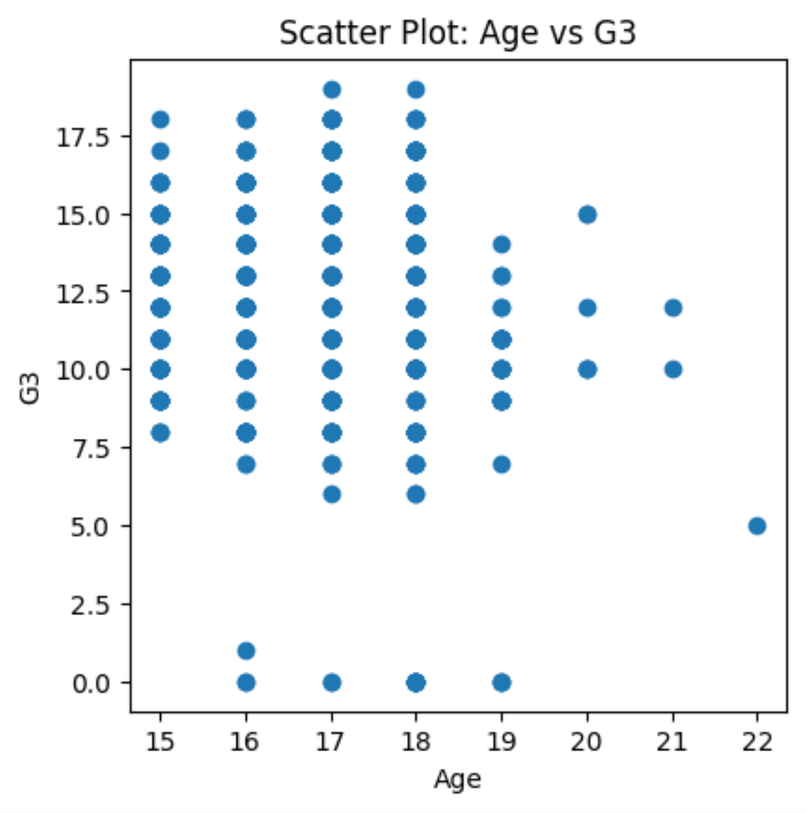
Pie plots visually break down proportions of a variable.

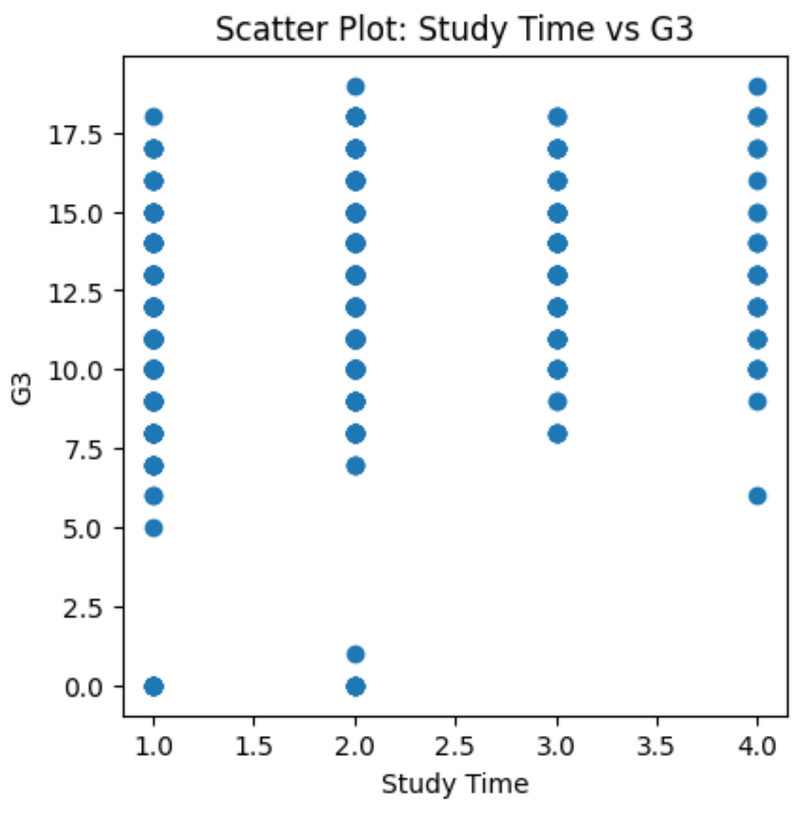
BOX PLOT

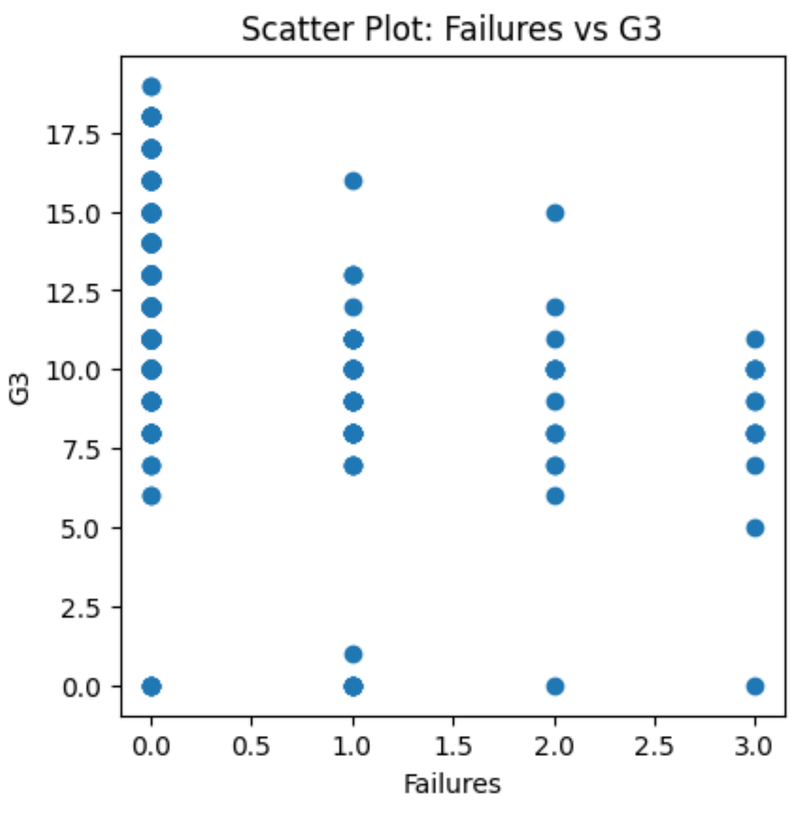


Box plots summarize the distribution of data, highlighting the median, variability, and outliers for easy comparison.

SCATTER PLOT







Multicollinearity check

variables with vif > 3.5 has been removed

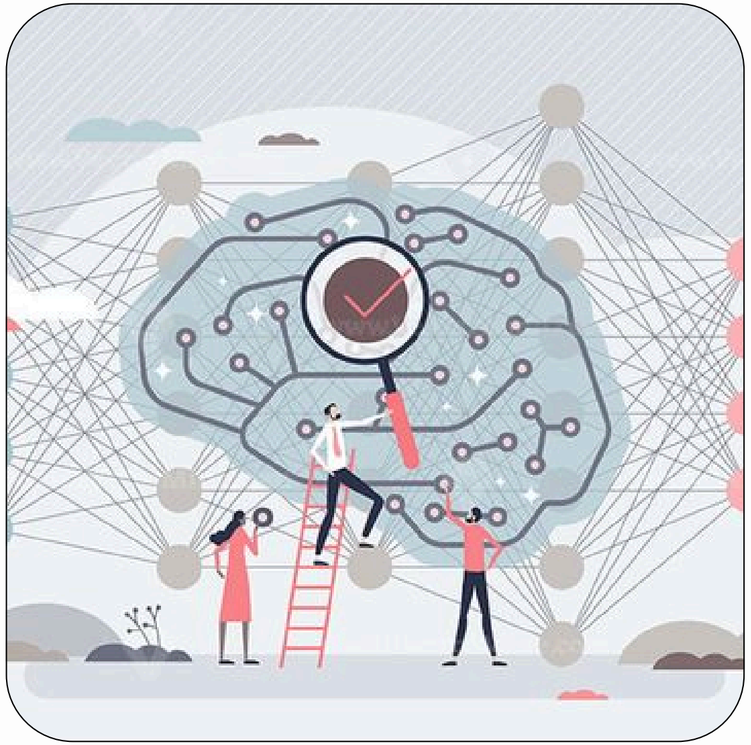
	variables	VIF
0	school	2.3
1	sex	2.3
2	age	69.7
3	address	1.9
4	famsize	3.8
5	Pstatus	1.3
6	Medu	15.9
7	Fedu	11.3
8	traveltime	6.8
9	studytime	7.9
10	failures	1.5
11	schoolsup	1.2
12	famsup	2.9
13	paid	1.2
14	activities	2.2
15	nursery	5.4
16	higher	11.3
17	internet	5.3
18	romantic	1.8
19	famrel	19.7
20	freetime	12.5
21	goout	12.1
22	Dalc	6.7
23	Walc	8.8
24	health	7.9
25	absences	2.0
26	Mjob_health	1.9
27	Mjob_other	3.3
28	Mjob_services	2.7
29	Mjob_teacher	2.6
30	Fjob_health	1.8
31	Fjob_other	10.4
32	Fjob_services	5.6
33	Fjob_teacher	2.3
34	reason_home	1.7
35	reason_other	1.4
36	reason_reputation	1.8
37	guardian_mother	4.5
38	guardian_other	1.6

Age has higher vif

	variables	VIF
0	school	1.9
1	sex	1.8
2	address	1.7
3	famsize	3.3
4	Pstatus	1.3
5	failures	1.3
6	schoolsup	1.2
7	famsup	2.5
8	paid	1.1
9	activities	2.1
10	romantic	1.7
11	absences	1.8
12	Mjob_health	1.4
13	Mjob_other	2.6
14	Mjob_services	2.1
15	Mjob_teacher	1.7
16	Fjob_health	1.2
17	Fjob_services	1.5
18	Fjob_teacher	1.2
19	reason_home	1.6
20	reason_other	1.3
21	reason_reputation	1.6
22	guardian_mother	3.5
23	guardian_other	1.4

These are the variables with vif < 3.5

Machine Learning Algorithms



Machine Learning Algorithms

Logistic Regression:
A classification algorithm that models the probability of a binary outcome using a logistic function. It predicts probabilities and applies a threshold to classify data points.

K-Nearest Neighbors (KNN):
A non-parametric algorithm that classifies data based on the majority class of the nearest neighbors. It calculates the distance between data points to make predictions.

Support Vector Machine (SVM):
A classification algorithm that finds the optimal hyperplane to separate classes in a high-dimensional space. It uses support vectors to maximize the margin between classes.

Decision Tree:
A tree-like model that splits the data into branches based on feature values to predict outcomes. Each internal node represents a decision, and each leaf node represents a final class or value.

Machine Learning Algorithms

Random Forest:
An ensemble learning method that creates multiple decision trees and aggregates their results to improve accuracy. It reduces overfitting by averaging the predictions of individual trees.

AdaBoost:
An ensemble technique that combines weak classifiers to form a strong classifier by assigning higher weights to misclassified instances. It iteratively adjusts the weights to improve prediction accuracy.

XGBoost:
A gradient boosting algorithm that improves model performance by combining multiple weak learners. It is optimized for speed and efficiency, often used for structured/tabular data.

MULTIPLE REGRESSION

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.193	1.682	0.177	1.712
72-25	0.225	1.834	0.205	1.854
70-30	0.274	1.867	0.249	1.905
60-40	0.171	2.130	0.164	2.164

K NEAREST NEIGHBOUR

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.90	1.934	0.006	1.955
72-25	0.123	2.058	0.045	2.169
70-30	0.146	2.111	0.061	2.273
60-40	0.121	2.291	0.066	2.395

SUPPORT VECTOR MACHINE

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.164	1.80	0.145	1.735
72-25	0.153	1.974	0.176	1.886
70-30	0.168	2.061	0.225	1.928
60-40	0.159	2.196	0.142	2.175

DECISION TREE

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	-0.825	2.731	-0.608	2.512
72-25	-0.470	2.607	-0.465	2.620
70-30	-0.44	2.764	-0.518	2.787
60-40	-0.391	2.877	-0.284	2.746

RANDOM FOREST

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.197	1.731	0.155	1.821
72-25	0.228	1.921	0.164	2.022
70-30	0.246	2.00	0.198	2.135
60-40	0.228	2.081	0.193	2.239

XG BOOST

SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	-0.077	2.012	-0.039	2.083
72-25	0.080	2.103	0.033	1.976
70-30	0.057	2.149	0.039	2.246
60-40	0.110	2.218	0.071	2.103

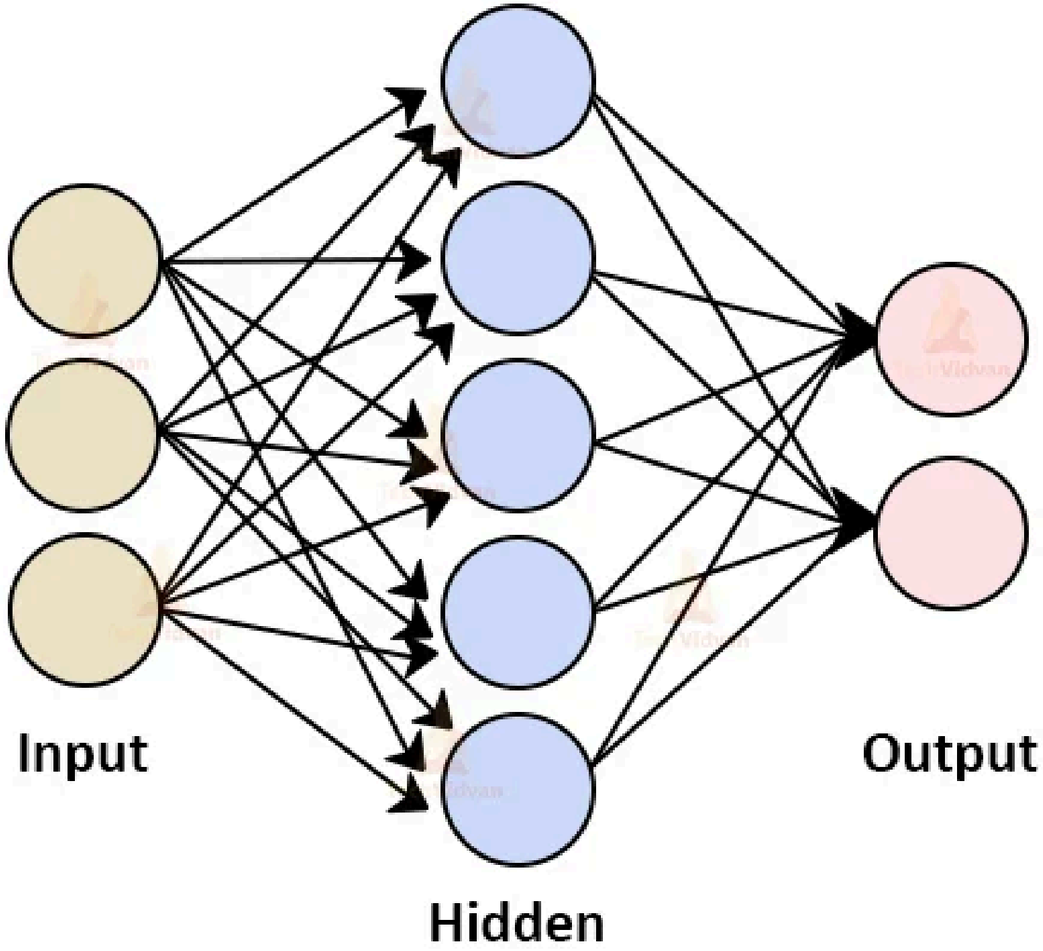
XG BOOST

SPILT	r^2_score (before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (Before vif)
80-20	0.139	1.855	0.017	2.435
72-25	0.233	1.857	0.113	2.195
70-30	0.195	2.008	-0.034	2.489
60-40	0.162	2.138	0.160	2.307

ADAPTIVE BOOSTING

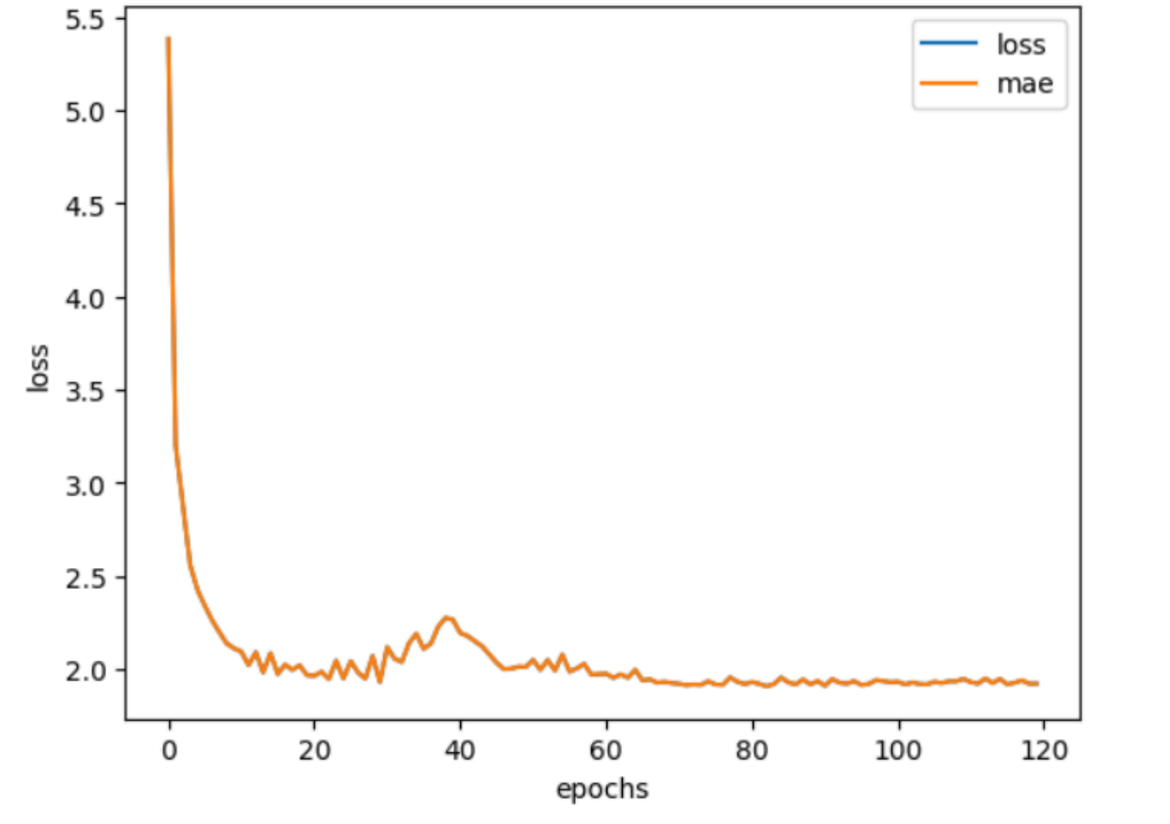
SPILT	r^2_score (Before vif)	MAE (Before vif)	r^2_score (After vif)	MAE (After vif)
80-20	0.073	1.871	0.206	1.767
72-25	0.178	2.022	0.208	1.930
70-30	0.220	2.008	0.235	2.020
60-40	0.228	2.081	0.284	2.132

Architecture of Artificial Neural Network



Train test	Architecture	Optimizer	Epochs	MAE
80-20	30-25-20-15-10-5-1	Adam	150	1.908
80-20	24-23-22-21-20-10-1	Adam	120	2.011
75-25	40-30-20-10-1	Adam	120	1.884
75-25	30-20-15-10-5-3-1	Adam	120	2.036
70-30	60-30-20-15-10-5-3-1	Adam	120	1.934
70-30	60-30-20-15-10-5-3-1	Adam	120	2.002
60-40	30-25-15-10-5-1	Adam	120	1.898
60-40	24-23-22-21-20-10-1	Adam	120	2.046

Neural Network Plot



Train Test Split	**75-25**
Architecture	**40-30-20-10-1**
Optimizer	**Adam**
Epochs	**120**

SUMMARY

1. Best Data Split:
   - The 80-20 split performs the best overall as it achieves the lowest Mean Absolute Error (MAE) across most models, making it the most reliable split for evaluation.
2. Best Performing Model:
   - The Adaptive Boosting algorithm is the best model with the lowest MAE of 1.767 on the 80-20 split.
   - This indicates that Adaptive Boosting can effectively predict student performance compared to other algorithms.
3. Comparison of Models:
   - Top 3 models based on MAE (after VIF correction):
     a. Adaptive Boosting: MAE = 1.767 (80-20 split).
     b. Support Vector Machine (SVM): MAE = 1.735 (80-20 split).
     c. Random Forest: MAE = 1.821 (80-20 split).
   - Other models, like Decision Tree and XGBoost, performed poorly with higher MAE and lower r^2 scores.
4. Before VIF vs After VIF:
   - The results after VIF correction were used to identify the best model.
   - VIF (Variance Inflation Factor) helped reduce multicollinearity, slightly improving or stabilizing the r^2 scores and maintaining accuracy.

FUTURESCOPE

1. Incorporating Diverse Data Sources: Future iterations of this project can include external factors such as school infrastructure, extracurricular participation, and neighborhood socio-economic conditions. This would enhance the model's ability to generalize and provide more accurate predictions.
2. Personalized Educational Interventions: The model can be integrated into school systems to provide personalized feedback and recommendations for students. For instance, students at risk of poor performance can receive tailored learning plans or additional support.
3. Real-Time Monitoring: Extending the model to work with real-time data, such as ongoing assignment grades, attendance, and behavioral indicators, can enable dynamic updates to performance predictions and timely interventions.
4. Cross-Cultural and Global Application: Expanding the dataset to include students from diverse cultural and geographical backgrounds will test the model’s robustness and scalability, making it applicable on a global scale.

INSIGHTS

- Key Determinants of Student Performance: Variables like parental education levels, study time, etc. appear to have a significant influence on the final grade (G3). Identifying these critical predictors can help educators prioritize resources effectively.
- Impact of Social and Lifestyle Factors: Features such as family relationships, freetime, and alcohol consumption (Dalc, Walc) reveal the importance of balancing academic and personal well-being for better performance outcomes.
- Value of Parental Involvement: Higher parental education levels and consistent cohabitation status correlate positively with student performance, underscoring the importance of a supportive home environment.
- Room for Continuous Improvement: While the model predicts final grades, incorporating qualitative aspects like motivation, emotional health, and teacher feedback could improve its interpretability and accuracy.

THANK YOU

Colab Notebook

Done by:
Vaishali Vadde | Syed Danish | Shravya | Pravalika

APPENDIX

UPLOADING DATA SET

```
from google.colab import files
uploaded = files.upload()
```

```
df=pd.read_csv('/content/student-por.csv')
df
```

CHECKING FOR NULL VALUES

```
df.isnull().sum()
```

```
df.info()
```

```
df.shape
```

```
df.describe()
```

```
df=df.drop('G1',axis=1)
df=df.drop('G2',axis=1)
```

```
df.columns
```

```
data_with_dummies = pd.get_dummies(df, columns=['Mjob', 'Fjob', 'reason', 'guardian'], drop_first=True)
data_with_dummies=data_with_dummies.replace({True: 1, False: 0})
print("Data with Dummy Variables:",data_with_dummies)
```

```
data_with_dummies.info()
```

```
data_with_dummies.iloc[:,7].value_counts()
```

```
for i in range(data_with_dummies.shape[1]):
  print(data_with_dummies.iloc[:,i].unique())
  print(data_with_dummies.iloc[:,i].value_counts())
```

```
data_with_dummies.iloc[:,7].value_counts()
```

```
for i in range(data_with_dummies.shape[1]):
   print(data_with_dummies.iloc[:,i].value_counts())
```

```
print(data_with_dummies)
```

EXPLORATORY DATA ANALYSIS

```
plt.figure(figsize=(5, 5))
sns.histplot(data_with_dummies['G3'], bins=10, kde=True, color='blue')
plt.title('Histogram of G3')
plt.xlabel('G3 Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

plt.figure(figsize=(5, 5))
sns.histplot(data_with_dummies['absences'], bins=10, kde=True, color='blue')
plt.title('Histogram of Absences')
plt.xlabel('Absences')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

```
corr_matrix =data_with_dummies.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm',fmt='.2f')
plt.show()
```

```
sns.countplot(data_with_dummies['G3'])
plt.title('Distribution of the Target Variable')
plt.show()
```

```python
X = df.drop(columns=['G3'])
y = df['G3']

# Display the shapes of X and y to ensure they are correct
print(X.shape)
print(y.shape)
```

```python
plt.figure(figsize=(10,10))
plt.subplot(2, 2, 1)
df['reason'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, cmap='autumn')
plt.title('Reason for Choosing School')
plt.show


health_labels = {1: 'Very Poor', 2: 'Poor', 3: 'Average', 4: 'Good', 5: 'Very Good'}

# Count the frequency of each health level
health_counts = df['health'].value_counts().sort_index()

# Create a pie chart with custom labels
plt.figure(figsize=(8, 6))
plt.pie(
    health_counts,
    labels=[health_labels[key] for key in health_counts.index],
    autopct='%1.1f%%',
    startangle=90,
    colors=['red', 'orange', 'yellow', 'green', 'blue']
)
plt.title('Health Status Distribution')
plt.show()
```

```
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 1)
plt.scatter(df['age'], df['G3'])
plt.xlabel('Age')
plt.ylabel('G3')
plt.title('Scatter Plot: Age vs G3')
plt.show()
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 2)
plt.scatter(df['studytime'], df['G3'])
plt.xlabel('Study Time')
plt.ylabel('G3')
plt.title('Scatter Plot: Study Time vs G3
plt.show()
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 3)
plt.scatter(df['failures'], df['G3'])
plt.xlabel('Failures')
plt.ylabel('G3')
plt.title('Scatter Plot: Failures vs G3')
plt.show()
plt.figure(figsize=(10, 10))
plt.subplot(2, 2, 4)
plt.scatter(df['absences'], df['G3'])
plt.xlabel('Absences')
plt.ylabel('G3')
plt.title('Scatter Plot: Absences vs G3')
```

```
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
df['guardian'].value_counts().plot(kind='bar')
plt.title('Frequency Plot of Guardian')

plt.subplot(2, 2, 2)
df['Mjob'].value_counts().plot(kind='bar')
plt.title('Frequency Plot of Mother\'s Job')

plt.subplot(2, 2, 3)
df['Fjob'].value_counts().plot(kind='bar')
plt.title('Frequency Plot of Father\'s Job')

plt.tight_layout()
plt.show()
```

```python
# Step: Box Plots for Selected Numerical Variables
plt.figure(figsize=(15, 10))
# Box plot for 'studytime'
plt.subplot(2, 3, 1)
sns.boxplot(y=df['studytime'])
plt.title('Box Plot of Study Time')
plt.ylabel('Study Time')


# Box plot for 'absences'
plt.subplot(2, 3, 2)
sns.boxplot(y=df['absences'])
plt.title('Box Plot of Absences')
plt.ylabel('Absences')

# Box plot for 'age'
plt.subplot(2, 3, 3)
sns.boxplot(y=df['age'])
plt.title('Box Plot of Age')
plt.ylabel('Age')

# Box plot for 'health'
plt.subplot(2, 3, 4)
sns.boxplot(y=df['health'])
plt.title('Box Plot of Health Status')
plt.ylabel('Health Status')

# Box plot for 'goout'
plt.subplot(2, 3, 5)
sns.boxplot(y=df['goout'])
plt.title('Box Plot of Going Out Frequency')
plt.ylabel('Going Out Frequency')

plt.tight_layout()
```

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import mean_squared_error
```

```
X = data_with_dummies.drop(columns=['G3'])
y=data_with_dummies['G3']
```

TRAIN TEST SPLIT

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.20, train_size=0.80,random_state=0)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.25, train_size=0.75,random_state=0)
X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.30, train_size=0.70,random_state=0)
X_train4, X_test4, y_train4, y_test4 = train_test_split(X, y, test_size=0.40, train_size=0.60,random_state=0)
```

80-20 SPLIT

```
lm=LinearRegression()
lm.fit(X_train1, y_train1)
y_pred1 =lm.predict(X_test1)
print("MAE:",metrics.mean_absolute_error(y_test1,y_pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

KNN

```
from sklearn.neighbors import KNeighborsRegressor
```

(80-20) SPLIT

```
model1=KNeighborsRegressor(n_neighbors=25)
model1.fit(X_train1, y_train1)
```

```
y_pred1 = model1.predict(X_test1)
y_pred1
```

```
knn1 = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
knn1
```

EVALUATION METRICS FOR(80-20) SPLITS

```
print("R2 SCORE:")
from sklearn.metrics import r2_score
r2_score(y_test1,y_pred1)
```

```
print("MAE:")
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test1,y_pred1)
```

```
print("MSE:")
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test1,y_pred1)
```

```
print("RMSE:")
mse = mean_squared_error(y_test1, y_pred1)
rmse = np.sqrt(mse)
rmse
```

```
from sklearn.metrics import mean_absolute_percentage_error
mape = mean_absolute_percentage_error(y_test1, y_pred1)
print(mape)
mape = mape * 100
```

```
plt.scatter(X_test1['absences'], y_test1, label='Test Data')
plt.scatter(X_test1['absences'], y_pred1, color='red', linewidth=1, label='KNN points')
plt.xlabel('absences')
plt.ylabel('G3')
plt.title('KNN Regression')
plt.legend()
plt.show()
```

SVM

```
from sklearn.svm import SVR
```

(80-20) SPLIT

```
model1 = SVR(kernel='rbf')
model1.fit(X_train1, y_train1)
```

```
y_pred1=model1.predict(X_test1)
```

```
svm1 = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
svm1
```

```
y_pred1=model1.predict(X_test1)
```

```
svm1 = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
svm1
```

EVALUATIN MATRICS (80-20) SPLITS

```
print("MAE:",metrics.mean_absolute_error(y_test1,y_pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

```
plt.scatter(X_test1['absences'], y_test1, label='Test Data')
plt.scatter(X_test1['absences'], y_pred1, color='red', linewidth=1, label='SVM points')
plt.xlabel('absences')
plt.ylabel('G3')
plt.title('SVM Regression')
plt.legend()
plt.show()
```

DECISION TREE

```
from sklearn.tree import DecisionTreeRegressor
```

(80-20)SPLITS

```
# Create Decision Tree classifer object
reg1 = DecisionTreeRegressor()

# Train Decision Tree Classifer
reg1 = reg1.fit(X_train1,y_train1)
```

```
#Predict the response for test dataset
y_pred1 = reg1.predict(X_test1)
print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))
```

EVALUATION METRIC (80-20) SPLIT

```
print("MAE:",metrics.mean_absolute_error(y_test1,y_pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

RANDOM FOREST

```
from sklearn.ensemble import RandomForestRegressor
```

(80-20) SPLIT

```
rf1 = RandomForestRegressor()
rf1.fit(X_train1, y_train1)
y_pred1=rf1.predict(X_test1)
```

EVALUATION METRICS (80-20) SPLIT

```
print("MAE:",metrics.mean_absolute_error(y_test1,y_pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

XGboost

```
import xgboost as xgb
```

(80-20) SPLIT

```
model1 = xgb.XGBRegressor()
model2 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)

train_model1 = model1.fit(X_train1, y_train1)
train_model2 = model2.fit(X_train1, y_train1)
```

```
pred1 = train_model1.predict(X_test1)
pred2 = train_model2.predict(X_test1)
```

EVALUATION METRICS (80-20) SPLIT

```
print("MAE:",metrics.mean_absolute_error(y_test1,pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1,pred1)))
print("R2 score:",metrics.r2_score(y_test1,pred1))
print("MAE:",metrics.mean_absolute_error(y_test1,pred2)) # Use pred2 instead of y_pred2
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, pred2))) # Use pred2 instead of y_pred2
print("R2 score:",metrics.r2_score(y_test1,pred2)) # Use pred2 instead of y_pred2
```

ADAboost

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor

# Replace 'base_estimator' with 'estimator'
base_estimator = DecisionTreeRegressor(max_depth=3, random_state=0)
```

(80-20) SPLIT

```
adaboost = AdaBoostRegressor(estimator=base_estimator, # Changed argument name he
                             n_estimators=3,random_state=0)
adaboost.fit(X_train1, y_train1)
```

```
y_pred1 = adaboost.predict(X_test1)
```

EVALUATION METRICS (80-20) SPLIT

```
print("MAE:",metrics.mean_absolute_error(y_test1,y_pred1))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test1, y_pred1)))
print("R2 score:",metrics.r2_score(y_test1,y_pred1))
```

ANN

```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
```

(80-20) SPLIT

```
# STEP1: Creating the model

model= tf.keras.Sequential([

                            tf.keras.layers.Dense(30),

                            tf.keras.layers.Dense(25),

                            tf.keras.layers.Dense(20),

                            tf.keras.layers.Dense(15),

                            tf.keras.layers.Dense(10),

                            tf.keras.layers.Dense(5),

                            tf.keras.layers.Dense(1)
])
```

```
# STEP2: Compiling the model   # optimizer can be SGD, Adam
# Set the learning rate when creating the Adam optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(loss= tf.keras.losses.mae,
              optimizer= optimizer,  # Use the optimizer with the specified learning rate
              metrics= ["mae"])

# STEP3: Fit the model
# Remove learning_rate from fit() arguments
history= model.fit(X_train1, y_train1, epochs= 150, verbose=1)
```

```
model.evaluate(X_test1, y_test1)
```

```
model.summary();
```

```
pd.DataFrame(history.history).plot()
plt.ylabel("loss")
plt.xlabel("epochs")
```

MULTICOLINEARITY

```
# Import library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
import pandas as pd # Make sure pandas is imported

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i).round(1) for i in range(X.shape[1])]

    return(vif)

calc_vif(X)
```

```
calc_vif(X.drop('age',axis=1))
calc_vif(X.drop(['age','famrel'],axis=1))
calc_vif(X.drop(['age','famrel','Medu'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu','Walc'],axis=1)
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu','Walc','health'],axis=1))
drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu','Walc','health','studytime'],axis=1))
calc_vif(X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu','Walc','health','studytime','traveltime'],axis=1))

x_nomulti = X.drop(['age','famrel','Medu','goout','higher','freetime','Fjob_other','Fedu','Walc','health','studytime','traveltime','nursery','internet','Dalc'], axis=1

x_nomulti_train1,x_nomulti_test1,y_nomulti_train1,y_nomulti_test1=train_test_split(x_nomulti,y,test_size=0.20,random_state=0)
x_nomulti_train2,x_nomulti_test2,y_nomulti_train2,y_nomulti_test2=train_test_split(x_nomulti,y,test_size=0.25,random_state=0)
x_nomulti_train3,x_nomulti_test3,y_nomulti_train3,y_nomulti_test3=train_test_split(x_nomulti,y,test_size=0.30,random_state=0)
x_nomulti_train4,x_nomulti_test4,y_nomulti_train4,y_nomulti_test4=train_test_split(x_nomulti,y,test_size=0.40,random_state=0)
```