INTRODUCTION TO DATA SCIENCE

Project report by: **Group 66**

STUDENT PERFORMANCE ANALYSIS WITH PYTHON USING STUDENT-MAT DATASET

- Pranjal Bansal 21UCS154
- Shiven Gupta 21UCS195
- Himanshi Keyal 21UCC051
- Prachi Bansal 21UCC076

Course Instructors:

- Dr. Lal Upendra Pratap Singh
- Dr. Aloke Dutta
- Dr. Subrat K Dash

Department of Computer Science and Engineering
The LNM Institute of Information Technology

TABLE OF CONTENTS

PROBLEM STATEMENT

We need to collect a dataset from the given website and perform the following steps:

1. Data pre-processing and its visualization
2. Explain all the inferences we got from our data.
3. Explain what ML Classification Algorithms are being used and why.
4. Implementing those algorithms.
5. Output the result of the testing set and its visualization.

All the tasks are performed with the help of pre-existing Python Libraries such as:

- scikit_learn: scikit_learn is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms.

- Seaborn: Seaborn is a library that uses Matplotlib underneath to plot graphs.

- NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays

and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension
  NumPy.

- Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis.

Importing initial libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import math
from tabulate import tabulate
import warnings
warnings.filterwarnings('ignore')
```

INTRODUCTION TO THE DATASET

This data approaches student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features) and it was collected by using school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por).

Note: The target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful.

Our dataset consists of the following:

- Data Set Characteristics: Multivariate
- Subject Area: Social Science
- Number of instances: 649
- Number of Attributes: 33
- Attribute characteristics: Categorical(binary),Integer
- Associated Tasks: Classification,Regression
- Missing Values: No
- Date Donated: 11-26-2014

Attribute 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** - reasons 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. **travel time** - 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. **study** - 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 (Math or 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)

These grades are related wit0.h the course subject, Math or Portuguese:

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)

Source of our dataset:
https://archive.ics.uci.edu/dataset/320/student+performance
(All the above information about the dataset is taken from this link)

DATA ANALYSIS

Importing the dataset

We use the read_csv() function of the Pandas library to import our dataset.

data.shape() describes the structure

```
df = pd.read_csv('student-mat.csv', delimiter=';')
print(df.shape)

# Display the first few rows of the DataFrame as a table
print(tabulate(df.head(), headers='keys', tablefmt='pretty'))

# Display the first 8 columns of the DataFrame as a table
df_subset = df.iloc[:, :8]
print(tabulate(df_subset.head(8), headers='keys', tablefmt='pretty'))
```

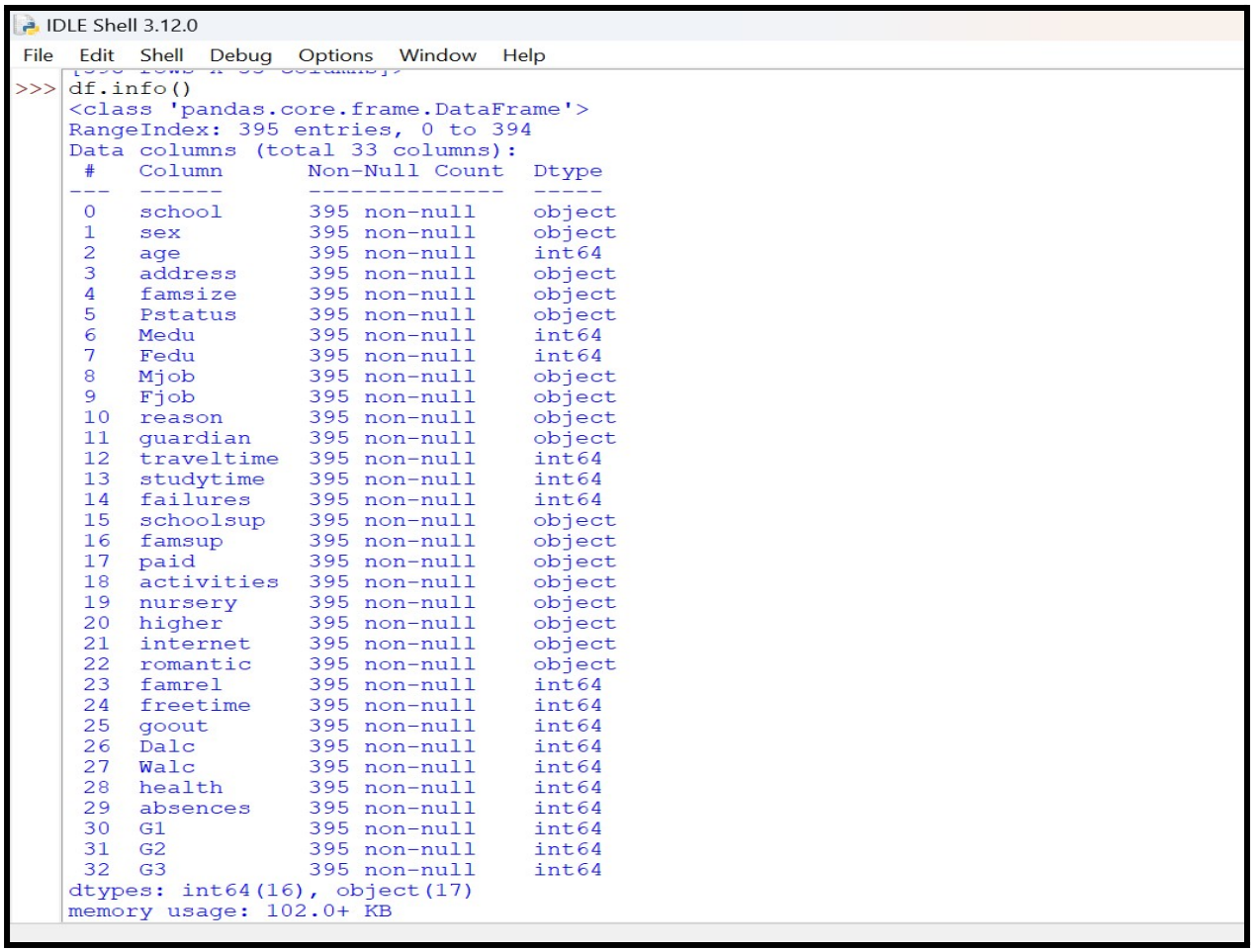
Output:

```
========== RESTART: C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py ==========
(395, 33)
+---+--------+-----+-----+---------+---------+---------+------+------+------------+----------+--------+------------+-----------+----------+---------+---+---+---+
|   | school | sex | age | address | famsize | Pstatus | Medu | Fedu |    Mjob    |   Fjob   | reason | guardian | traveltime | studytime | failures | schoo
lsup | famsup | paid | activities | nursery | higher | internet | romantic | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | G2 | G3 |
+---+--------+-----+-----+---------+---------+---------+------+------+------------+----------+--------+------------+-----------+----------+---------+---+---+---+
| 0 |   GP   |  F  | 18  |    U    |   GT3   |    A    |  4   |  4   | at_home | teacher | course |  mother |     2      |     2     |    0     |    ye
s    |   no   |  no |     no     |   yes   |   yes   |    no    |    no    |  4   |  3   |  4  |  1  |  1  |  3  |    6     | 5  | 6  | 6  |
| 1 |   GP   |  F  | 17  |    U    |   GT3   |    T    |  1   |  1   | at_home |  other  | course |  father |     1      |     2     |    0     |    no
    |  yes   |  no |     no     |    no   |   yes   |   yes    |    no    |  5   |  3   |  3  |  1  |  1  |  3  |    4     | 5  | 5  | 6  |
| 2 |   GP   |  F  | 15  |    U    |   LE3   |    T    |  1   |  1   | at_home |  other  | other  |  mother |     1      |     2     |    3     |    ye
s    |   no   | yes |     no     |   yes   |   yes   |   yes    |    no    |  4   |  3   |  2  |  2  |  3  |  3  |    10    | 7  | 8  | 10 |
| 3 |   GP   |  F  | 15  |    U    |   GT3   |    T    |  4   |  2   | health  | services |  home  |  mother |     1      |     3     |    0     |    no
    |  yes   | yes |     yes    |   yes   |   yes   |   yes    |   yes    |  3   |  2   |  2  |  1  |  1  |  5  |    2     | 15 | 14 | 15 |
| 4 |   GP   |  F  | 16  |    U    |   GT3   |    T    |  3   |  3   |  other  |  other  |  home  |  father |     1      |     2     |    0     |    no
    |  yes   | yes |     no     |   yes   |   yes   |    no    |    no    |  4   |  3   |  2  |  1  |  2  |  5  |    4     | 6  | 10 | 10 |
+---+--------+-----+-----+---------+---------+---------+------+------+------------+----------+--------+------------+-----------+----------+---------+---+---+---+
+---+--------+-----+-----+---------+---------+---------+------+------+
|   | school | sex | age | address | famsize | Pstatus | Medu | Fedu |
+---+--------+-----+-----+---------+---------+---------+------+------+
| 0 |   GP   |  F  | 18  |    U    |   GT3   |    A    |  4   |  4   |
| 1 |   GP   |  F  | 17  |    U    |   GT3   |    T    |  1   |  1   |
| 2 |   GP   |  F  | 15  |    U    |   LE3   |    T    |  1   |  1   |
| 3 |   GP   |  F  | 15  |    U    |   GT3   |    T    |  4   |  2   |
| 4 |   GP   |  F  | 16  |    U    |   GT3   |    T    |  3   |  3   |
| 5 |   GP   |  M  | 16  |    U    |   LE3   |    T    |  4   |  3   |
| 6 |   GP   |  M  | 16  |    U    |   LE3   |    T    |  2   |  2   |
| 7 |   GP   |  F  | 17  |    U    |   GT3   |    A    |  4   |  4   |
+---+--------+-----+-----+---------+---------+---------+------+------+
>>>
```

Information about the dataset:

Using the DataFrame.info( ), we find the information about our dataset, i.e., how many attributes are there, the datatype of each attribute, and whether there is any null value to it or not.

Output:



Looking at the datatypes of all the attributes present :

```
>>> df.dtypes
school         object
sex            object
age             int64
address        object
famsize        object
Pstatus        object
Medu            int64
Fedu            int64
Mjob           object
Fjob           object
reason         object
guardian       object
traveltime      int64
studytime       int64
failures        int64
schoolsup      object
famsup         object
paid           object
activities     object
nursery        object
higher         object
internet       object
romantic       object
famrel          int64
freetime        int64
goout           int64
Dalc            int64
Walc            int64
health          int64
absences        int64
G1              int64
G2              int64
G3              int64
dtype: object
>>>
```

Using DataFrame.describe( ).T to view basic statistical details about our dataset, such as min-max values, standard deviation, mean, etc.

Output:

```
>>> 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
```

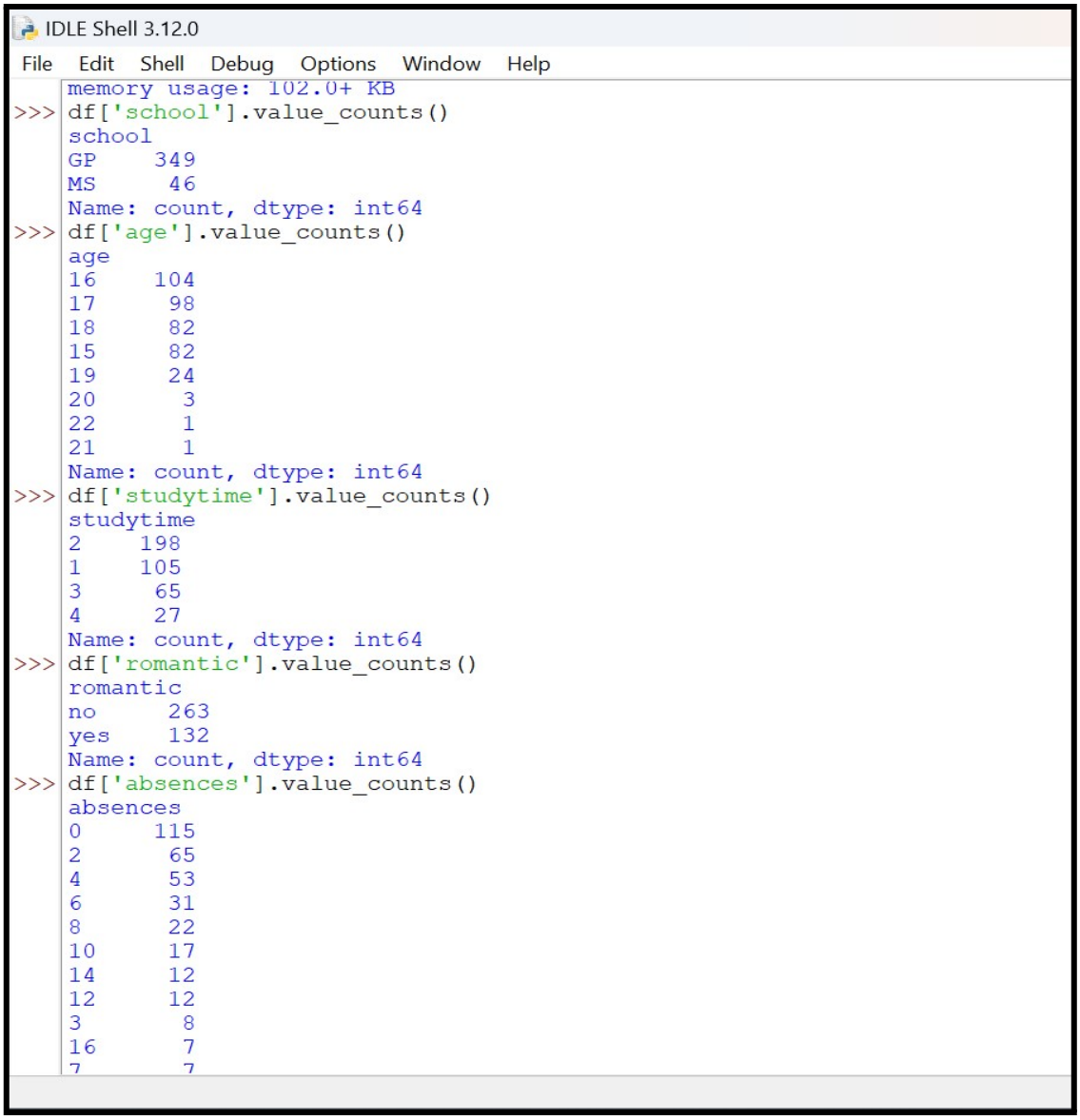
Using DataFrame.isnull() to check if there are any null values in the dataset or not.

Output:

```
>>> df.isnull()
     school    sex    age  address  ...  absences     G1     G2     G3
0     False  False  False    False  ...     False  False  False  False
1     False  False  False    False  ...     False  False  False  False
2     False  False  False    False  ...     False  False  False  False
3     False  False  False    False  ...     False  False  False  False
4     False  False  False    False  ...     False  False  False  False
..      ...    ...    ...      ...  ...       ...    ...    ...    ...
390   False  False  False    False  ...     False  False  False  False
391   False  False  False    False  ...     False  False  False  False
392   False  False  False    False  ...     False  False  False  False
393   False  False  False    False  ...     False  False  False  False
394   False  False  False    False  ...     False  False  False  False

[395 rows x 33 columns]
```

Using DataFrame.value_counts() to check different entries of each attribute

Output:



Checking for Class Imbalance: We check the distribution of the target variable to see if the dataset is balanced or not.

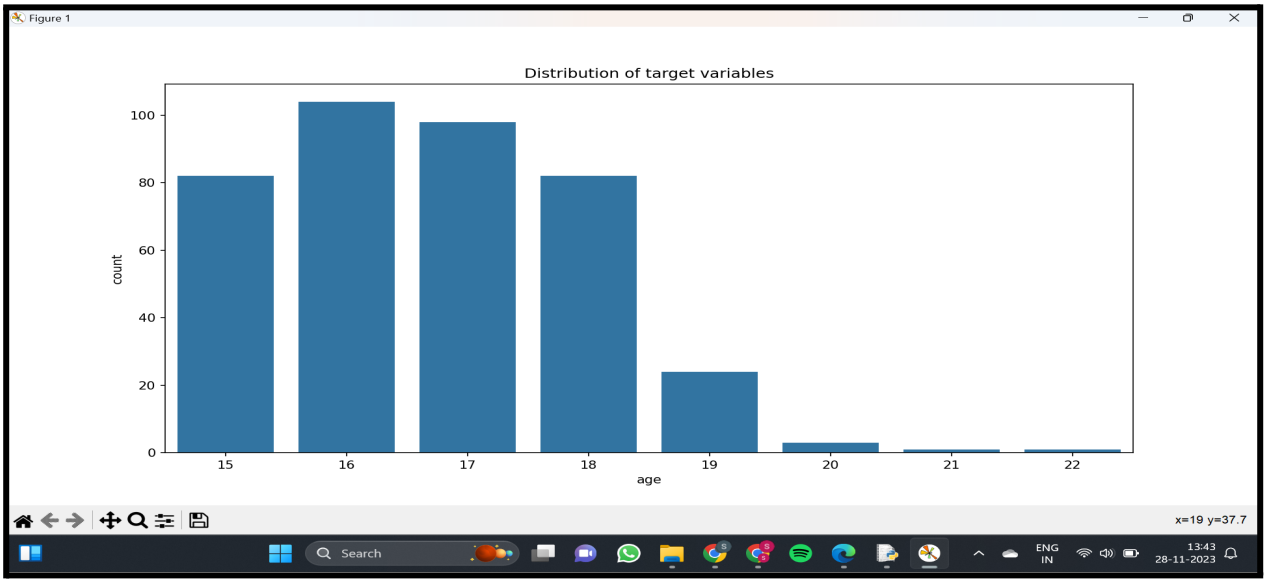
Syntax:

```
target_cnt=df["age"].value_counts()
print(target_cnt)

sns.countplot(x='age',data=df).set_title("Distribution of target variables")
plt.show()
```

Output:

```
================================ RESTART: C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py ================================
(395, 33)
age
16    104
17     98
18     82
15     82
19     24
20      3
22      1
21      1
Name: count, dtype: int64
```

Graph:



Inference:

In our dataset, we can observe the following things:

- All our data is numeric as well as categorical.
- There isn't any unnecessary attribute in our dataset, such as ID or Roll No, which we need to remove before we start visualization. All attributes we have are scientific values. So, we can begin with our next part, Data Visualization, to get inferences from the dataset.

So, we can begin with our next part, Data Visualization, to get inferences from the dataset.

DATA VISUALIZATION

1. First, we use DataFrame.hist() to plot a histogram.

```
df.hist(figsize=(6, 14), layout=(4,4), sharex=False)

#Show the plot
plt.show()
```

This histogram shows us the range of values for all our attributes in the dataset.

Inference:

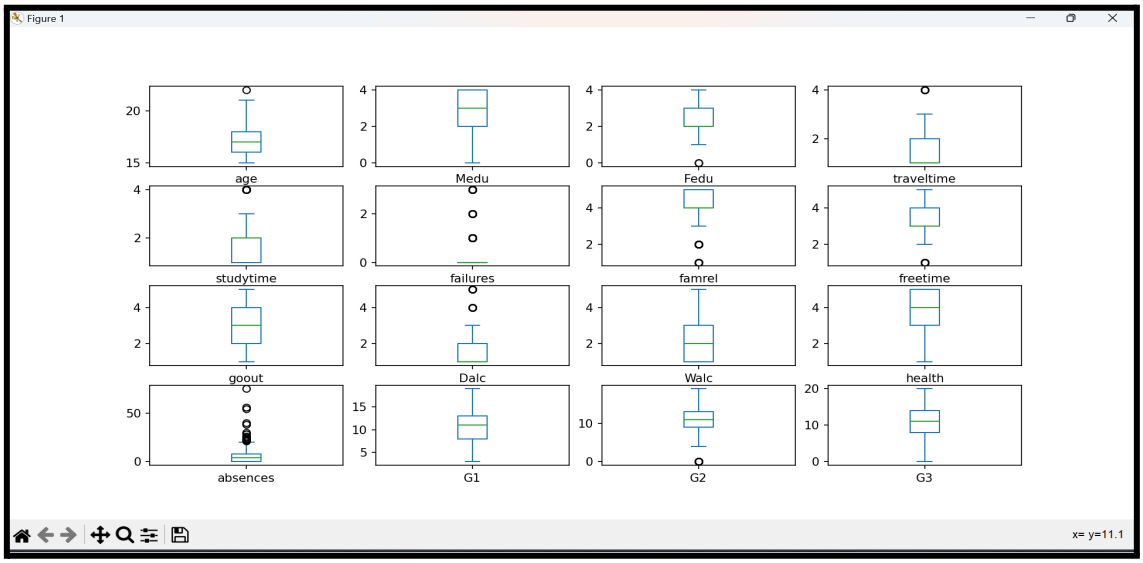
- This graph gives the histogram plots for each attribute in our dataset.

- It tells us the count of appearances of each value in each of the attributes.
- The x-values give the range of values of the particular attribute and the y-values give the count of each range

2. DataFrame.plot() to plot the box plot of different attributes

```
#Boxplot
df.plot(kind='box', figsize=(15, 12), layout=(4, 4), sharex=False, subplots=True)
#display
plt.show()
```

Output:



Inference:

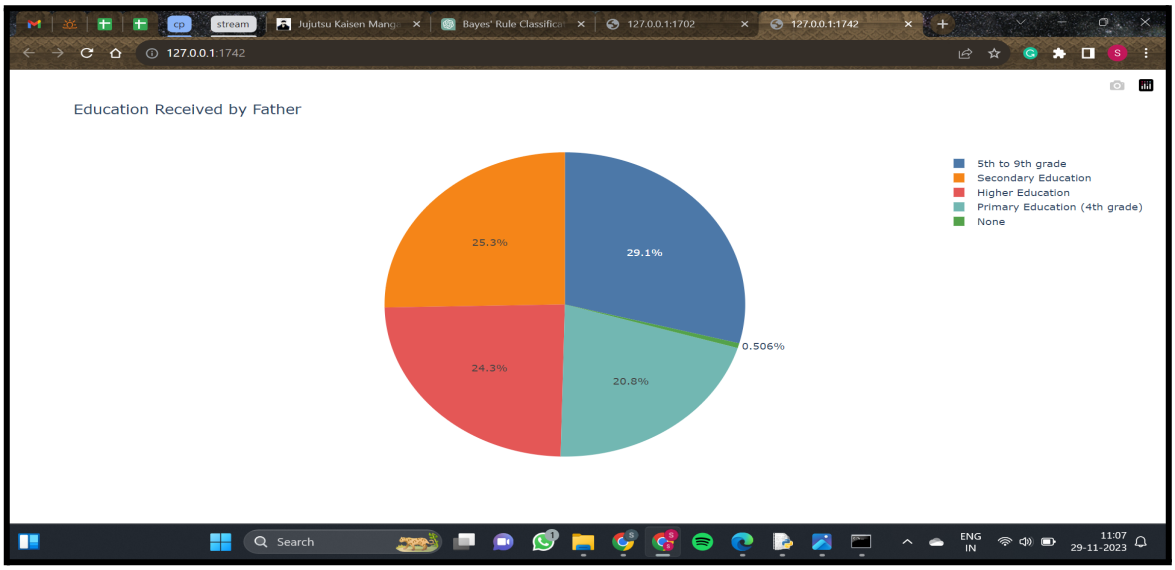
It tells us the overall patterns of response for attributes like health, study time, Age, failures etc.

3. We have used the plotly.pie() to display pie chart for attributes :

Fedu:(Father's Education):

```
# PLOTLY

# Mapping dictionary for Medu values
fedu_mapping = {
    0: 'None',
    1: 'Primary Education (4th grade)',
    2: '5th to 9th grade',
    3: 'Secondary Education',
    4: 'Higher Education'
}

# Map numeric values to their corresponding descriptions
df['Fedu'] = df['Fedu'].map(fedu_mapping)

# Create a pie chart using plotly.express
fig = px.pie(df, names='Fedu', title='Education Received by Father', color_discrete_sequence=px.colors.qualitative.T10)

# Show the plot
fig.show()
```

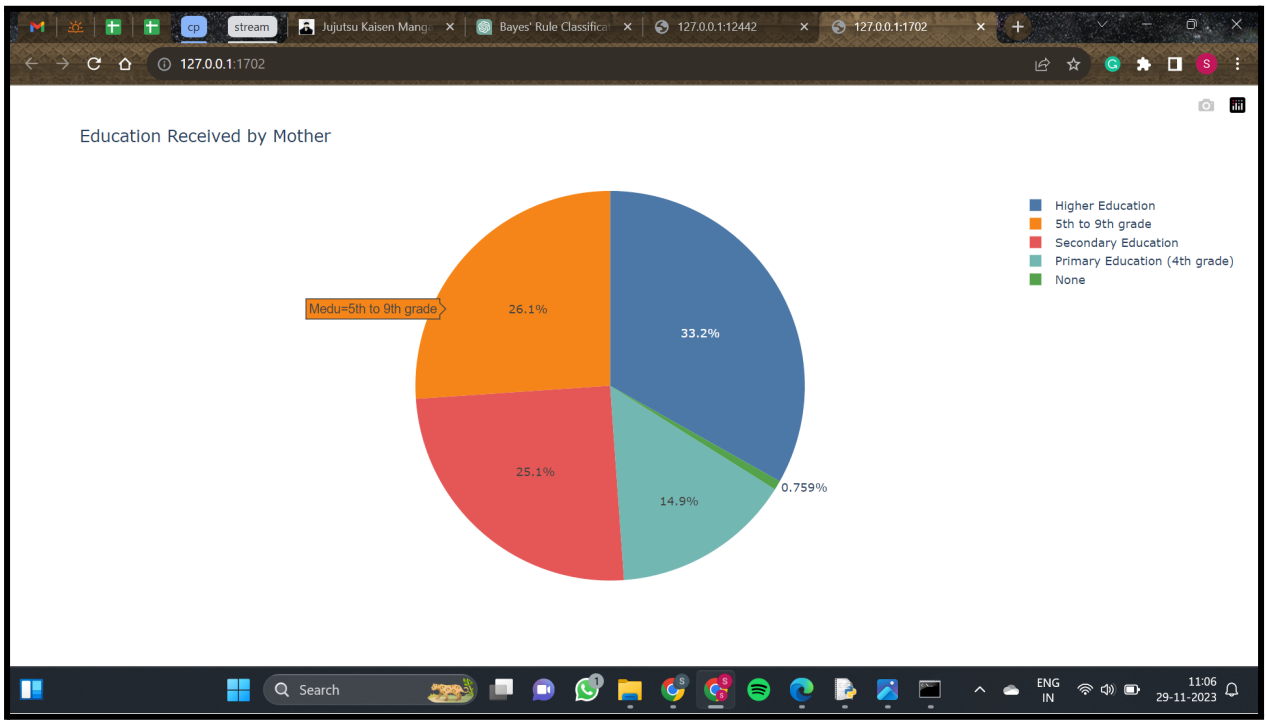
Output:



Medu:(Mother's Education):

```
# PLOTLY

# Mapping dictionary for Medu values
medu_mapping = {
    0: 'None',
    1: 'Primary Education (4th grade)',
    2: '5th to 9th grade',
    3: 'Secondary Education',
    4: 'Higher Education'
}

# Map numeric values to their corresponding descriptions
df['Medu'] = df['Medu'].map(medu_mapping)

# Create a pie chart using plotly.express
fig = px.pie(df, names='Medu', title='Education Received by Mother', color_discrete_sequence=px.colors.qualitative.T10)

# Show the plot
fig.show()
```

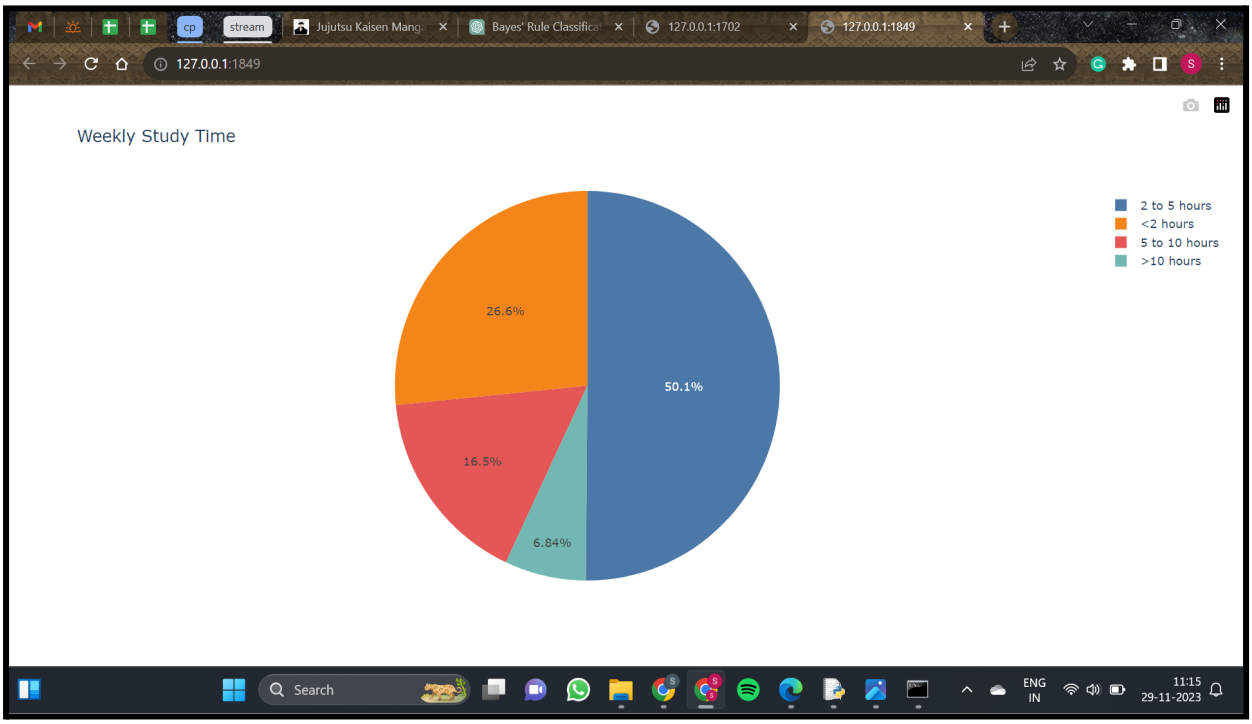
Output:



Weekly Study Time:

```
# PLOTLY

# Mapping dictionary for studytime values
studytime_mapping = {
    1: '<2 hours',
    2: '2 to 5 hours',
    3: '5 to 10 hours',
    4: '>10 hours'
}

# Map numeric values to their corresponding descriptions
df['studytime'] = df['studytime'].map(studytime_mapping)

# Create a pie chart using plotly.express
fig = px.pie(df, names='studytime', title='Weekly Study Time', color_discrete_sequence=px.colors.qualitative.T10)

# Show the plot
fig.show()
```

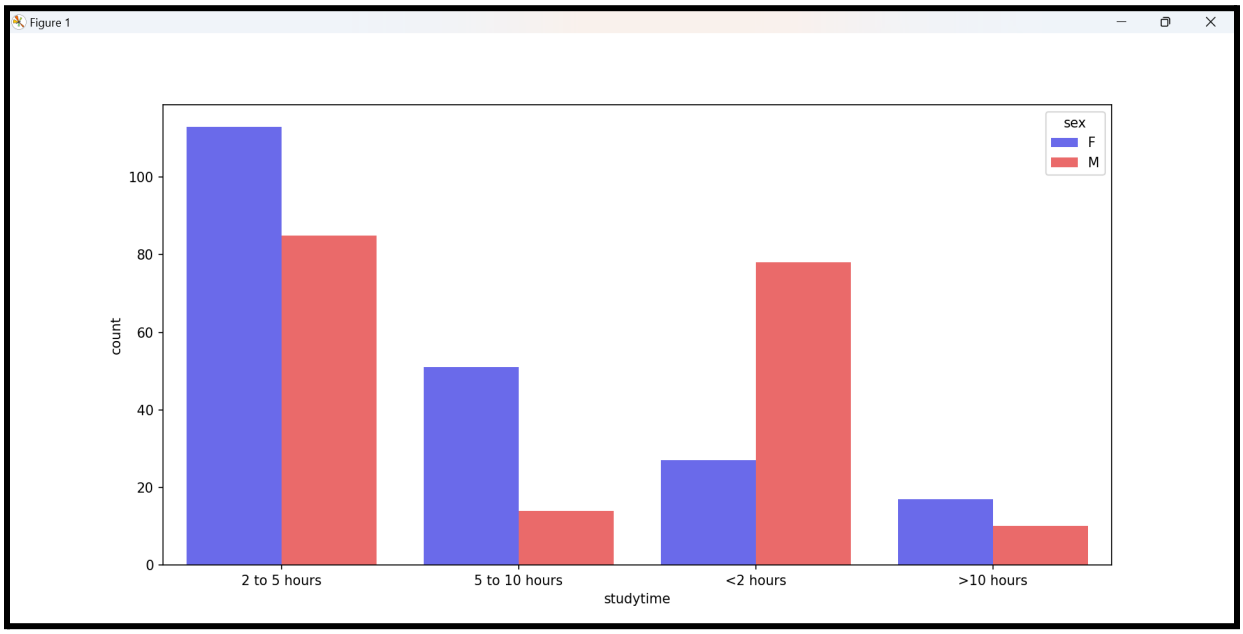
Output:



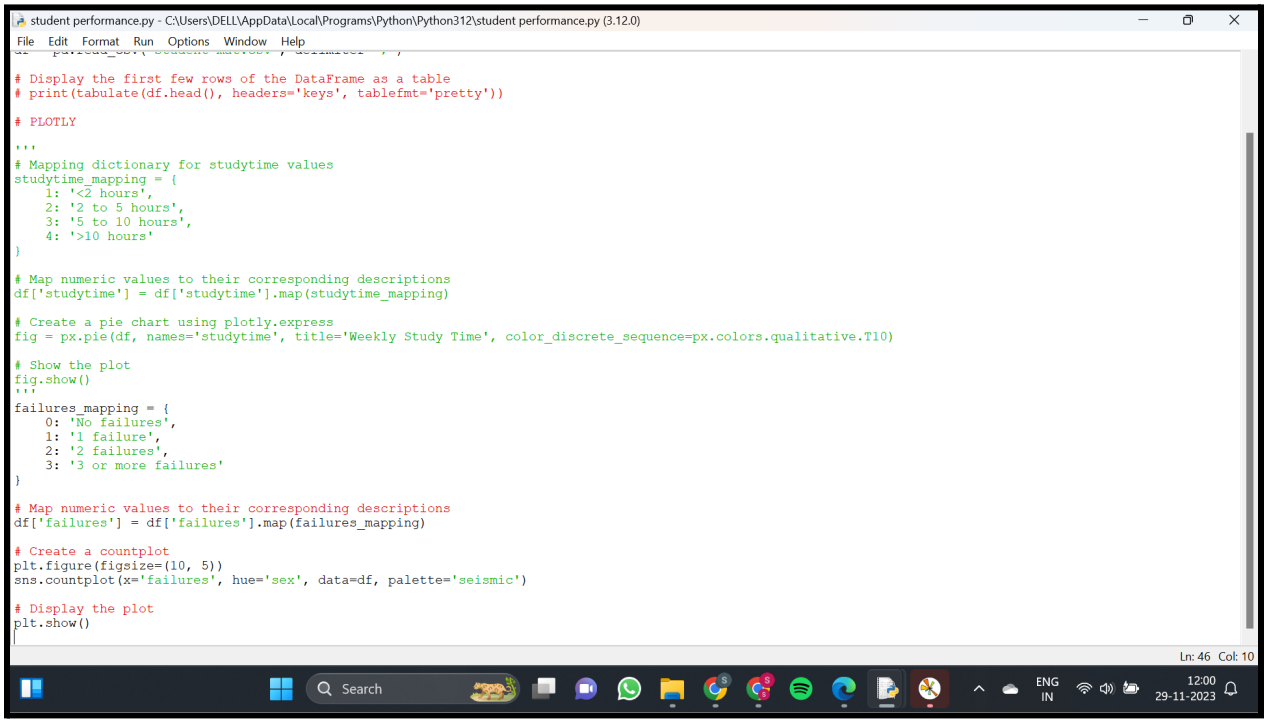
4. Sex and studytime

```
studytime_mapping = {
    1: '<2 hours',
    2: '2 to 5 hours',
    3: '5 to 10 hours',
    4: '>10 hours'
}

# Map numeric values to their corresponding descriptions
df['studytime'] = df['studytime'].map(studytime_mapping)

# Create a countplot
plt.figure(figsize=(10, 5))
sns.countplot(x='studytime', hue='sex', data=df, palette='seismic')

# Display the plot
plt.show()
```

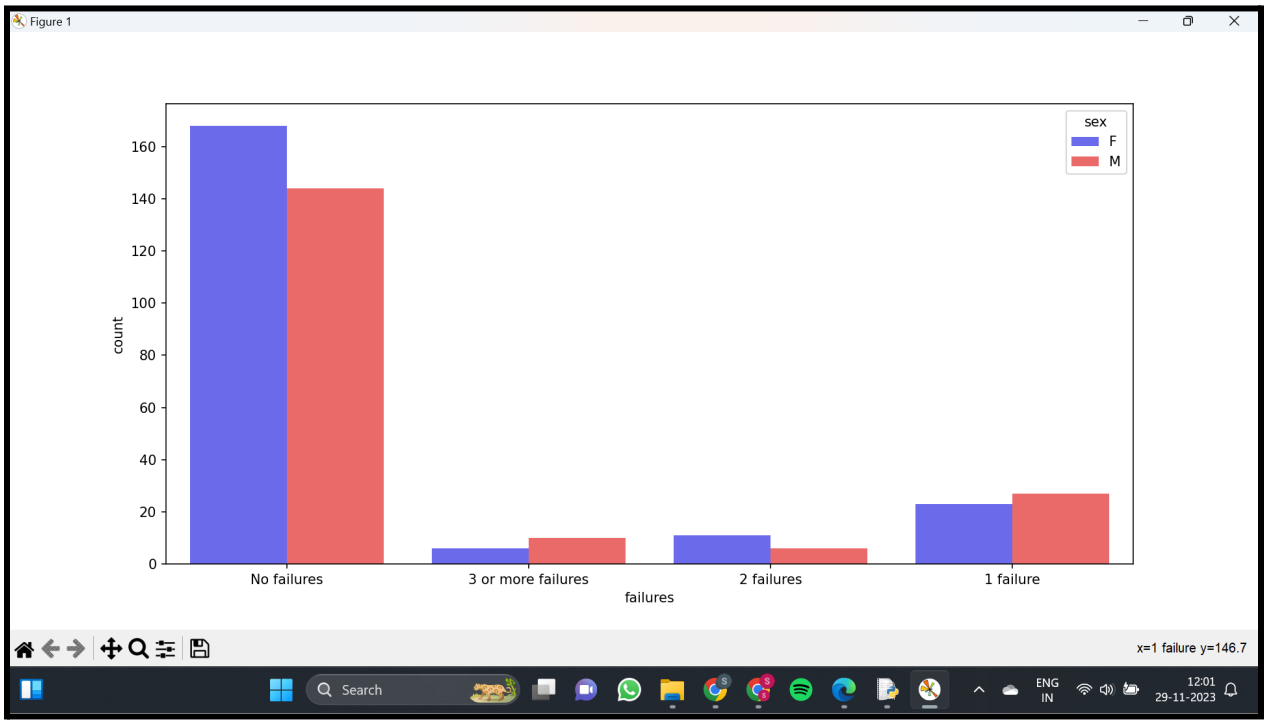
Output:



Sex and no .of failures:



Output:



Studytime and failures:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming the file is 'student-mat.csv'
df = pd.read_csv('student-mat.csv', delimiter=';')

# Mapping dictionary for studytime values
studytime_mapping = {
    1: '<2 hours',
    2: '2 to 5 hours',
    3: '5 to 10 hours',
    4: '>10 hours'
}

# Mapping dictionary for failures values
failures_mapping = {
    0: 'No failures',
    1: '1 failure',
    2: '2 failures',
    3: '3 or more failures'
}

# Map numeric values to their corresponding descriptions
df['studytime'] = df['studytime'].map(studytime_mapping)
df['failures'] = df['failures'].map(failures_mapping)

# Create a countplot
plt.figure(figsize=(12, 6))
sns.countplot(x='studytime', hue='failures', data=df, palette='viridis')

# Display the plot
plt.title('Study Time vs Failures')
plt.show()
```

Output:

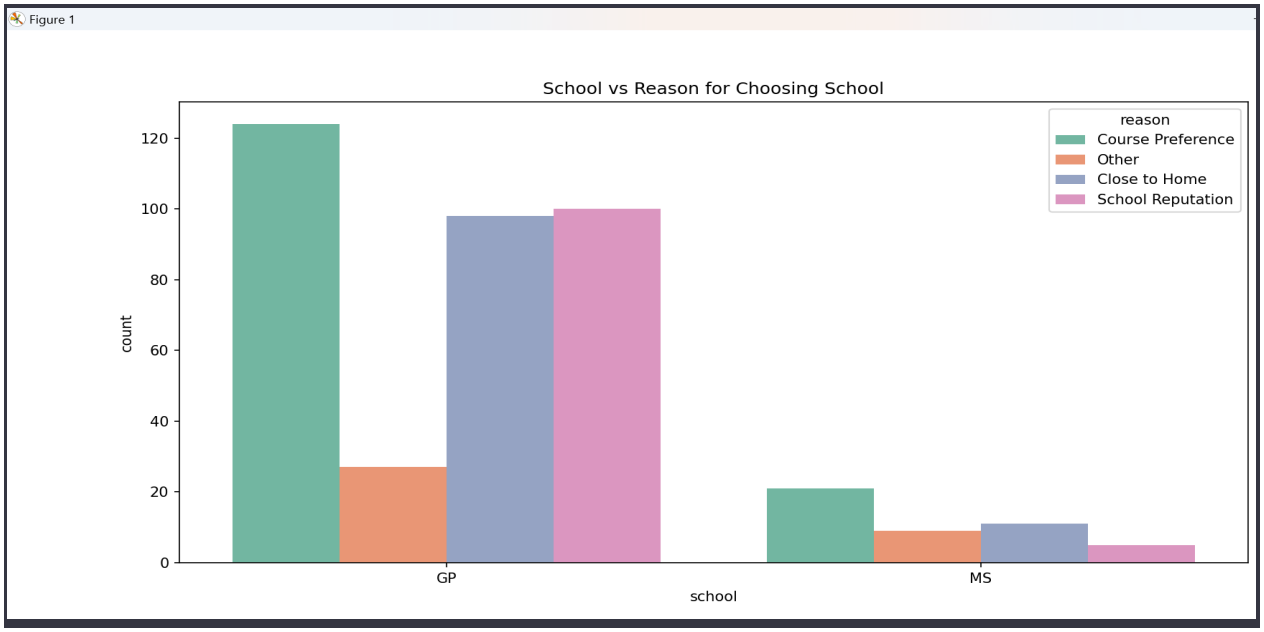
School - student's school (binary: 'GP' - Gabriel Pereira or 'MS'

Mousinho da Silveira)

Reason - reason to choose this school (nominal: close to 'home', school, 'reputation', 'course' preference or 'other').

```
student performance.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py (3.12.0)
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming the file is 'student-mat.csv'
df = pd.read_csv('student-mat.csv', delimiter=';')

# Mapping dictionary for reason values
reason_mapping = {
    'home': 'Close to Home',
    'reputation': 'School Reputation',
    'course': 'Course Preference',
    'other': 'Other'
}

# Map nominal values to their corresponding descriptions
df['reason'] = df['reason'].map(reason_mapping)

# Create a countplot
plt.figure(figsize=(12, 6))
sns.countplot(x='school', hue='reason', data=df, palette='Set2')

# Display the plot
plt.title('School vs Reason for Choosing School')
plt.show()
```

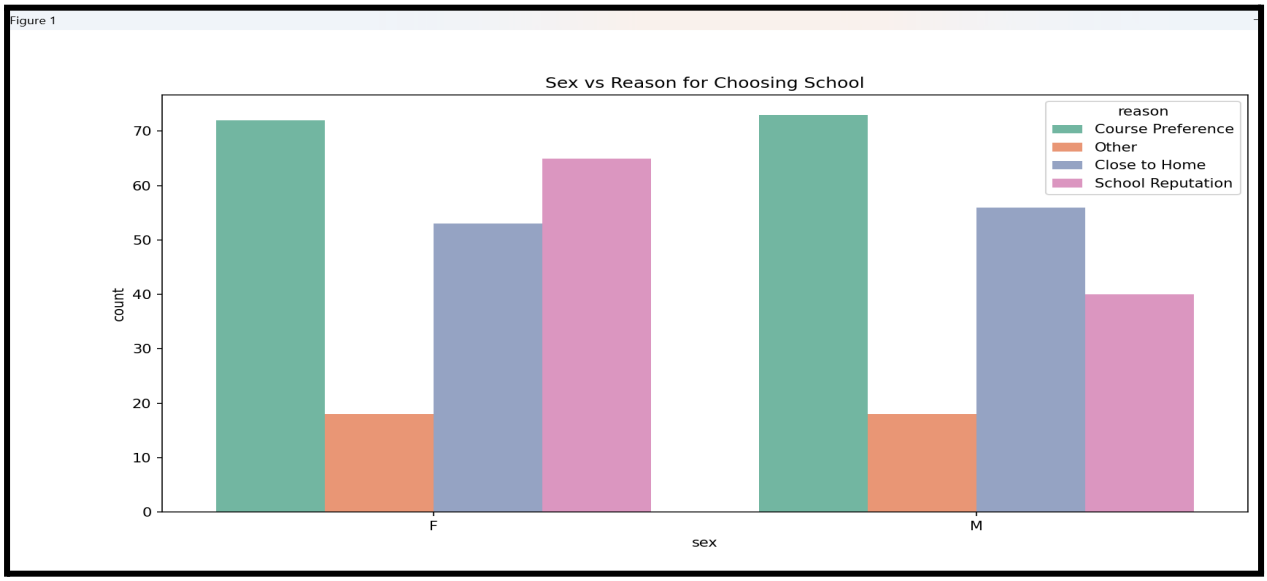
Output:



Sex vs reason for choosing school:

```
student performance.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py (3.12.0)
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming the file is 'student-mat.csv'
df = pd.read_csv('student-mat.csv', delimiter=';')

# Mapping dictionary for reason values
reason_mapping = {
    'home': 'Close to Home',
    'reputation': 'School Reputation',
    'course': 'Course Preference',
    'other': 'Other'
}

# Map nominal values to their corresponding descriptions
df['reason'] = df['reason'].map(reason_mapping)

# Create a countplot
plt.figure(figsize=(12, 6))
sns.countplot(x='sex', hue='reason', data=df, palette='Set2')

# Display the plot
plt.title('Sex vs Reason for Choosing School')
plt.show()
```

Output:



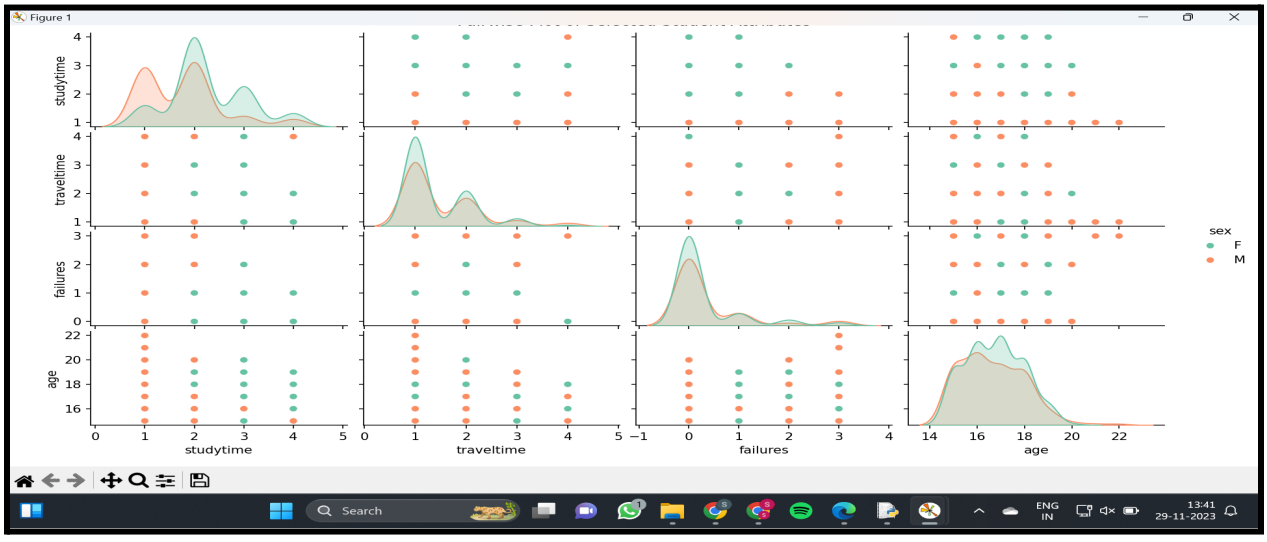
4. we use seaborn.pairplot() to pairwise plot all our attributes.

- The diagonal plots are univariate. The rest are bivariate plots.
- It shows the (n,2) combination relationship of attributes of the dataset.

sex vs Studytime traveltime age failures:

```
student performance.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py (3.12.0)
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming the file is 'student-mat.csv'
df = pd.read_csv('student-mat.csv', delimiter=';')

# Selecting specific attributes for the pairplot
selected_attributes = ['studytime', 'traveltime', 'failures','age', 'sex']

# Create a pairplot for the selected attributes
sns.pairplot(df[selected_attributes], hue='sex', palette='Set2')
plt.suptitle("Pairwise Plot of Selected Student Attributes", y=1.02)
plt.show()
```

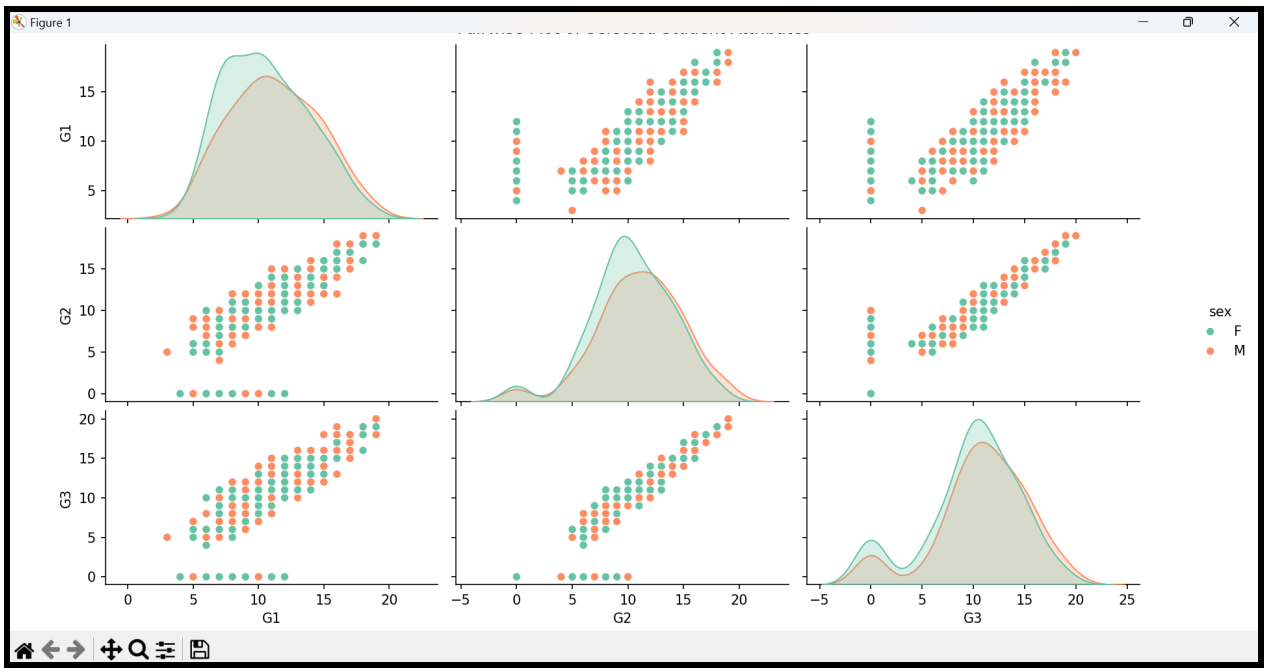
Output:



SEX vs G1,G2,G3:

```
student performance.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py (3.12.0)
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming the file is 'student-mat.csv'
df = pd.read_csv('student-mat.csv', delimiter=';')

# Selecting specific attributes for the pairplot
selected_attributes = ['G1', 'G2','G3', 'sex']

# Create a pairplot for the selected attributes
sns.pairplot(df[selected_attributes], hue='sex', palette='Set2')
plt.suptitle("Pairwise Plot of Selected Student Attributes", y=1.02)
plt.show()
```

Output:

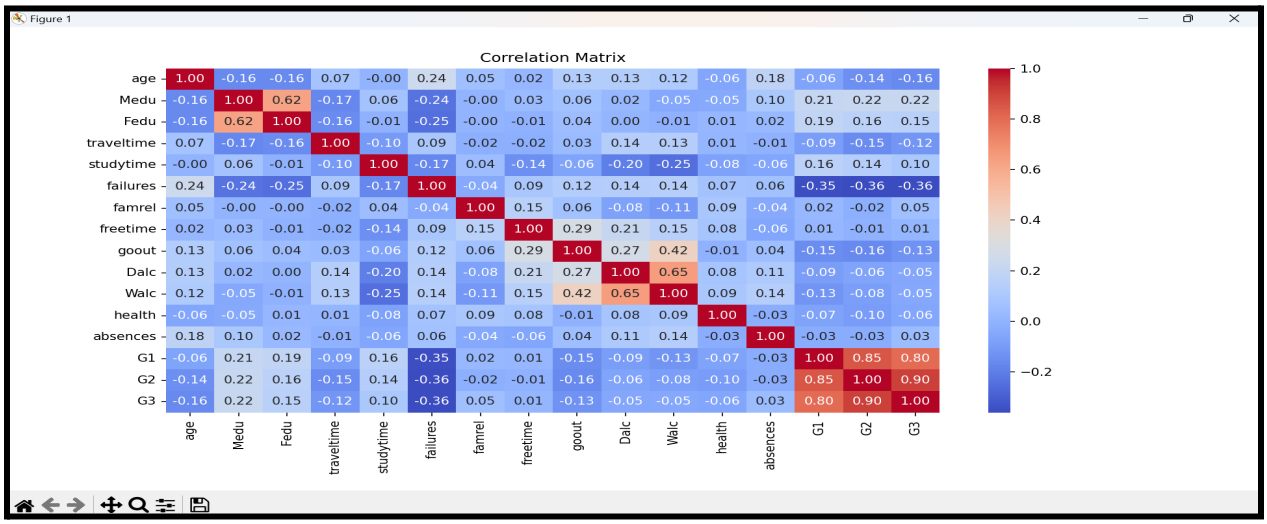


Inference:

- After inferencing the dataset from our graphs, we move towards the mathematical values of correlation between the attributes.

We plot the Correlation Matrix, using DataFrame.corr()

```
# Exclude non-numeric columns
numeric_df = df.select_dtypes(include='number')

# Calculate the correlation matrix
correlation_matrix = numeric_df.corr()

# Plot the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



DATA PRE-PROCESSING

1. Load the Data:

Load the Mathematics (mat) or Portuguese language (por) dataset using Pandas or any preferred data manipulation library.

2. Data Cleaning and Handling Missing Values:

Check for missing values in the dataset and handle them appropriately (remove, impute, or interpolate missing values based on the context).

3. Feature Selection/Engineering:

Given the importance of G1 and G2 for predicting G3, select relevant features for your clustering analysis. Include G1 and G2 along with other potential relevant features that might aid in the clustering task.

4. Encoding Categorical Variables:

If your dataset includes categorical variables (such as school, sex, and address), encode them into a numerical format suitable for clustering algorithms (e.g., one-hot encoding, label encoding).

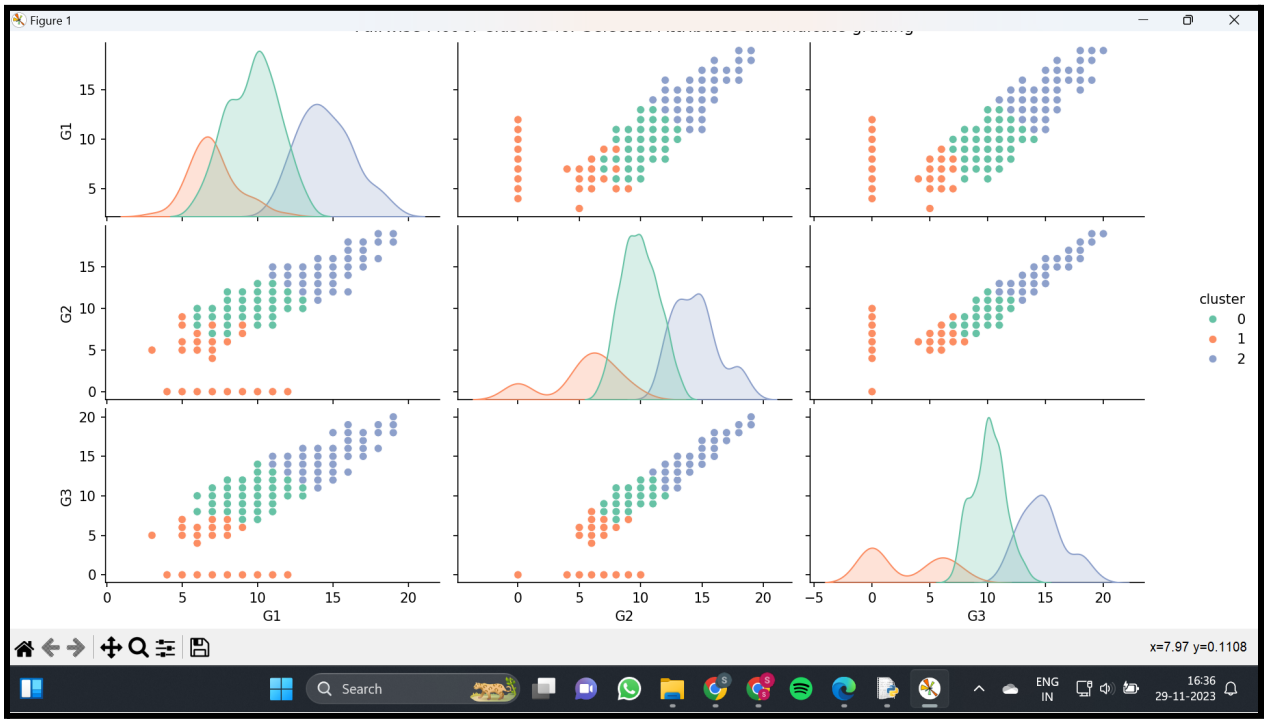
5. Scaling/Normalization:

Standardize or normalize the numerical features to bring them to a similar scale. K-means is sensitive to the scale of features, so scaling the features is crucial. Use techniques like StandardScaler or MinMaxScaler from scikit-learn for this purpose.

K-means clustering :

Splitting g1,g2,g3 ‘s data in 2 and 3 clusters:

```
# Selecting specific attributes for clustering
selected_attributes = ['G1', 'G2', 'G3']

# Selecting relevant numerical columns
numeric_columns = df[selected_attributes]

# Standardize the data
scaler = StandardScaler()
data_standardized = scaler.fit_transform(numeric_columns)

num_clusters = 2

# Apply K-Means clustering
kmeans = KMeans(n_clusters=num_clusters, n_init=10, random_state=42)  # Set n_init explicitly
df['cluster'] = kmeans.fit_predict(data_standardized)

# Visualize the clusters
sns.pairplot(df, hue='cluster', palette='Set2', vars=selected_attributes)
plt.suptitle("Pairwise Plot of Clusters for Selected Attributes that indicate grading", y=1.02)
plt.show()
```



Hierarchical clustering:

Agglomerative between g1,g2,g3:

```
# Selecting specific attributes for clustering
selected_attributes = ['G1', 'G2', 'G3']

# Selecting relevant numerical columns
numeric_columns = df[selected_attributes]

# Calculate the linkage matrix using Ward's method
linkage_matrix = linkage(numeric_columns, method='ward')

# Plot the dendrogram
plt.figure(figsize=(12, 6))
dendrogram(linkage_matrix, orientation='top', labels=df.index, distance_sort='descending', show_leaf_counts=True)
plt.title("Hierarchical Clustering Dendrogram for Selected Attributes")
plt.xlabel("Student Index")
plt.ylabel("Distance")
plt.show()

# Extract clusters using a chosen distance threshold
from scipy.cluster.hierarchy import fcluster

distance_threshold = 30  # Adjust this threshold based on the dendrogram
df['cluster'] = fcluster(linkage_matrix, distance_threshold, criterion='distance')

# Visualize the clusters
sns.pairplot(df, hue='cluster', palette='Set2', vars=selected_attributes)
plt.suptitle("Pairwise Plot of Hierarchical Clusters for Selected Attributes", y=1.02)
plt.show()
```

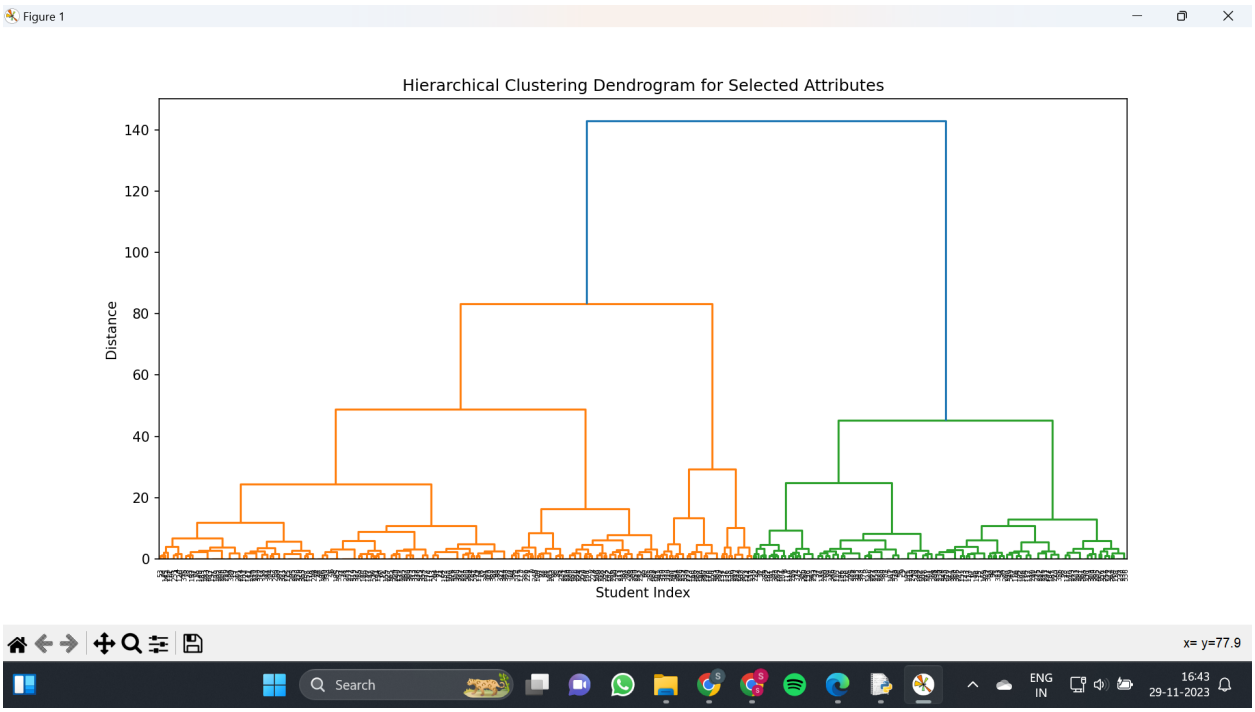


Figure 1
Hierarchical Clustering Dendrogram for Selected Attributes
Distance
0
20
40
60
80
100
120
140
Student Index
x= y=77.9
Search
ENG
IN
16:43
29-11-2023

ML CLASSIFICATION ALGORITHMS AND THEIR IMPLEMENTATION

- After getting a cleaned dataset, we can now apply our prediction algorithms.
- In our project, instead of applying just one, we use 3 different classification algorithms to predict the results.
- The motivation to apply all these algorithms was that we wanted to compare their accuracy results to see which algorithm works better on our dataset.

We applied the algorithms given below.

1. Decision Tree Classifier
2. Support Vector Machine
3. Naive Bayes Classifier

- For each case, we've visualized the **Confusion Matrix** along with it.
- We have also displayed the c**lassification report for each case.**
- We have also displayed the **accuracy percentage for each case too.**

1. Decision Tree

Decision trees are one of the most practically used methods for supervised learning. They are used for both classification and regression tasks. The basic working ideology is based on predicting the value of a target variable by learning simple decision rules from the training set. They are generally in the form of if-else statements and deeper we go into the tree the rules become more complex and the model becomes fitter.
It's basically a tree data structure with nodes representing an attribute which causes the split when asked a question.
The edges represent answers to questions and leaf nodes represent the output class labels. The entire process is recursive and is repeated for every subtree

node.The attribute from which split occurs depends on measures like GINI Index, Gain, Entropy etc.

Code:

```
import time

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import sklearn.metrics
from sklearn import tree
from sklearn.metrics import accuracy_score

start_time = time.time()


df = pd.read_csv('student-mat.csv', delimiter=';')
df['sex'] = df['sex'].map({'M': 0, 'F': 1})
df['school'] = df['school'].map({'GP': 0, 'MS': 1})
df['famsize'] = df['famsize'].map({'LE3': 0, 'GT3': 1})
df['address'] = df['address'].map({'U': 0, 'R': 1})
df['Pstatus'] = df['Pstatus'].map({'A': 0, 'T': 1})
df['schoolsup'] = df['schoolsup'].map({'yes': 0, 'no': 1})
df['famsup'] = df['famsup'].map({'yes': 0, 'no': 1})
df['paid'] = df['paid'].map({'yes': 0, 'no': 1})
df['activities'] = df['activities'].map({'yes': 0, 'no': 1})
df['nursery'] = df['nursery'].map({'yes': 0, 'no': 1})
df['higher'] = df['higher'].map({'yes': 0, 'no': 1})
df['internet'] = df['internet'].map({'yes': 0, 'no': 1})
df['romantic'] = df['romantic'].map({'yes': 0, 'no': 1})
df['guardian'] = df['guardian'].map({'mother': 0, 'father': 1})
```

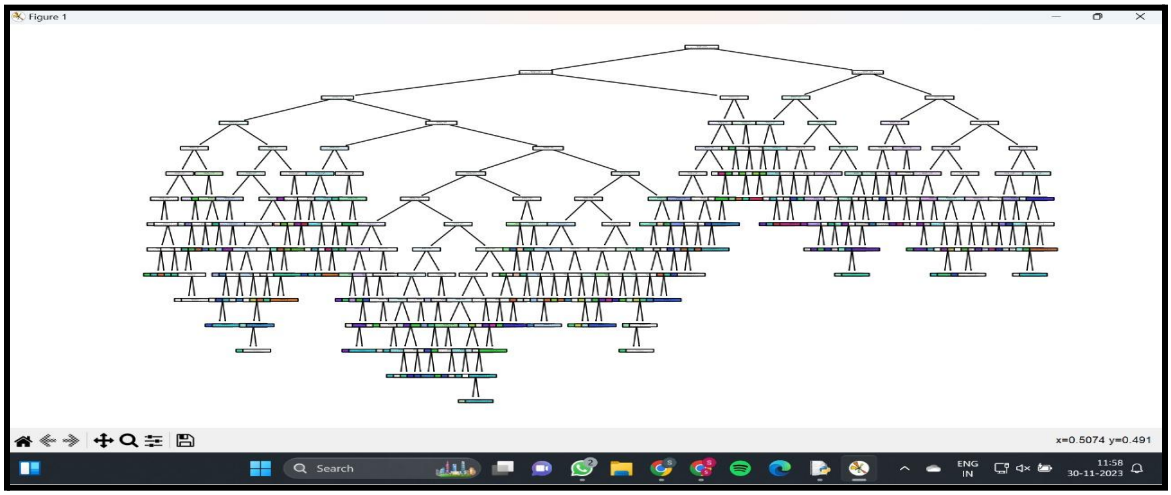
```
predictors = df.iloc[:, :8].values
target = df.iloc[:, 32].values
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=0.25)
print(pred_train.shape)
print(pred_test.shape)
print(tar_train.shape)
print(tar_test.shape)

features = list(df.columns[:8])
```

```
classifier = DecisionTreeClassifier(criterion = "entropy", random_state = 1, splitter='best')
'''classifier=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=5,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=100, splitter='best')
#max_depth=6, min_samples_leaf=7
#bagging = BaggingClassifier(DecisionTreeClassifier())'''
classifier = classifier.fit(pred_train,tar_train)

predictions = classifier.predict(pred_test)

print(sklearn.metrics.confusion_matrix(tar_test, predictions))
```

```
#classification accuracy
print("accuracy of training dataset is{:.2f}".format(classifier.score(pred_train,tar_train)))
print("accuracy of test dataset is {:.2f}".format(classifier.score(pred_test,tar_test)))
#print(accuracy_score(tar_test, predictions, normalize = True))

#error rate
print("Error rate is",1- accuracy_score(tar_test, predictions, normalize = True))

#sensitivity
print("sensitivity is", sklearn.metrics.recall_score(tar_test, predictions,labels=None, average =  'micro', sample_weight=None))
#specificity
print("specificity is", 1 - sklearn.metrics.recall_score(tar_test, predictions,labels=None, average =  'micro', sample_weight=None))

# Plot the decision tree (textual representation)
plt.figure(figsize=(20, 10))
tree.plot_tree(classifier, feature_names=features, class_names=None, filled=True, rounded=True)
plt.show()
```

Output:

```
(296,  8)
(99,  8)
(296,)
(99,)
[[0  1  0  0  1  0  1  1  2  0  0  0  0  0  0  0]
 [0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [0  0  0  0  0  0  2  2  2  0  0  0  0  0  0  0]
 [0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0]
 [1  0  0  0  0  0  2  1  0  0  0  0  0  1  0  0]
 [1  0  0  0  2  0  3  1  0  0  0  0  1  0  0  0]
 [2  2  0  0  0  1  3  2  2  1  1  1  2  1  0  0]
 [5  1  1  0  1  2  1  2  0  0  0  1  0  0  0  0]
 [0  0  0  0  0  1  1  0  1  2  1  1  0  0  0  0]
 [1  0  0  0  0  0  1  3  1  1  0  0  0  0  0  0]
 [1  0  0  0  1  0  2  0  1  0  2  0  0  0  0  0]
 [1  0  0  1  0  2  2  1  1  0  1  0  0  0  1  0]
 [1  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0]
 [0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0]
 [2  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]]
accuracy of training dataset is0.77
Error rate is 0.898989898989899
sensitivity is 0.10101010101010101
specificity is 0.898989898989899
```



2. Support Vector Machine

Support Vector Machine uses a supervised learning algorithm. It is used to find a hyperplane that will separate the data classes.
Now, there may be many hyperplanes which can separate the data, but SVM tries to find the best fit line for this separation.
This classifier generally works well when there is a clear line of separation between the classes, and the dataset is not large enough.

Code:

```
import time
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.multiclass import OneVsOneClassifier
from sklearn.preprocessing import label_binarize
from sklearn.exceptions import UndefinedMetricWarning
import warnings

# Load the data for Mathematics (assuming it's stored in a CSV file)
math_data = pd.read_csv('student-mat.csv', delimiter=';')

# Convert categorical variables to numerical using one-hot encoding
math_data = pd.get_dummies(math_data, columns=['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian',
                                               'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic'])

selected_features = ['age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'famrel', 'freetime',
                     'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2']  # Features

X_mat = math_data[selected_features]  # Feature matrix X_mat
y_mat = math_data['G3']

# Splitting the data into training and testing sets for Mathematics dataset
X_train_mat, X_test_mat, y_train_mat, y_test_mat = train_test_split(X_mat, y_mat, test_size=0.2, random_state=42)

# Create an SVM classifier for Mathematics using One-vs-One strategy
svm_classifier_mat = OneVsOneClassifier(SVC(kernel='linear'))

# Train the SVM classifier for Mathematics
svm_classifier_mat.fit(X_train_mat, y_train_mat)
```

```
svm.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\svm.py (3.12.0)
File  Edit  Format  Run  Options  Window  Help

# Make predictions on Mathematics test set
predictions_mat = svm_classifier_mat.predict(X_test_mat)

# Calculate accuracy for Mathematics predictions
accuracy_mat = accuracy_score(y_test_mat, predictions_mat)
print(f"Accuracy for Mathematics dataset: {accuracy_mat}")

# Filter out classes with no positive samples
with warnings.catch_warnings():
    warnings.simplefilter("ignore", category=UndefinedMetricWarning)
    y_test_bin = label_binarize(y_test_mat, classes=np.unique(y_mat))

fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(len(np.unique(y_mat))):
    with warnings.catch_warnings():
        warnings.simplefilter("ignore", category=UndefinedMetricWarning)
        fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], svm_classifier_mat.decision_function(X_test_mat)[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])
```

```
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), svm_classifier_mat.decision_function(X_test_mat).ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

plt.figure(figsize=(8, 6))
plt.plot(fpr["micro"], tpr["micro"], color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.format(roc_auc["micro"]))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Mathematics dataset')
plt.legend(loc="lower right")
plt.show()

# Confusion Matrix and Classification Report
conf_mat = confusion_matrix(y_test_mat, predictions_mat)
print("Confusion Matrix:\n", conf_mat)
print("\nClassification Report:\n", classification_report(y_test_mat, predictions_mat))
```

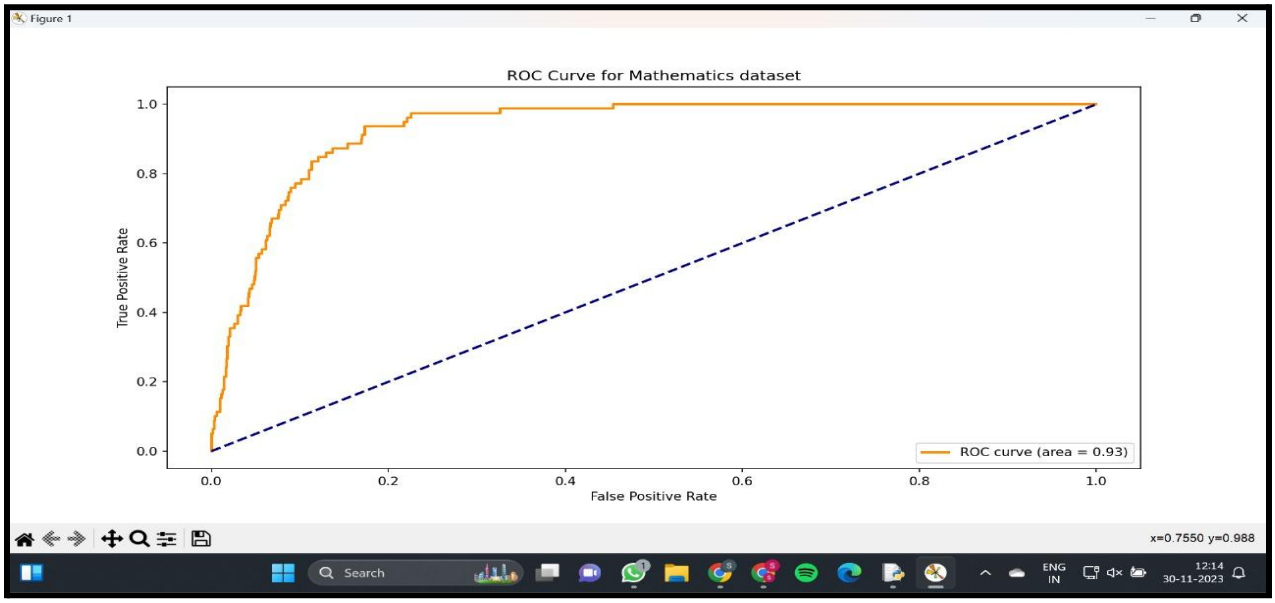
Classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.60      0.67         5
           5       1.00      0.25      0.40         4
           6       0.50      0.33      0.40         6
           7       0.00      0.00      0.00         1
           8       0.25      0.33      0.29         6
           9       0.40      0.40      0.40         5
          10       0.38      0.27      0.32        11
          11       0.36      0.80      0.50         5
          12       0.00      0.00      0.00         5
          13       0.80      0.80      0.80         5
          14       0.50      0.67      0.57         6
          15       0.58      0.70      0.64        10
          16       0.50      0.25      0.33         4
          17       0.00      0.00      0.00         3
          18       0.00      0.00      0.00         1
          19       0.00      0.00      0.00         2
          20       0.00      0.00      0.00         0

    accuracy                           0.42        79
   macro avg       0.35      0.32      0.31        79
weighted avg       0.44      0.42      0.41        79

>>>
========================================== RESTART: C:\Users\
Accuracy for Mathematics dataset: 0.4177215189873418
```

Confusion matrix:

```
Confusion Matrix:
[[3 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 2 2 1 1 0 0 0 0 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]
 [1 0 1 0 2 0 1 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 2 1 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 1 2 3 3 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 4 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 2 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 4 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 4 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 3 7 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 3 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0]
 [0 0 0 0 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 0 1 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

ROC Curve :



3.Naive Bayes classifier

A Naive Bayes classifier is a machine learning algorithm that is based on Bayes' theorem, which is a probability theory concept. It's a simple and probabilistic classification algorithm that makes assumptions about the independence of features. Despite its simplicity and the "naive" assumption, it often performs well in practice, especially in text classification tasks such as spam filtering and sentiment analysis.

Code:

student performance.py - C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py (3.12.0)

File Edit Format Run Options Window Help

```
# Use LabelEncoder for categorical features
le = preprocessing.LabelEncoder()
sex_cat = le.fit_transform(adult_df.sex)
address_cat = le.fit_transform(adult_df.address)
guardian_cat = le.fit_transform(adult_df.guardian)

# Add new columns with encoded values
adult_df_rev['sex_cat'] = sex_cat
adult_df_rev['address_cat'] = address_cat
adult_df_rev['guardian_cat'] = guardian_cat

# Drop the original categorical columns
dummy_fields = ['sex', 'address', 'guardian']
adult_df_rev = adult_df_rev.drop(dummy_fields, axis=1)
```

```
# Reorder the columns
adult_df_rev = adult_df_rev[['sex_cat', 'age', 'Medu', 'Fedu', 'studytime', 'failures', 'famrel', 'freetime', 'health', 'absences', 'address_cat', 'guardian

# Extract features and target
features = adult_df_rev.values[:, :11]
target = adult_df_rev.values[:, 11]

# Split the data into training and testing sets
features_train, features_test, target_train, target_test = train_test_split(features, target, test_size=0.3, random_state=10)

# Initialize Gaussian Naive Bayes classifier
clf = GaussianNB()

# Train the classifier
clf.fit(features_train, target_train)
```

```
# Predict target values for the test set
target_pred = clf.predict(features_test)

# Evaluate the classifier
print("Predicted values:", target_pred)
print("Accuracy:", accuracy_score(target_test, target_pred))
print("Classification error:", 1 - accuracy_score(target_test, target_pred))
print("Sensitivity:", recall_score(target_test, target_pred, average='micro'))
print("Specificity:", 1 - recall_score(target_test, target_pred, average='micro'))
```

Output: Predicted values, accuracy, classification error, specificity, sensitivity

```
>>>
================================ RESTART: C:\Users\DELL\AppData\Local\Programs\Python\Python312\student performance.py ===
Predicted values: [1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2
 1 1 1 0 2 1 1 1 1 1 2 1 1 2 1 0 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 2 0 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2
 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1]
Accuracy: 0.6413502109704642
Classification error: 0.3586497890295358
Sensitivity: 0.6413502109704642
Specificity: 0.3586497890295358
>>>
```

CONCLUSION

To summarize our results from the implementations, we get the following table:

Algorithm	Accuracy
Decision Tree	77%
Support Vector Machine	41%
Naive Bayes Classifier	64%

From the results, it is clear that **Decision Tree** gives the highest accuracy of **77%** on our dataset.

REFERENCES

1. The class presentations of IDS

2. Official documentations of Seaborn, Matplotlib, scikit-learn, Pandas, Numpy

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