King Saud University
College of Computer and Information Sciences
Information Technology department

IT 326: Data Mining
Course Project

Students Adaptability Level Prediction in Online Education

Project final Report

LAB Day/Time:
Thursday/1

Group#:	1	
Section#:	64073	
Group Members:	**Name**	**ID**
	Nouf Alaskar	443200456
	Raghad Hassan	443204743
	Walaa Almutairi	443200973
	Raghad Alhulwah	444200453
	Latefa Alshareef	443200620

1. Problem

With the rapid shift toward online education, many students face challenges adapting to virtual learning environments. This difficulty has been exacerbated by technological limitations, varying levels of financial stability, and differences in learning resources. Such challenges can negatively impact students' academic performance and overall well-being. In our project, we aim to study and analyze student data to identify the key factors influencing adaptability to online education. By predicting students' adaptability levels, we can help educators and policymakers implement targeted interventions, thereby improving the effectiveness of online learning and enhancing students' experiences.

2. Data Mining Task

In our project, we will employ two data mining tasks to analyze and predict students' adaptability levels in online education: classification and clustering.

For classification, we will train a model to determine whether a student falls into the "High Adaptability" or "Low Adaptability" category, based on features such as age, gender, education level, financial condition, network type, class duration, and device type. The classification will be based on the "Adaptivity Level" class.

As for clustering, our model will group students who share similar characteristics, independent of the class attribute (Adaptivity Level). These clusters will help identify patterns and similarities among students, providing valuable insights into the factors influencing adaptability. This may also uncover previously unnoticed groupings, allowing educators and policymakers to better tailor their strategies for improving online learning experiences.

3. Data

- The Source: https://www.kaggle.com/datasets/mdmahmudulhasansuzan/students-adaptability-level-in-online-education [1]

- Number of attributes: 11

- No. of objects: 1205

- Class label: Adaptivity Level

To try to understand our data, we reviewed:

- **Attributes' description**

Attributes Name	Data type	Description	Possible Values
Gender	Binary	Student's gender type	Girl, Boy
Age	Ordinal	Student's age range	1-5, 6-10, 11-15, 16-20, 21-25, 26-30
Education Level	Nominal	Student's education institution level	School, College, University
Institution Type	Binary	Student's education institution type	Government, Non Government
IT Studen	Binary	Whether the student is studying IT or not	Yes, No
Location	Binary	Whether the student is studying in their hometown	Yes, No
Load-shedding	Binary	Level of load shedding	High, Low
Financial Condition	Ordinal	Student's family's financial condition	Rich, Mid, Poor
Internet Type	Binary	Student's most used internet type	Wifi, Mobile Data
Network Type	Ordinal	Network connectivity type	2G, 3G, 4G
Class Duration	Ordinal	Student's daily class duration in hours	0, 1-3, 3-6
Self Lms	Binary	Whether the student's institution has its own LMS	Yes, No
Device	Nominal	Student's most used device in class	Computer, Tab, Mobile
Adaptivity Level	Ordinal	Student's adaptibility level to online education	High, Moderate, Low

- **Missing values**

```
missing_values = df.isna().sum()
print("\nTotal number of missing values in the dataset:", missing_values.sum())

# Creates a table that counts the number of missing values for each variable in the dataset
print("\nMissing Values:")
missing_table = pd.DataFrame({'Variable': missing_values.index, 'Missing Values': missing_values.values})
display(missing_table)
```

```
Total number of missing values in the dataset: 0

Missing Values:
```

	Variable	Missing Values
0	Gender	0
1	Age	0
2	Education Level	0
3	Institution Type	0
4	IT Student	0
5	Location	0
6	Load-shedding	0
7	Financial Condition	0
8	Internet Type	0
9	Network Type	0
10	Class Duration	0
11	Self Lms	0
12	Device	0
13	Adaptivity Level	0

We have no missing values.

- **Statical Measures for each numeric column:**

<u>-Show Five Number Summary:</u>

Using the *df.describe()* function, several key observations can be made from the summary statistics of the "Age" column:

- Age: The age distribution shows a significant range, with the minimum age being 3 years and the maximum age reaching 28 years. The average age of the dataset is 17.22 years, with a standard deviation of 6.29, indicating that there is moderate variability in the age of individuals. The median age (50th percentile) is 18 years, with the lower quartile (25th percentile) at 13 years, and the upper quartile (75th percentile) at 23 years. This suggests that the majority of individuals fall between the ages of 13 and 23 years, but there are a few individuals outside this range.

```
df.describe()
```

	Age
count	1205.000000
mean	17.219917
std	6.285479
min	3.000000
25%	13.000000
50%	18.000000
75%	23.000000
max	28.000000

<u>-Show the Variance:</u>

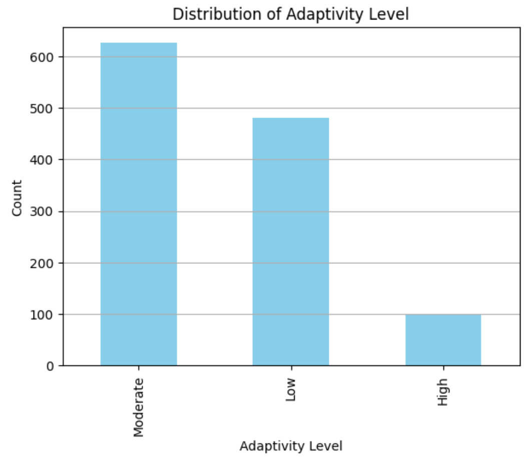
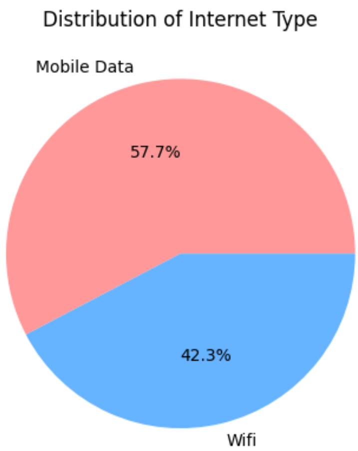
Variance helps understand the extent of dispersion or scatter of values in each column. As the variance increases, it indicates that the values are more spread out and scattered away from the mean, whereas decreasing variance suggests that the values are less scattered and closer to the mean value. Therefore, our variance results indicate the following:

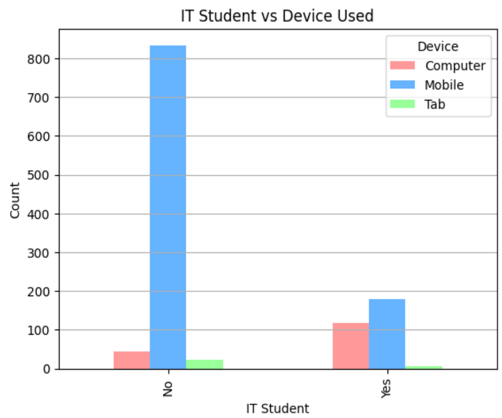
- Age: The variance of 39.51 indicates a relatively high level of dispersion, suggesting that the ages in this dataset vary significantly around the mean value of 17.22 years.
- Median of Age: The median age of 18.0 indicates that 50% of the values fall below this age, which shows that the distribution is fairly centered around the mean.
- Mode of Age: The mode is 23, meaning this is the most frequent age in the dataset. This suggests that many individuals share this age.

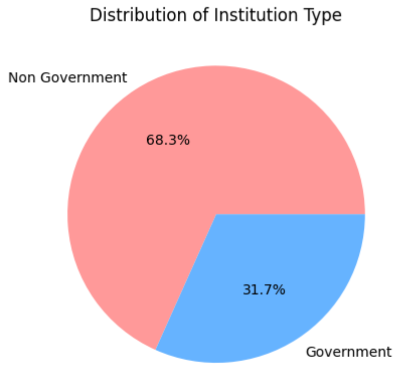
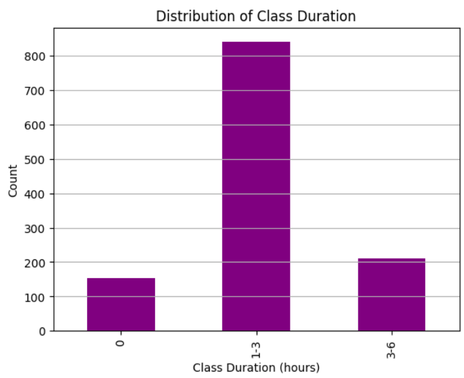
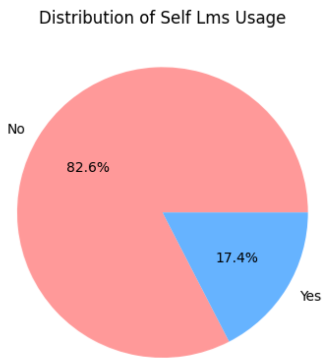
```
Variance of Age: 39.50724417915386
Median of Age: 18.0
Mode of Age: 0    23
Name: Age, dtype: int64
```

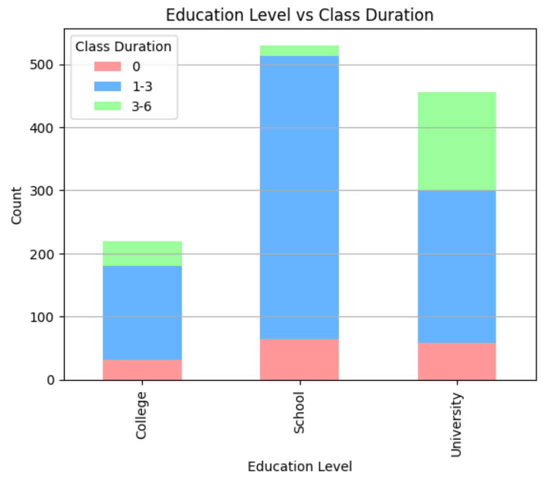
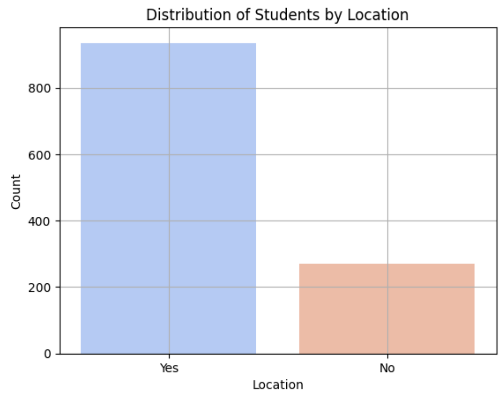
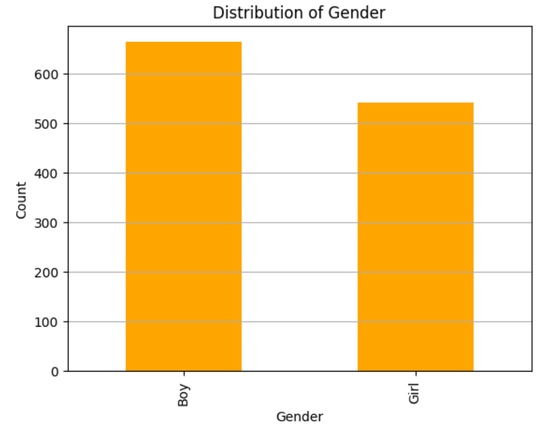
- **Understanding the data through graph representations:**

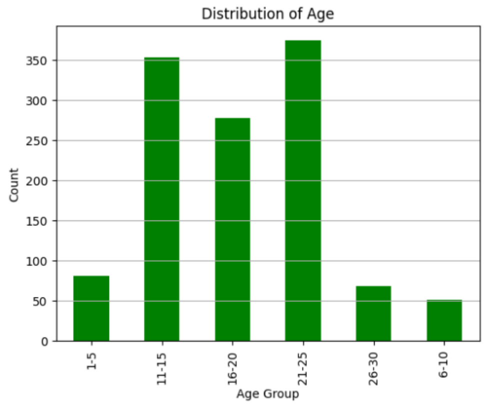
The "**Adaptivity Level**" column was used to analyse the relationship between students' adaptability levels in online education and various attributes such as age, gender, and internet type. Graphical representations aim to uncover patterns and the most influential factors on adaptability, with a focus on differences across demographic groups. These analyses provide insights to enhance online education experiences and strengthen adaptability strategies.

<table>
<tr><th>Name of Graph</th><th>Picture of Graph</th><th>Description</th></tr>
<tr><td>Bar plot</td><td>
</td><td>The graph shows that most students exhibit moderate adaptability to online education
(over 600 students), followed by those with low adaptability (over 400 students), while high adaptability represents the smallest proportion (100 students). This indicates that most students require additional support to enhance their adaptability.</td></tr>
<tr><td>Pie Chart</td><td>
</td><td>The chart shows that 57.7% of students rely on mobile data for online education, compared to 42.3% who use Wi-Fi.

This indicates a greater reliance on mobile data, highlighting the need to improve internet connectivity to better support online learning.</td></tr>
<tr><td>Bar chart</td><td>
</td><td>The bar chart illustrates the relationship between being an IT student and the type of device used for studies. It shows that most non-IT students heavily rely on mobile phones, with a smaller proportion using computers, and very limited use of tablets. On the other hand, IT students show a more balanced distribution, where mobile phones are still the most used, but a notable percentage also rely on computers, reflecting the demands of their specialization for more advanced devices. Overall, mobile phones are the most common device among all students, regardless of their specialization.</td></tr>
</table>

Pie Chart		The pie chart shows that 68.3% of students are enrolled in non-government institutions, making them the most common, while 31.7% are in government institutions. This suggests that non-government institutions play a larger role in education, potentially reflecting student preferences or greater options.
Bar chart		The chart shows that classes lasting 1-3 hours are the most common, attended by about 800 students, followed by 3-6 hours classes with around 200 students, while only 150 students did not attend any classes. This indicates a preference for medium-duration classes among students.
Pie Chart		The pie chart shows that 82.6% of students do not use self-learning management systems (Self LMS), while only 17.4% rely on them. This highlights the limited adoption of these systems and the need to raise awareness about their benefits to support independent learning and improve students' educational experiences.

<table>
<tr><td>Stacked bar</td><td>
</td><td>The chart shows that classes lasting 1-3 hours are the most common across all education levels, especially among school and college students. In universities, there is a balance between medium-duration classes (1-3 hours) and longer ones (3-6 hours), reflecting the demands of university education, while the proportion of students not attending any classes is minimal.</td></tr>
<tr><td>Bar chart</td><td>
</td><td>The chart shows that more than 800 students’ study in their hometown, while only about 200 students’ study outside their hometown, indicating that few students need to travel for their education.</td></tr>
<tr><td>Bar chart</td><td>
</td><td>The bar chart shows that male students exceed 600, making them the larger group, while female students number around 550.

Overall, the distribution reflects a relatively balanced representation between genders.</td></tr>
<tr><td>Bar chart</td><td>
</td><td>The bar chart shows that the 21-25 age group is the most common, with over 350 students, followed by the 11-15 and 16-20 age groups. Younger children (1-5, 6-10) and older adults (26-30) are less represented, with fewer than 100 students in each.</td></tr>
</table>

4. Data preprocessing:

- **Removing duplicates:**

```
# Check for duplicate rows
num_duplicates = df.duplicated().sum()
print("Number of duplicate rows:", num_duplicates , "\n")

df = df.drop_duplicates()
print("DataFrame after dropping all duplicate rows:\n")
print(df)
df.to_csv('Dataset/Cleaned_dataset.csv', index=False)
```

```
Number of duplicate rows: 949

DataFrame after dropping all duplicate rows:

     Gender  Age Education Level Institution Type IT Student Location  \
0       Boy   23      University   Non Government         No      Yes
1      Girl   23      University   Non Government         No      Yes
2      Girl   18         College       Government         No      Yes
3      Girl   13          School   Non Government         No      Yes
4      Girl   18          School   Non Government         No      Yes
...     ...  ...             ...              ...        ...      ...
1124    Boy   23      University   Non Government        Yes       No
1132    Boy   18         College       Government         No      Yes
1149   Girl   18         College   Non Government         No       No
1160    Boy   23      University   Non Government        Yes       No
1197    Boy   23      University   Non Government        Yes      Yes

     Load-shedding Financial Condition Internet Type Network Type  \
0              Low                 Mid          Wifi           4G
1             High                 Mid   Mobile Data           4G
2              Low                 Mid          Wifi           4G
3              Low                 Mid   Mobile Data           4G
4              Low                Poor   Mobile Data           3G
...            ...                 ...           ...          ...
1124          High                 Mid   Mobile Data           3G
1132           Low                 Mid   Mobile Data           3G
1149           Low                 Mid   Mobile Data           3G
1160          High                 Mid   Mobile Data           3G
1197           Low                 Mid   Mobile Data           4G

     Class Duration Self Lms    Device Adaptivity Level
0               3-6       No       Tab         Moderate
1               1-3      Yes    Mobile         Moderate
2               1-3       No    Mobile         Moderate
3               1-3       No    Mobile         Moderate
4                 0       No    Mobile              Low
...             ...      ...       ...              ...
1124            3-6       No  Computer              Low
1132            1-3       No    Mobile         Moderate
1149            1-3      Yes    Mobile              Low
1160            1-3      Yes    Mobile         Moderate
1197            3-6       No  Computer         Moderate

[256 rows x 14 columns]
```

Description:

Duplicates can lead to inaccuracies in analysis by artificially inflating certain statistics or biasing results. Removing duplicates helps maintain the integrity of your dataset and to give Accurate Model Training beside Duplicate entries can cause inconsistencies and removing the duplicates ensures the efficiency of the data to make reliable decisions.

- **Checking for missing values:**

```
missing_values = df.isna().sum()
print("\nTotal number of missing values in the dataset:", missing_values.sum())

# Creates a table that counts the number of missing values for each variable in the dataset
print("\nMissing Values:")
missing_table = pd.DataFrame({'Variable': missing_values.index, 'Missing Values': missing_values.values})
display(missing_table)
```

```
Total number of missing values in the dataset: 0

Missing Values:
```

	Variable	Missing Values
0	Gender	0
1	Age	0
2	Education Level	0
3	Institution Type	0
4	IT Student	0
5	Location	0
6	Load-shedding	0
7	Financial Condition	0
8	Internet Type	0
9	Network Type	0
10	Class Duration	0
11	Self Lms	0
12	Device	0
13	Adaptivity Level	0

Description:
Identifying and addressing missing values in datasets is crucial for maintaining the integrity and reliability of data analysis. Missing values can compromise statistical estimates and lead to misleading conclusions. Analyzing missing data patterns helps refine data collection strategies, ensuring more accurate and robust analysis outcomes.

- **Detecting the outliers:**

```python
df = pd.read_csv('Dataset/Cleaned_dataset.csv')

# Detect and count outliers in numerical columns using IQR

def detect_outliers_iqr(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column_name] < lower_bound) | (df[column_name] > upper_bound)]
    return outliers

# List of numerical columns
numeric_columns = ['Age']

# Detect and count outliers in numerical columns
num_outliers_numeric = 0
for col in numeric_columns:
    outliers = detect_outliers_iqr(df, col)
    num_outliers_numeric += outliers.shape[0]
    print(f"Number of outliers in '{col}': {outliers.shape[0]}")



# Detect and count outliers in categorical columns using frequency analysis

# Function to detect rare categories in categorical columns
def detect_outliers_categorical(df, column_name, threshold=5):
    value_counts = df[column_name].value_counts()
    rare_categories = value_counts[value_counts < threshold]
    return rare_categories

# List of categorical columns
categorical_columns = ['Education Level', 'Financial Condition', 'Network Type', 'Class Duration', 'Device']

# Detect and count outliers in categorical columns
num_outliers_categorical = 0
for col in categorical_columns:
    rare_categories = detect_outliers_categorical(df, col)
    num_outliers_categorical += rare_categories.shape[0]
    print(f"Number of rare categories in '{col}': {rare_categories.shape[0]}")

total_outliers = num_outliers_numeric + num_outliers_categorical
print(f"Total number of outliers: {total_outliers}")
```

```
Number of outliers in 'Age': 0
Number of rare categories in 'Education Level': 0
Number of rare categories in 'Financial Condition': 0
Number of rare categories in 'Network Type': 0
Number of rare categories in 'Class Duration': 0
Number of rare categories in 'Device': 0
Total number of outliers: 0
```

Description:
Since there were no outliers detected in the dataset, no adjustments were necessary. The data appears to be consistent, with no extreme values that would significantly impact the analysis. This ensures that the dataset is already well-suited for modeling and that the results we obtain will be more accurate and reliable.

- **Data Transformation:**

1. **Encoding:**

```python
#importing LabelEncoder for encoding and StandardScaler for normalizing the data from sklearn library
from sklearn.preprocessing import LabelEncoder , StandardScaler

df = pd.read_csv('Dataset/Cleaned_dataset.csv')
encoder  = LabelEncoder()

df['Education Level'] = encoder.fit_transform(df['Education Level'])
df['Institution Type'] = encoder.fit_transform(df['Institution Type'])
df['Gender'] = encoder.fit_transform(df['Gender'])
df['Location'] = encoder.fit_transform(df['Location'])
df['Financial Condition'] = encoder.fit_transform(df['Financial Condition'])
df['Internet Type'] = encoder.fit_transform(df['Internet Type'])
df['Network Type'] = encoder.fit_transform(df['Network Type'])
df['Device'] = encoder.fit_transform(df['Device'])
df['IT Student'] = encoder.fit_transform(df['IT Student'])
df['Adaptivity Level'] = encoder.fit_transform(df['Adaptivity Level'])
df['Self Lms'] = encoder.fit_transform(df['Self Lms'])
df['Load-shedding'] = encoder.fit_transform(df['Load-shedding'])
df['Class Duration'] = encoder.fit_transform(df['Class Duration'])


#Display the data after Encode it
print("Dataset after Encoding : ")
print(df)

df.to_csv('Dataset/Cleaned_dataset.csv', index=False)
```

```
Dataset after Encoding :
     Gender  Age  Education Level  Institution Type  IT Student  Location  \
0         0   23                2                 1           0         1
1         1   23                2                 1           0         1
2         1   18                0                 0           0         1
3         1   13                1                 1           0         1
4         1   18                1                 1           0         1
..      ...  ...              ...               ...         ...       ...
251       0   23                2                 1           1         0
252       0   18                0                 0           0         1
253       1   18                0                 1           0         0
254       0   23                2                 1           1         0
255       0   23                2                 1           1         1

     Load-shedding  Financial Condition  Internet Type  Network Type  \
0                1                    0              1             2
1                0                    0              0             2
2                1                    0              1             2
3                1                    0              0             2
4                1                    1              0             1
..             ...                  ...            ...           ...
251              0                    0              0             1
252              1                    0              0             1
253              1                    0              0             1
254              0                    0              0             1
255              1                    0              0             2

     Class Duration  Self Lms  Device  Adaptivity Level
0                 2         0       2                 2
1                 1         1       1                 2
2                 1         0       1                 2
3                 1         0       1                 2
4                 0         0       1                 1
..              ...       ...     ...               ...
251               2         0       0                 1
252               1         0       1                 2
253               1         1       1                 1
254               1         1       1                 2
255               2         0       0                 2

[256 rows x 14 columns]
```

Description:
This encoding method provides a numerical representation for 'Education Level', 'Institution Type', 'Gender', 'Location', 'Financial Condition', 'Internet Type', 'Network Type', 'Device', 'IT Student', 'Adaptivity Level', 'Self Lms', 'Load-shedding', and 'Class Duration', where assigning the values 0,1, and 2 helps standardize variables for computational purposes. This enables easier processing and analysis of the attributes data in various algorithms and models. [2]

2. Normalization:

```python
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv('Dataset/Cleaned_dataset.csv')

scaler = MinMaxScaler()

df['Age'] = scaler.fit_transform(df[['Age']])

df.to_csv('Dataset/Cleaned_dataset.csv', index=False)
print("Age column after Normalization : ")
print(df)
```

```
Age column after Normalization :
     Gender  Age  Education Level  Institution Type  IT Student  Location  \
0         0  0.8                2                 1           0         1
1         1  0.8                2                 1           0         1
2         1  0.6                0                 0           0         1
3         1  0.4                1                 1           0         1
4         1  0.6                1                 1           0         1
..      ...  ...              ...               ...         ...       ...
251       0  0.8                2                 1           1         0
252       0  0.6                0                 0           0         1
253       1  0.6                0                 1           0         0
254       0  0.8                2                 1           1         0
255       0  0.8                2                 1           1         1

     Load-shedding  Financial Condition  Internet Type  Network Type  \
0                1                    0              1             2
1                0                    0              0             2
2                1                    0              1             2
3                1                    0              0             2
4                1                    1              0             1
..             ...                  ...            ...           ...
251              0                    0              0             1
252              1                    0              0             1
253              1                    0              0             1
254              0                    0              0             1
255              1                    0              0             2

     Class Duration  Self Lms  Device  Adaptivity Level
0                 2         0       2                 2
1                 1         1       1                 2
2                 1         0       1                 2
3                 1         0       1                 2
4                 0         0       1                 1
..              ...       ...     ...               ...
251               2         0       0                 1
252               1         0       1                 2
253               1         1       1                 1
254               1         1       1                 2
255               2         0       0                 2

[256 rows x 14 columns]
```

Description:
Here in normalization, we normalize the 'Age' column using Min-Max Scaling method since Age has a fixed and bounded range (1-30). This helps us to format all the 'Age' values in the dataset and facilities the analysis process.[2]

3. Aggregation:

We decided not to apply aggregation to our dataset because the type of analysis and insights we seek donot benefit significantly from summarizing the data into higher-level metrics. Additionally, our dataset contains only one numeric variable, which is Age. Aggregating this single numeric feature would not provide meaningful insights, our dataset primarily focuses on understanding individual student adaptability levels in online education this require detailed data rather than aggregated summaries, as we aim to capture patterns at the individual level, rather than at the group level. Therefore, aggregation would not add significant value to our specific goals.

4. Discretization:

```
data = pd.read_csv('Dataset/Cleaned_dataset.csv')

# Map class duration ranges to numeric equivalents
duration_mapping = {
    "0": 0,
    "1-3": 2,
    "3-6": 4.5
}

# Replace the 'Class Duration' column with numeric equivalents
data['Class Duration'] = data['Class Duration'].replace(duration_mapping)

# Define bins for discretization
bins = [0, 1, 3, 6]  # Bins: 0-1 (No class), 1-3 (2hours), 3-6 (3hours)
labels = ['No class', '2 hours', '3 hours']

# Apply "cut()" method to discretize the numeric class durations
data['Class Duration'] = pd.cut(data['Class Duration'], bins=bins, labels=labels, include_lowest=True)

# Display the discretized data
print("Class Duration column after Discretization : ")
print(data['Class Duration'])

# Save the tranmissioning result into new dataset
data.to_csv('Dataset/AfterTransmission_dataset.csv', index=False)
```

```
Class Duration column after Discretization :
0       2 hours
1      No class
2      No class
3      No class
4      No class
         ...
251     2 hours
252    No class
253    No class
254    No class
255     2 hours
Name: Class Duration, Length: 256, dtype: category
Categories (3, object): ['No class' < '2 hours' < '3 hours']
```

Description:
In the discretization method, we categorize ordinal class duration values into three groups: No class (0), 2 hours (1-3), and 3 hours (3-6). By simplifying data, the variability in the class duration was summarized, making the results more interpretable for stakeholders and enhancing data usability.

Raw data:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

df = pd.read_csv('Dataset/students_adaptability_level_online_education.csv')
df
```

	Gender	Age	Education Level	Institution Type	IT Student	Location	Load-shedding	Financial Condition	Internet Type	Network Type	Class Duration	Self Lms	Device	Adaptivity Level
0	Boy	21-25	University	Non Government	No	Yes	Low	Mid	Wifi	4G	3-6	No	Tab	Moderate
1	Girl	21-25	University	Non Government	No	Yes	High	Mid	Mobile Data	4G	1-3	Yes	Mobile	Moderate
2	Girl	16-20	College	Government	No	Yes	Low	Mid	Wifi	4G	1-3	No	Mobile	Moderate
3	Girl	11-15	School	Non Government	No	Yes	Low	Mid	Mobile Data	4G	1-3	No	Mobile	Moderate
4	Girl	16-20	School	Non Government	No	Yes	Low	Poor	Mobile Data	3G	0	No	Mobile	Low
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
00	Girl	16-20	College	Non Government	No	Yes	Low	Mid	Wifi	4G	1-3	No	Mobile	Low
01	Girl	16-20	College	Non Government	No	No	High	Mid	Wifi	4G	3-6	No	Mobile	Moderate
02	Boy	11-15	School	Non Government	No	Yes	Low	Mid	Mobile Data	3G	1-3	No	Mobile	Moderate
03	Girl	16-20	College	Non Government	No	No	Low	Mid	Wifi	4G	1-3	No	Mobile	Low
04	Girl	11-15	School	Non Government	No	Yes	Low	Poor	Mobile Data	3G	1-3	No	Mobile	Moderate

5 rows × 14 columns

Data after processing:

```python
import pandas as pd
from scipy import stats
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn import tree

df = pd.read_csv("Dataset/Processed_dataset.csv")
df
```

Gender	Age	Education Level	Institution Type	IT Student	Location	Load-shedding	Financial Condition	Internet Type	Network Type	Class Duration	Self Lms	Device	Adaptivity Level
0	0.800000	2	1	0	1	1	0	1	2	0	0	2	1
1	0.800000	2	1	0	1	0	0	0	2	1	1	1	1
1	0.600000	0	0	0	1	1	0	1	2	1	0	1	1
1	0.400000	1	1	0	1	1	0	0	2	1	0	1	1
1	0.600000	1	1	0	1	1	1	0	1	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	0.600000	0	0	0	0	1	0	0	1	1	0	1	0
0	0.400000	1	0	0	0	1	0	0	1	1	0	1	0
0	0.339046	1	0	0	0	1	0	0	1	1	0	1	0
1	0.600000	0	1	0	1	0	0	1	2	1	0	1	0
0	0.400000	1	1	0	1	1	0	0	2	1	0	1	0

ws × 14 columns

- **Balance Data:**

Before starting the Data Mining Technique, we investigated whether the data was balanced or not:

```
Number of high : 24
Number of moderate : 118
Number of low : 114
___
Percentage of high : 9.38%
Percentage of moderate : 46.09%
Percentage of low : 44.53%
```

In the beginning, we reviewed the percentage for each of the three classes in the Adaptivity Level (High, Low, Moderate Adaptivity Level), and we noticed that the percentage is imbalanced (not ranging between 40% to 60%).

- Process of correcting data balancing

```
Adaptivity Level_Binary
1    142
0    142
Name: count, dtype: int64
```

- Data after the balancing process:

By analyzing the counts of the "high" and "low" adaptivity levels, we ensured balance between the two classes. This was achieved by calculating the number of samples in each class and their respective percentages. <u>This process prevents the model from being biased towards the majority class, improving its ability to generalize to new data.</u>

```
Number of high : 142
Number of low: 142
___
Percentage of high : 50.00%
Percentage of low : 50.00%
```

We then calculated the percentage of each class to ensure that the data was balanced. Both adaptivity levels ("high" and "low") were equally represented, with the percentages for each class being comparable, thus confirming the balance in the dataset.

5- **Data Mining Technique:**

We employed both supervised and unsupervised learning methods on our dataset using classification and clustering techniques.

For the classification task, we used a decision tree. This recursive algorithm creates a tree-like structure where each leaf node represents a final prediction. Our model is designed to predict the adaptability level of students, categorizing them into "high" or "low" adaptivity levels (1 for high and 0 for low). The decision tree makes predictions based on several features, including age and other relevant attributes in the dataset. Since classification is a supervised learning technique, we used a training dataset to train the model and a testing dataset to evaluate its performance. We experimented with different training data splits of 70%, 60%, and 80%, and utilized attribute selection measures like Information Gain (IG) and Gini index. To evaluate the model's performance, we calculated its accuracy and used a confusion matrix to summarize key metrics such as sensitivity, specificity, precision, and error rate.

For the clustering process, which falls under unsupervised learning, we excluded the "adaptivity level" labels since clustering doesn't rely on class labels. We used all other attributes in the dataset, we first encoded all categorical and ordinal attributes into numerical values during the preprocessing step. This transformation was essential since clustering algorithms, such as K-means, require numerical data. After that, we utilized all attributes, now represented as numerical values, for the K-means clustering algorithm. The algorithm assigns data points to K clusters, each represented by a centroid, and iteratively recalculates the centroids until they stabilize, ensuring accurate cluster assignment.

To validate the clustering results, we calculated the average silhouette score for each cluster and visualized these scores. Additionally, we used the WSS (Within-Cluster Sum of Squares) method to compare different cluster sizes and determine the optimal number of clusters by assessing the compactness and separation of the clusters.

6- **Evaluation and Comparison:**

- Classification [70% training, 30% testing] Information Gain:

Figure (1) (decision tree):

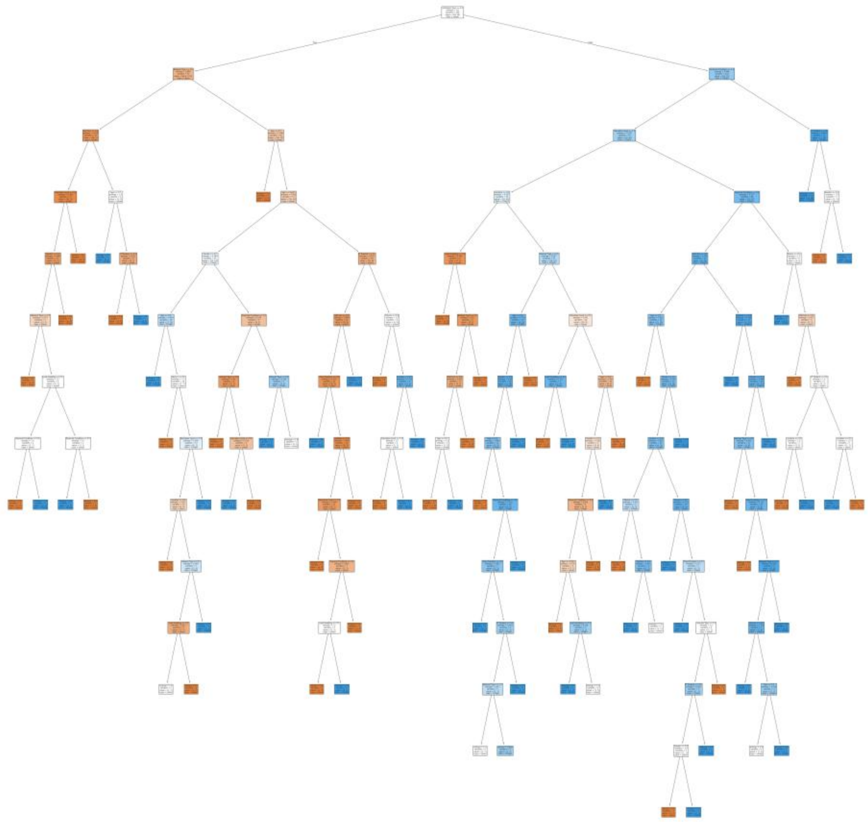
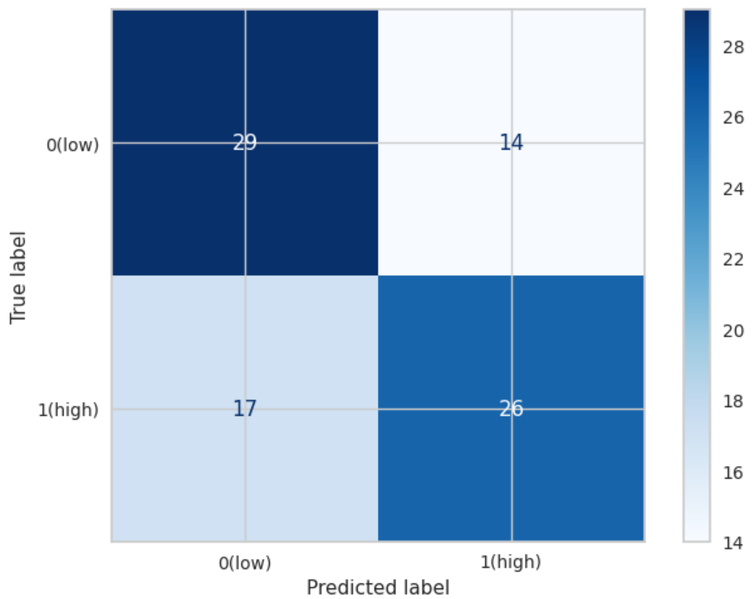


Figure (2) (confusion matrix):



- Classification [60% Training and 40% testing] Information Gain:

Figure (1) (decision tree):

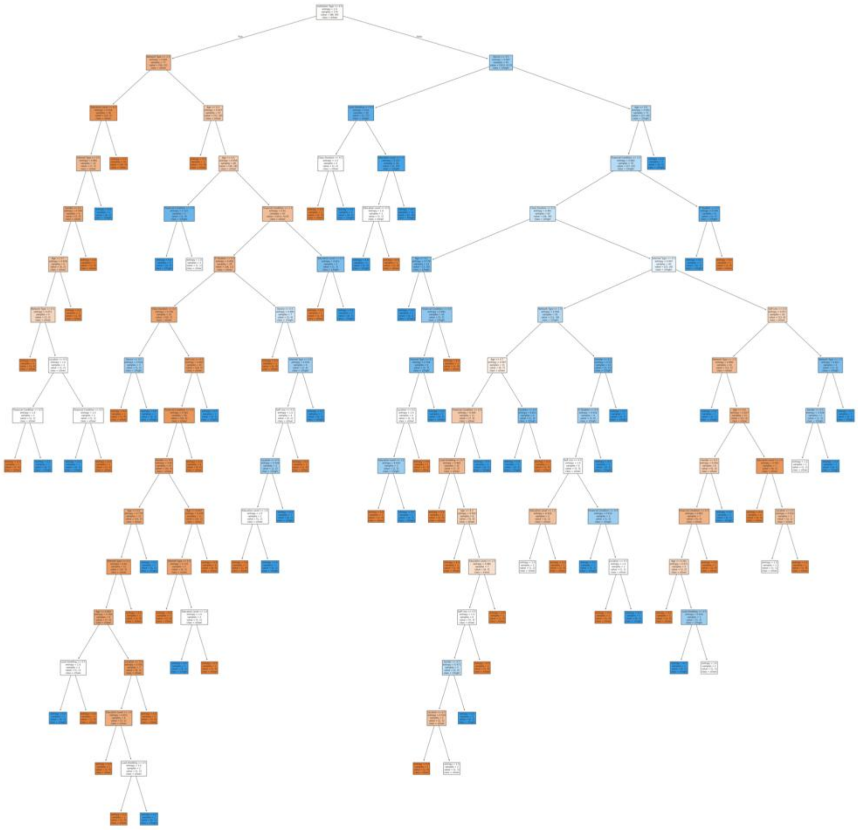
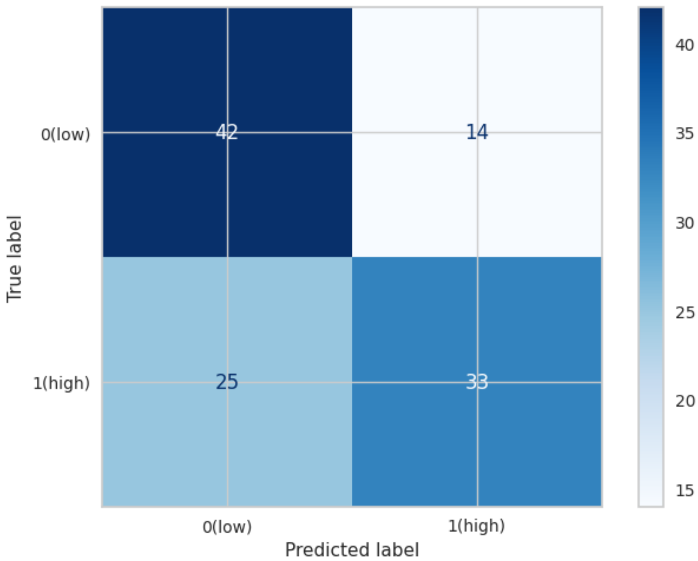


Figure (2) (confusion matrix):



- Classification [80% training and 20% testing] Information Gain:

Figure (1) (decision tree):

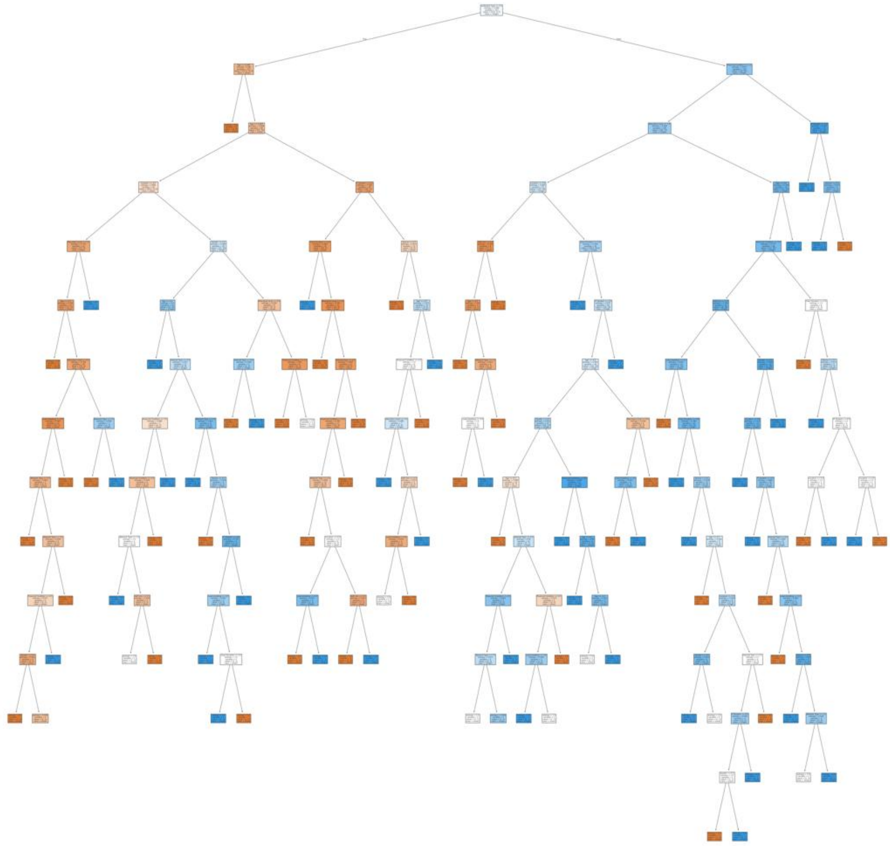
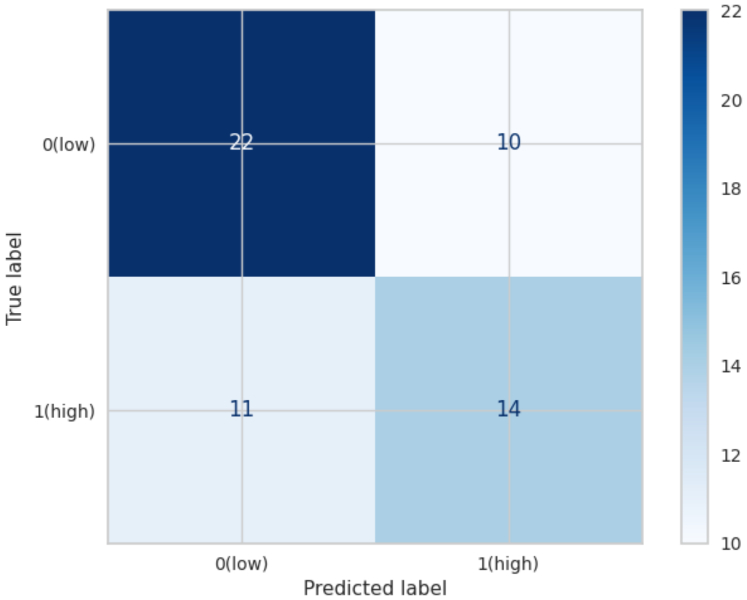


Figure (2) (confusion matrix):



Mining task	Comparison Criteria
Classification for Information Gain	We tried 3 different sizes for dataset splitting to create the decision tree:

- 70% Training data, 30% Test data.

Accuracy	63%
Precision	65%
Sensitivity	60%
Specificity	67%
Error rate	36%

- 60% Training data, 40% Test data.

Accuracy	65.79%
Precision	70.21%
Sensitivity	56.90%
Specificity	75%
Error rate	34.21%

- 80% Training data, 20% Test data.

Accuracy	63%
Precision	58.33%
Sensitivity	56%
Specificity	68.75%
Error rate	36%

- Classification [70% training, 30% testing] Gini Index:

Figure (1) (decision tree):

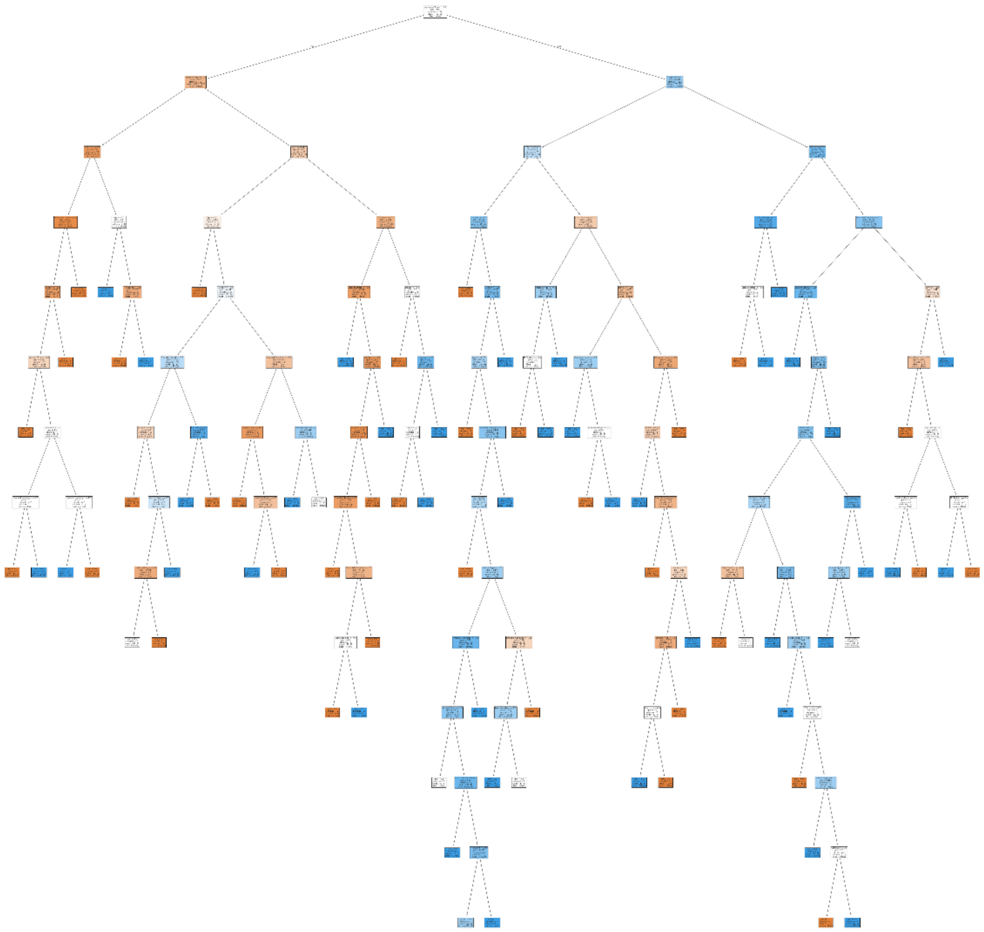
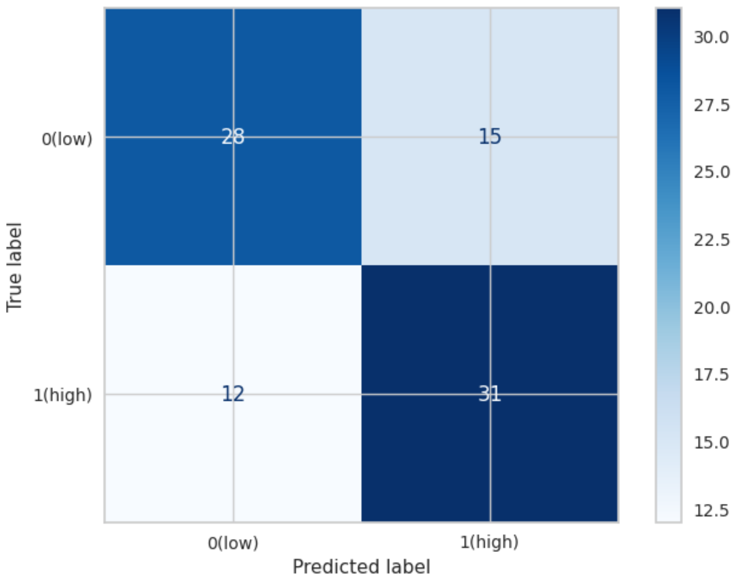


Figure (2) (confusion matrix):



- Classification [60% training, 40% testing] Gini Index:

Figure (1) (decision tree):

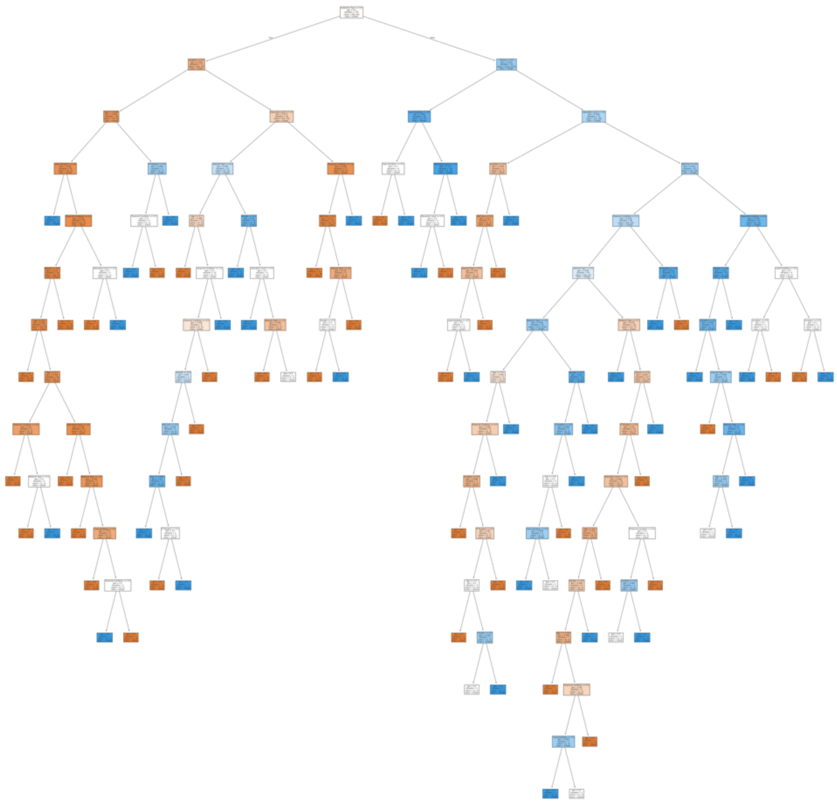
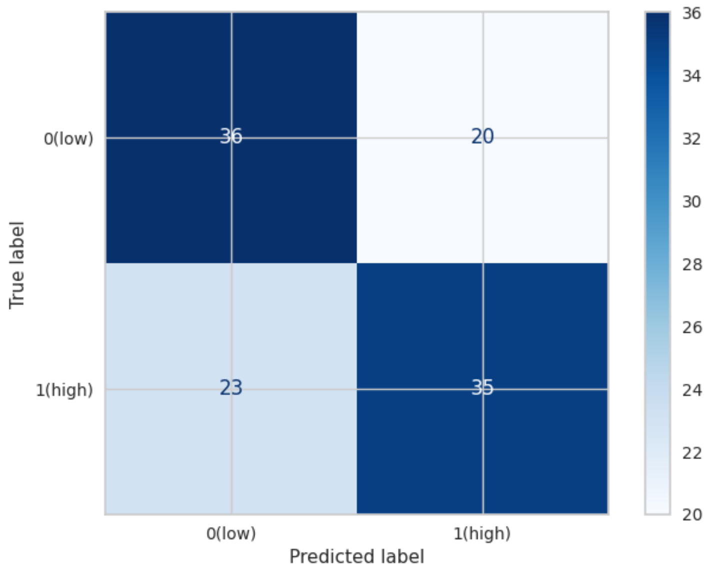


Figure (2) (confusion matrix):



- Classification [80% training, 20% testing] Gini Index:

Figure (1) (decision tree):

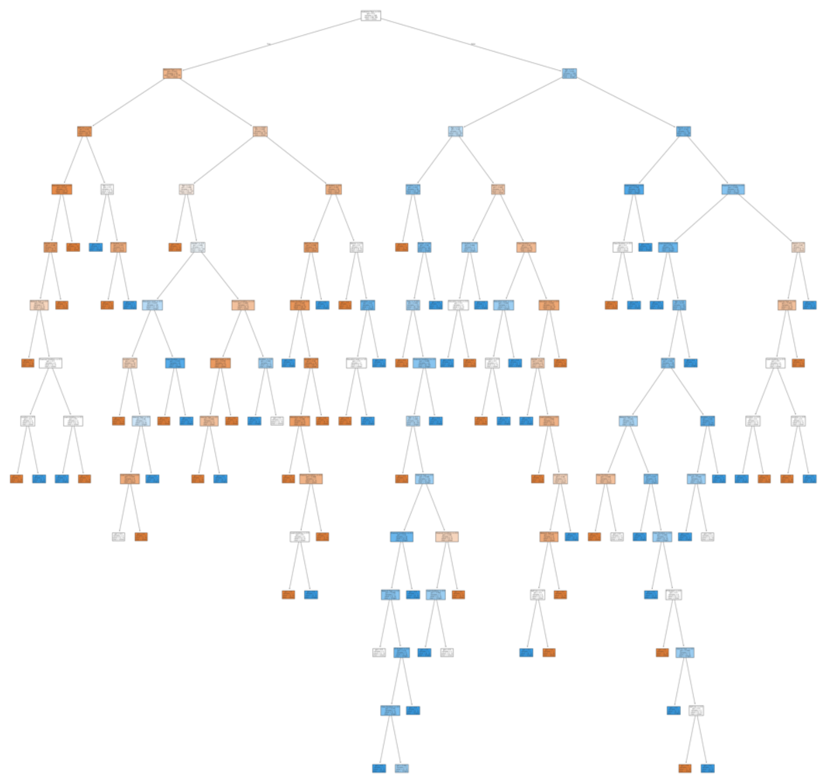
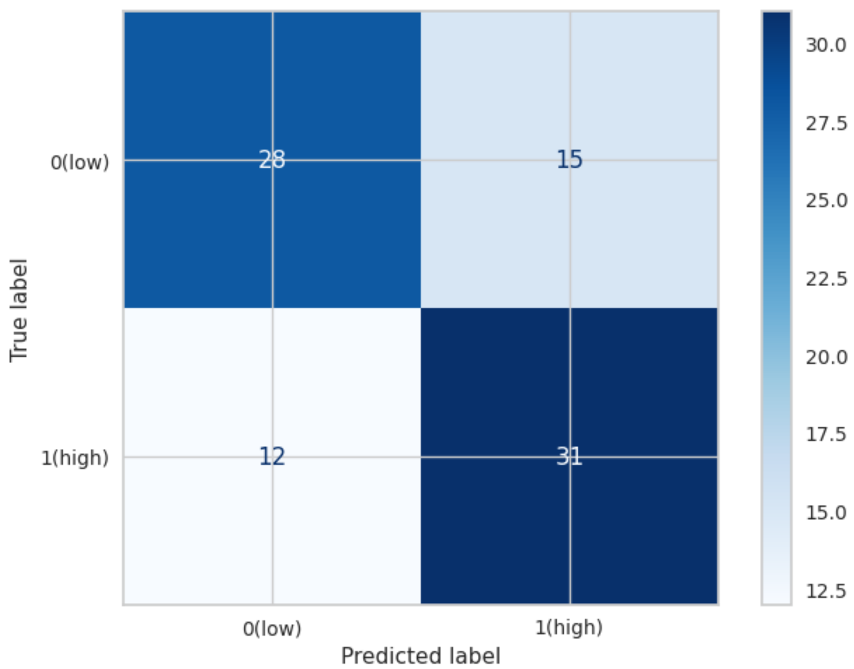


Figure (2) (confusion matrix):



Mining task	Comparison Criteria
Classification for Gini Index	We tried 3 different sizes for dataset splitting to create the decision tree:

- 70% Training data, 30% Test data.

Accuracy	68%
Precision	67%
Sensitivity	72%
Specificity	65%
Error rate	31%

- 60% Training data, 40% Test data.

Accuracy	62%
Precision	64%
Sensitivity	60%
Specificity	64%
Error rate	38%

- 80% Training data, 20% Test data.

Accuracy	68%
Precision	67%
Sensitivity	72%
Specificity	65%
Error rate	31%

- **The better partitioning:**

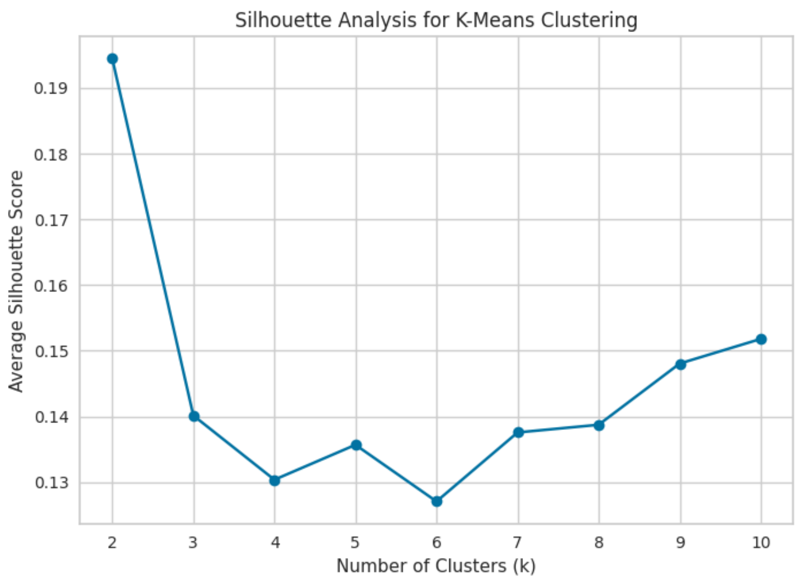
In summary, the models trained with 70% training, 30% testing and 80% training, 20% testing are the most effective across several key metrics, including accuracy, sensitivity, specificity, and precision. These models are more balanced in terms of both positive and negative classifications, while the 60% training, 40% testing model has the highest error rate and slightly lower performance in comparison.

- **Clustering**

We choose 3 different sizes [2,3,10] based on the result of the validation methods that we will apply then we will use these sizes to perform the k-means clustering.

Silhouette method:

The Silhouette method is a technique used to evaluate the quality of clustering results. It measures how well each data point fits within its assigned cluster compared to neighboring clusters.



Elbow method:

The Elbow method is a technique used to determine the optimal number of clusters in a dataset for K-means clustering.

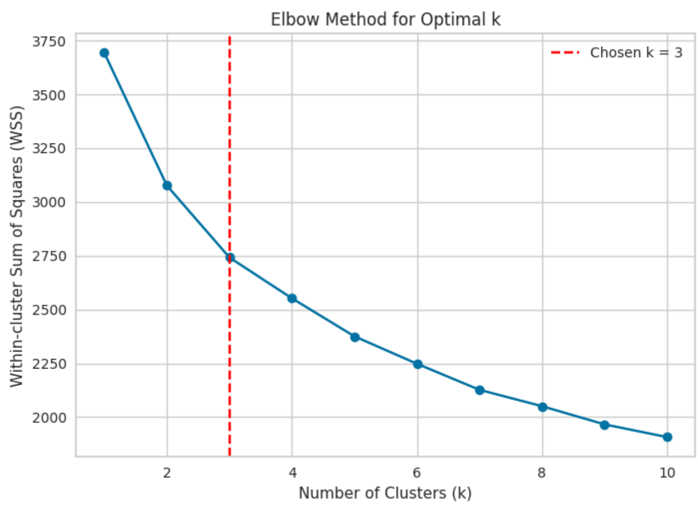


Figure (1): silhouette scores [K=2]

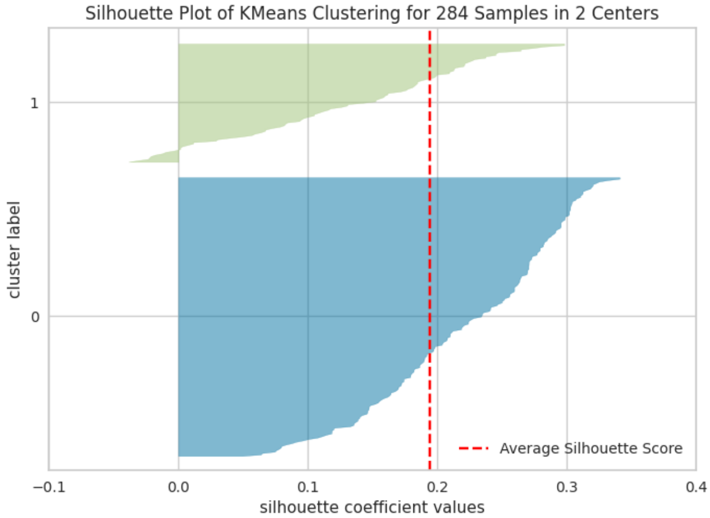


Figure (2): silhouette scores [K=3]

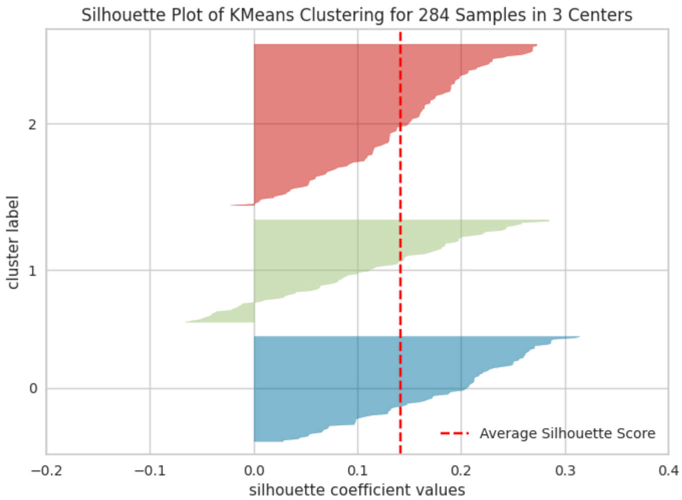
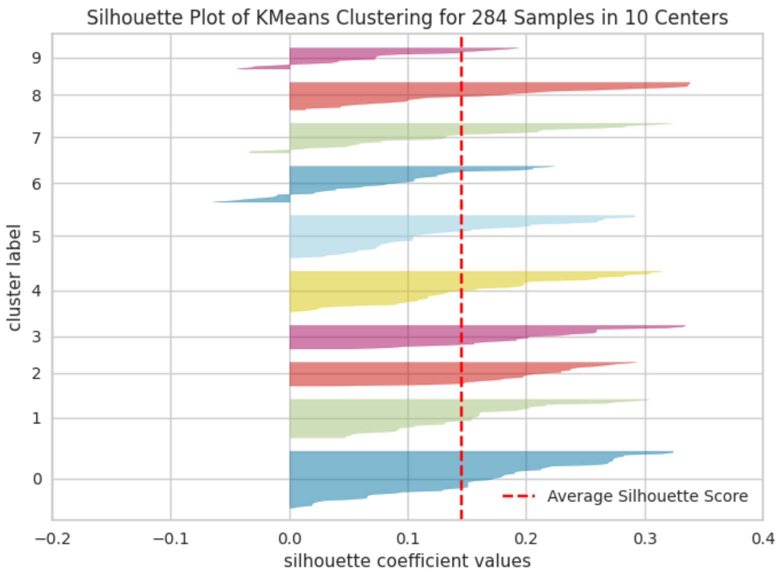


Figure (3): silhouette scores [K=10]



Mining task	Comparison Criteria			
Clustring	We tried 3 different sizes for dataset splitting to create the decision tree: K=2, K=3, K=10			
	No. of clusters	K=2	K=3	K=10
	Average Silhouette width	0.1944	0.1434	0.1433
	total within-cluster sum of square	3075.963	2741.921	1907.36

7. Findings:

Initially, we selected a dataset representing students' adaptability levels in online education with the aim of understanding the factors that influence their success in this environment and identifying potential areas for improvement.

To ensure accuracy, reliability, and precision in our results, we applied several data preprocessing techniques to enhance the dataset's efficiency. Using various visualization methods, such as bar plots, Pie Chart, and Stacked bar, we clarified the data and made it easier to interpret. This allowed us to apply appropriate data processing techniques. Based on these visualizations and other analyses, we ensured that the dataset was free from any outliers or missing values that could negatively impact the results.

Furthermore, we implemented data transformations, including normalization, and balanced data process to assign equal weight to certain features and streamline data processing during mining tasks.

Consequently, we conducted data mining tasks, encompassing classification and partitioning. For classification, we employed the Gini index and information gain metrics. Experimenting with three different sizes of training and testing data allowed us to achieve optimal results for both model construction and evaluation. Here are our findings:

	70% training, 30% testing	60% training, 40% testing	80% training, 20% testing
Accuracy	0.6395348837209303	0.6578947368421053	0.631578947368421
Error Rate	0.36046511627906974	0.3421052631578947	0.368421052631579
Sensitivity	0.6046511627906976	0.5689655172413793	0.56
Specificity	0.6744186046511628	0.75	0.6875
Precision	0.65	0.7021276595744681	0.5833333333333334

- **Information Gain:**

Based on the results for the models trained using different training/testing splits, the following observations can be made:

- Accuracy: The model trained with the 60% training set and 40% testing set achieved the highest accuracy (65.79%), indicating it performed slightly better overall.
- Error Rate: The model trained with the 80% training set and 20% testing set exhibited the highest error rate (36.84%), suggesting the 70-30 split has a lower error rate and better performance in reducing classification errors.
- Sensitivity: The 70% training set and 30% testing set model achieved the highest sensitivity (60.47%), meaning it is better at identifying positive instances.
- Specificity: The model trained with the 60% training set and 40% testing set obtained the highest specificity (75%), indicating better performance in correctly identifying negative instances.
- Precision: The model trained with the 60% training set and 40% testing set achieved the highest precision (70.21%), meaning it has the best accuracy when predicting positive outcomes.

In summary, the model trained with a **60% training set and 40% testing set** generally performs better across various evaluation metrics compared to the other partitioning schemes.

	70% training, 30% testing	60% training, 40% testing	80% training, 20% testing
Accuracy	0.686046511627907	0.6228070175438597	0.686046511627907
Error Rate	0.313953488372093	0.3771929824561403	0.313953488372093
Sensitivity	0.7209302325581395	0.603448275862069	0.7209302325581395
Specificity	0.6511627906976745	0.6428571428571429	0.6511627906976745
Precision	0.6739130434782609	0.6363636363636364	0.6739130434782609

- **Gini index:**

Based on the results for the models trained using the Gini Index criterion, the following observations can be made:

- Accuracy: The models trained with 70% training, 30% testing and 80% training, 20% testing achieved the highest accuracy (68.61%), showing better overall performance compared to the 60% training, 40% testing model (62.28%).

- Error Rate: The 60% training, 40% testing model has the highest error rate (37.72%), while the 70% training, 30% testing and 80% training, 20% testing models have the lowest error rate (31.40%).

- Sensitivity: The 70% training, 30% testing and 80% training, 20% testing models achieved the highest sensitivity (72.09%), meaning they are more effective at identifying positive instances.

- Specificity: The models trained with 70% training, 30% testing and 80% training, 20% testing achieved the highest specificity (65.12%), indicating better performance in correctly identifying negative cases.

- Precision: The 70% training, 30% testing and 80% training, 20% testing models achieved the highest precision (67.39%), indicating they are more accurate in predicting positive outcomes.

In summary, the models trained with **70% training, 30% testing and 80% training, 20% testing** are the most effective across several key metrics, including accuracy, sensitivity, specificity, and precision. These models are more balanced in terms of both positive and negative classifications based on the results provided.

- **The best model between information gain and the Gini index:**

After selecting the best model split from Information Gain, which was 60% training, 40% testing, and the best split from Gini Index, which was both the 70% training, 30% testing, and 80% training, 20% testing, we reviewed the values of each for comparison between Information Gain and Gini Index, and we reached the following conclusion:

	Information gain	Gini index
Accuracy	0.6578947368421053	0.686046511627907
Error Rate	0.3421052631578947	0.313953488372093
Sensitivity	0.5689655172413793	0.7209302325581395
Specificity	0.75	0.6511627906976745
Precision	0.7021276595744681	0.6739130434782609

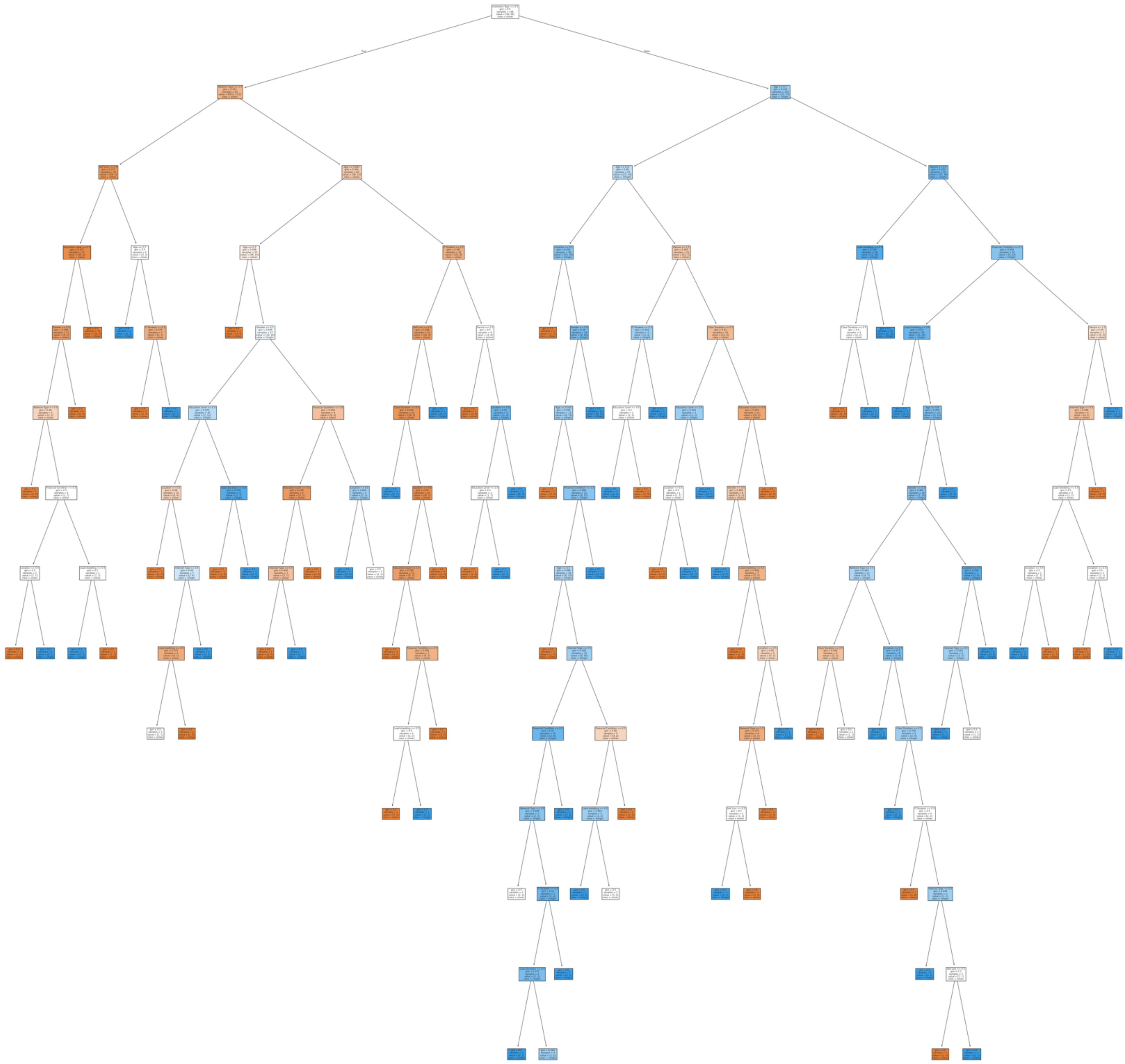
- **Accuracy and Error Rate:** The Gini Index split demonstrates higher accuracy (68.60% or 0.686) compared to Information Gain (65.79% or 0.658), resulting in a lower error rate of 31.39% (0.313) for Gini Index versus 34.21% (0.342) for Information Gain. This suggests that the Gini Index model classifies cases more accurately, making it more dependable.

- **Sensitivity and Specificity:** The Gini Index split outperforms in sensitivity (72.09% or 0.7209) compared to Information Gain (56.90% or 0.569). However, Information Gain achieves higher specificity (75% or 0.75) than Gini Index (65.16% or 0.6516). Sensitivity reflects the ability to identify positive cases, where Gini Index excels, while Information Gain is better at predicting negative cases due to its higher specificity.

- **Precision:** Information Gain shows higher precision (70.21% or 0.702) compared to Gini Index (67.39% or 0.673). This indicates that when the model predicts positive cases, Information Gain is correct more often than Gini Index.

Conclusion: The 80%-20% split using the Gini Index offers better overall performance with higher accuracy, lower error rate, and superior sensitivity. However, if specificity and precision are more critical, Information Gain could be a better choice. For a balanced approach prioritizing classification accuracy, the Gini Index model is the preferred option.

This was the decision tree associated with this division:

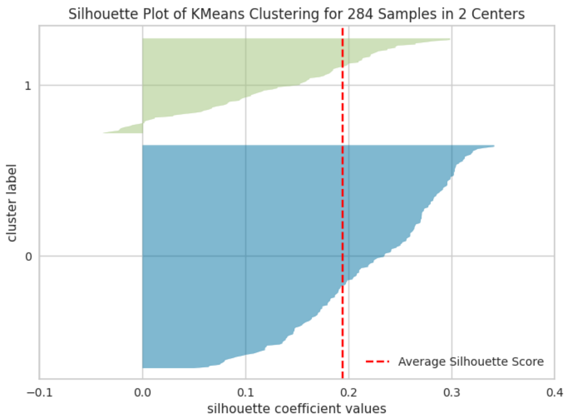


In this tree, the splitting process begins with the criterion of Institution Type, where samples are segregated based on their Institution Type values. The selection of features at each node is determined by their Gini values. Following the split on Institution Type, the tree considers the Network Type and Age, dividing samples accordingly. Subsequently, the Self Lms is examined, leading to further division of samples. This splitting procedure persists for each attribute, guided by their respective values at each level, until reaching the leaf nodes. These leaf nodes act as terminal points, providing the final classification (whether 'low' or 'high' adaptivity level) based on the path followed through the tree.

For Clustering, we used K-means algorithm with 3 different K to find the optimal number of clusters, we calculated the average silhouette width for each K, and we concluded the following results:

	K=3	K=2	K=10
WSS	2741.921553	3075.963028	1907.360703
Average Silhouette Score	0.143437	0.194434	0.143397

We've decided that K=2 is the best choice for our clustering model based on the metrics we've analyzed(WSS, Average Sihouette Score, Visualization of K-mean). This choice is because K=2 gives the highest silhouette width, also k=2 have a highest value of WSS Comparison of WSS value for K=3,k=10 Also, having a silhouette plot of kmeans clustring of 284 samples of 2 centers was one of the most important criteria for choosing k=2 as the best k, indicating that it creates distinct and cohesive clusters. And this was the corresponding chart:



From the graph of KMeans Clustering for 284Samples in 2 Centers, the fact that most of the silhouette scores with a positive value reinforces the notion that the samples are well-matched to their clusters and are distant from neighboring clusters. This indicates that the clustering solution has successfully separated the data points into distinct and well-defined clusters. Note that while most silhouette scores being positive is a positive indicator, it does not necessarily imply that the clustering solution is "extremely perfect" or flawless. There might still be some degree of overlap or ambiguity between clusters, especially if there are samples as above in the first center with silhouette scores close to 0 or negative values

---

Finally, both models have proven valuable in predicting the level of flexibility exhibited by students, thereby contributing significantly to our overarching goal of assisting individuals in adapting to online learning environments. However, given that our dataset includes a class label " Adaptivity Level " supervised learning models, particularly classification models, are deemed more accurate and suitable for application. Supervised learning approaches are more accurate than unsupervised learning model(clustering), as the expected output is known beforehand this way we make use of the class label attribute. We harness this existing knowledge to refine the accuracy and relevance of our predictive models, empowering students to make informed decisions about their learning strategies and adaptability in online educational settings.

8. References:

- [1] Md. Mahmudul Hasan, "Students Adaptability Level in Online Education", Kaggle, Available: https://www.kaggle.com/datasets/mdmahmudulhasansuzan/students-adaptability-level-in-online-education

- [2] “Labs and Lecture Slides," College of Computer Science, Department of Information Technology, King Saud University.