King Saud University
College of Computer and Information Sciences
Information Technology Department
IT326: Data Mining
Project "Students Exam Scores"

Section number: 56544	ID
Raghad Aldajani	443200551
Rana Alharbi	442202346
Nora Albyahi	443200479
Mariam Ahmed	443204628
Sadeem albargash	443200679

1. Problem

We want to solve the problem of limited personalized educational support for students in core subjects. This is important because it directly impacts educational equity, future opportunities for students, and the overall progress of society. By working together to develop innovative solutions, we can help bridge the educational gap and empower all students to succeed.

2. Data Mining Task

In our project, we addressed the problem as a data mining task utilizing **classification** and **clustering** to analyze student performance.
For **classification**, we aimed to predict students' performance levels (high or low) using the target variable **Average Score**, which was converted into binary classes based on a defined threshold (e.g., 0.07). This task helped us identify the main factors influencing performance and provided insights to support underperforming students.
For **clustering**, we grouped students based on their attributes, excluding **Average Score**, to uncover natural patterns and similarities within the data. This unsupervised approach allowed us to identify distinct student groups, enabling the creation of personalized educational strategies.
By integrating both classification and clustering, our project offered a well-rounded analysis of student performance, supporting data-driven interventions and informed decision-making.

3. Data

Dataset Description and Analysis
Dataset Overview

- **Source**: Kaggle - Students Exam Scores Dataset [1].
- **Number of Objects (Records)**: 30,641 students
- **Number of Attributes**: 14 attributes
- **Main Characteristics of Attributes**:

Attribute Name	Description	Data Type	Possible Values
Gender	The gender of the student.	Categorical	"Male", "Female"
EthnicGroup	The ethnic group of the student.	Categorical	Group A, B, C, D, E
ParentEduc	The education level of the parent(s).	Categorical	"Some High School", "High School", "Some College", "Bachelor's", "Master's"
LunchType	The type of lunch received by the student.	Categorical	"Standard", "Free/Reduced"
TestPrep	Completion of test preparation courses.	Categorical	"Completed", "None"
ParentMaritalStatus	The marital status of the parent(s).	Categorical	"Married", "Single", "Widowed", "Divorced"
PracticeSport	The frequency of the student's sports activities.	Categorical	"Never", "Sometimes", "Regularly"
IsFirstChild	Indicates if the student is the first child in the family.	Binary	"Yes", "No"
NrSiblings	The number of siblings the student has.	Numeric	0 to 7
TransportMeans	The student's primary means of transportation to school.	Categorical	"School Bus", "Private"
WklyStudyHours	The number of hours the student spends studying per week.	Categorical	"< 5 hours", "5 - 10 hours", "> 10 hours"
MathScore	The student's score in mathematics.	Numeric	0 to 100
ReadingScore	The student's score in reading.	Numeric	0 to 100
WritingScore	The student's score in writing.	Numeric	0 to 100
Average Score	The average of MathScore, ReadingScore, and WritingScore, added during preprocessing for analysis purposes.	Numeric	Continuous (0 to 100)

Key Dataset Characteristics:

1. **Missing Values**:

4- Check missing "NA" :

In this step, we identify and count missing values (represented as NA or NaN) in the dataset. This helps in understanding how much and which parts of the dataset are incomplete, allowing for informed decisions on how to handle the missing data.

Missing Values: Checking for missing values is a crucial part of data cleaning. It provides insight into how much data needs to be handled (through methods such as imputation or removal).

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

# Create a table showing missing values for each variable
print("\nMissing Values:")
missing_values = df.isnull().sum()
missing_table = pd.DataFrame({'Variable': missing_values.index, 'Missing Values': missing_values.values})
display(missing_table)
```

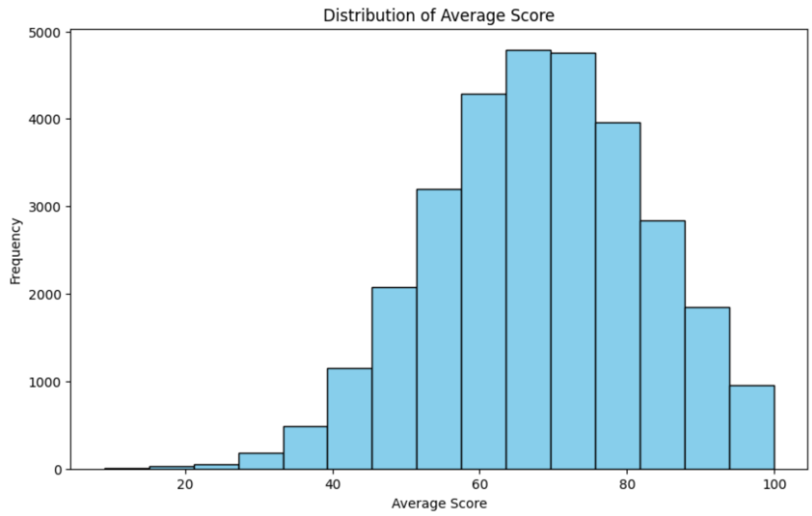
```
Total number of missing values in the dataset: 13433


Missing Values:
```

	Variable	Missing Values
0	Unnamed: 0	0
1	Gender	0
2	EthnicGroup	1771
3	ParentEduc	1783
4	LunchType	0
5	TestPrep	1779
6	ParentMaritalStatus	1156
7	PracticeSport	611
8	IsFirstChild	881
9	NrSiblings	1527
10	TransportMeans	3044
11	WklyStudyHours	0
12	Average Score	0
13	IsFirstChildNumeric	881

2. **Distributions**:

```
df = df.dropna(subset=['MathScore', 'ReadingScore','WritingScore'])
df['Average Score'] = df[['MathScore', 'ReadingScore', 'WritingScore']].mean(axis=1)


df.drop(['MathScore', 'ReadingScore', 'WritingScore'], axis=1, inplace=True)

mean = df['Average Score'].mean()
median = df['Average Score'].median()
std_dev = df['Average Score'].std()

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
plt.figure(figsize=(10, 6))
plt.hist(df['Average Score'], bins=15, color='skyblue', edgecolor='black')
plt.title("Distribution of Average Score")
plt.xlabel("Average Score")
plt.ylabel("Frequency")
plt.show()

df.to_csv('Original_dataset.csv', index=False)
```

```
Mean: 68.11818587295889
Median: 68.33333333333333
Standard Deviation: 14.454326705496916
```



Based on the calculated values, the "Average Score" appears to be relatively balanced. The mean score is 68.12, which is very close to the median of 68.33, indicating a central distribution of scores around this value. Additionally, the standard deviation of 14.45 suggests moderate variance among the scores, meaning that scores are not widely spread from the mean. Overall, the "Average Score" can be considered reasonably balanced, with scores distributed around a stable central value without significant dispersion.

Statistical Summary for Numeric Attributes:

1. **Five Number& Variance Summary**:

1. Statistical summary of the numerical data

The statistical summary provides an overview of the numerical data in the dataset. It includes important metrics such as count, mean, standard deviation, minimum, and maximum values, as well as the quartile distribution (25%, 50%, 75%). This summary helps understand the central tendency and spread of the data.

```
#  Statistical Summary
print("\nStatistical Summary of the DataFrame:")
display(df.describe())
```

Statistical Summary of the DataFrame:

	Unnamed: 0	NrSiblings	Average Score
count	30641.000000	29069.000000	30641.000000
mean	499.556607	2.145894	68.118186
std	288.747894	1.458242	14.454327
min	0.000000	0.000000	9.000000
25%	249.000000	1.000000	58.333333
50%	500.000000	2.000000	68.333333
75%	750.000000	3.000000	78.666667
max	999.000000	7.000000	100.000000

2- Correlation Coefficient & Variance & Mean & Median:

These are key descriptive statistics for understanding the relationships between variables and the overall distribution of your data.

Correlation Coefficient: This metric shows the relationship between numerical variables. A value close to 1 or -1 indicates a strong correlation, while a value close to 0 indicates no correlation.

Mean: The average value of the numerical columns. It helps to understand the central tendency of the data.

Variance: A measure of how much the data is spread out from the mean. High variance indicates more variability, while low variance indicates that data points are closer to the mean.

Median: The middle value when the data is sorted. The median is less affected by outliers compared to the mean.

```
# Select only numeric columns (float64 and int64)
numeric_df = df.select_dtypes(include=['float64', 'int64'])

# Correlation Coefficient Matrix
correlation = numeric_df.corr()
print("\nCorrelation Coefficient Matrix:")
display(correlation)

# Mean of numerical columns
mean_values = numeric_df.mean()
print("\nMean of the numerical columns:")
display(mean_values)

# Variance of numerical columns
variance = numeric_df.var()
print("\nVariance of the numerical columns:")
display(variance)

# Median of numerical columns
median_values = numeric_df.median()
print("\nMedian of the numerical columns:")
display(median_values)
```

Correlation Coefficient Matrix:

	Unnamed: 0	NrSiblings	Average Score
Unnamed: 0	1.000000	-0.000096	-0.000906
NrSiblings	-0.000096	1.000000	-0.001631
Average Score	-0.000906	-0.001631	1.000000

Mean of the numerical columns:

	0
Unnamed: 0	499.556607
NrSiblings	2.145894
Average Score	68.118186

dtype: float64

Variance of the numerical columns:

	0
Unnamed: 0	83375.346543
NrSiblings	2.126471
Average Score	208.927561

dtype: float64

Median of the numerical columns:

0

	0
Unnamed: 0	83375.346543
NrSiblings	2.126471
Average Score	208.927561

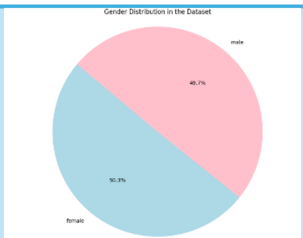
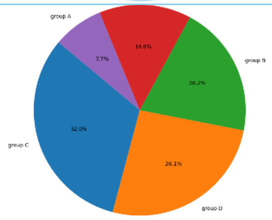
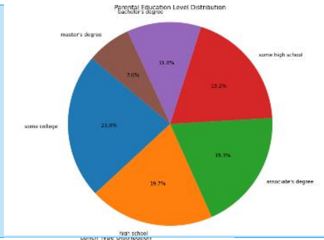
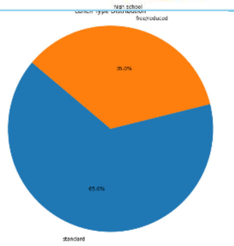
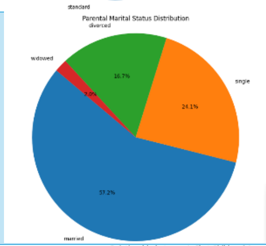
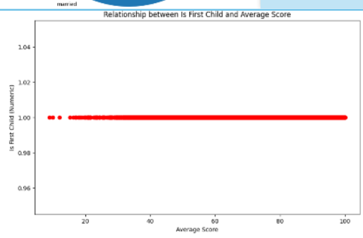
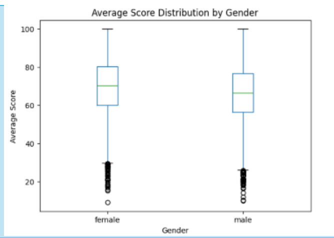
dtype: float64

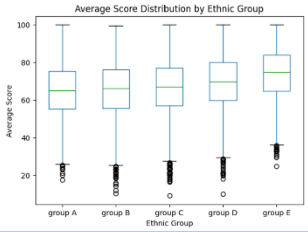
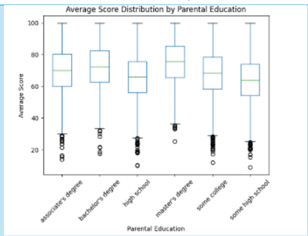
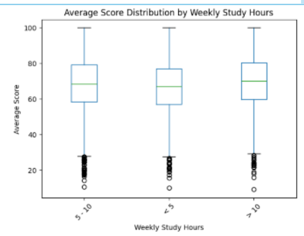
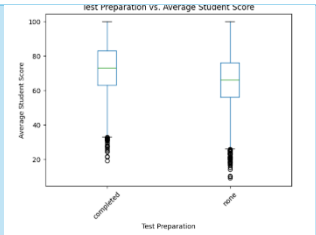
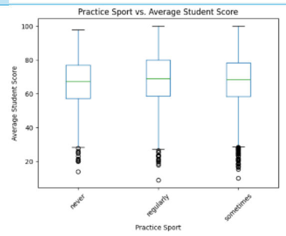
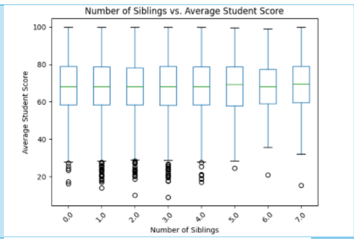
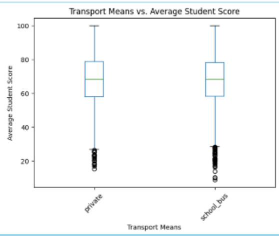
Median of the numerical columns:

	0
Unnamed: 0	500.000000
NrSiblings	2.000000
Average Score	68.333333

dtype: float64

Graphical Presentation:

Name of Graph	Picture of Graph	Description
Pie Chart		**The distribution of "Gender" shows an almost equal ratio of male and female students.**
Pie Chart		**A pie chart depicting the distribution of students among various ethnic groups.**
Pie Chart		**Highlights the distribution of students by parental education levels.**
Pie Chart		**Displays the ratio of students with standard versus free/reduced lunch types.**
Pie Chart		**Visualizes the marital status distribution of students' parents.**
Scatter Plot		**Indicates no clear correlation between being a first child and academic performance.**
Boxplot		**Compares average scores between male and female students, with similar performance ranges.**

<table>
<tr><td>Boxplot</td><td>
</td><td>Highlights score distributions across different ethnic groups, showing similar variability.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Reveals higher parental education correlates with improved student performance.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Shows students with >10 hours of weekly study tend to achieve higher scores.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Students who completed test preparation have notably higher average scores.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Indicates that regular sports participation does not significantly affect academic performance.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Displays no strong correlation between the number of siblings and academic performance.</td></tr>
<tr><td>Boxplot</td><td>
</td><td>Compares scores for students using private transport versus school buses, showing minimal impact.</td></tr>
</table>

4. Data preprocessing

- **Filling the missing values:**

```
Missing Values:
```

	Variable	Missing Values
0	Unnamed: 0	0
1	Gender	0
2	EthnicGroup	1840
3	ParentEduc	1845
4	LunchType	0
5	TestPrep	1830
6	ParentMaritalStatus	1190
7	PracticeSport	631
8	IsFirstChild	904
9	NrSiblings	1572
10	TransportMeans	3134
11	WklyStudyHours	955
12	IsFirstChildNumeric	904
13	Average Score	0

The columns EthnicGroup, ParentEduc, ParentMaritalStatus, and IsFirstChild were processed by filling their missing values using their respective modes, which is the most frequent value in each column. This approach is suitable for handling missing categorical data as it maintains consistency without altering the original distribution. As a result, missing values were replaced with the most common value in each column, ensuring the dataset remains complete while preserving its original data pattern.

```
[ ]  # For categorical columns, fill missing values with mode (most frequent value).

     df['EthnicGroup'].fillna(df['EthnicGroup'].mode()[0], inplace=True)
     df['ParentEduc'].fillna(df['ParentEduc'].mode()[0], inplace=True)
     df['ParentMaritalStatus'].fillna(df['ParentMaritalStatus'].mode()[0], inplace=True)
     df['IsFirstChild'].fillna(df['IsFirstChild'].mode()[0], inplace=True)
```

	Variable	Missing Values
0	Unnamed: 0	0
1	Gender	0
2	EthnicGroup	0
3	ParentEduc	0
4	LunchType	0
5	TestPrep	1830
6	ParentMaritalStatus	0
7	PracticeSport	631
8	IsFirstChild	0
9	NrSiblings	1572
10	TransportMeans	3134
11	WklyStudyHours	955
12	IsFirstChildNumeric	904
13	Average Score	0

- **Detect and Removing Outliers:**

```
for col in numerical_columns:
    lower_bound, upper_bound = outlier_bounds[col]
    outliers_count = df[(df[col] < lower_bound) | (df[col] > upper_bound)].shape[0]
    print(f"Number of outliers in '{col}': {outliers_count}")
```

```
Number of outliers in 'Average Score': 72
```

Justification for Not Handling Outliers:

Although outliers were detected in the dataset, their number is relatively small compared to the total sample size. This minimal proportion of outliers does not significantly impact the credibility or validity of the data. Therefore, handling outliers in this case is not deemed necessary, as their influence on the overall analysis and model performance is negligible.

Data Transmission:

- **Encoding:**

```
       Unnamed: 0  Gender EthnicGroup          ParentEduc     LunchType  \
0               0  female     group C   bachelor's degree      standard
1               1  female     group C        some college      standard
2               2  female     group B     master's degree      standard
3               3    male     group A  associate's degree  free/reduced
4               4    male     group C        some college      standard
...           ...     ...         ...                 ...           ...
30636         816  female     group D         high school      standard
30637         890    male     group E         high school      standard
30638         911  female     group C         high school  free/reduced
30639         934  female     group D  associate's degree      standard
30640         960    male     group B        some college      standard

        TestPrep ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings  \
0           none             married     regularly          yes         3.0
1            NaN             married     sometimes          yes         0.0
2           none              single     sometimes          yes         4.0
3           none             married         never           no         1.0
4           none             married     sometimes          yes         0.0
...          ...                 ...           ...          ...         ...
30636       none              single     sometimes           no         2.0
30637       none              single     regularly           no         1.0
30638  completed             married     sometimes           no         1.0
30639  completed             married     regularly           no         3.0
30640       none             married         never           no         1.0

      TransportMeans WklyStudyHours  IsFirstChildNumeric  Average Score
0         school_bus            < 5                  1.0      72.000000
1                NaN         5 - 10                  1.0      82.333333
2         school_bus            < 5                  1.0      90.333333
3                NaN         5 - 10                  0.0      47.666667
4         school_bus         5 - 10                  1.0      76.333333
...              ...            ...                  ...            ...
30636     school_bus         5 - 10                  0.0      61.666667
30637        private         5 - 10                  0.0      54.000000
30638        private         5 - 10                  0.0      66.000000
30639     school_bus         5 - 10                  0.0      88.333333
30640     school_bus         5 - 10                  0.0      60.666667

[30641 rows x 14 columns]
```

Label Encoding was applied to transform categorical columns ('Gender', 'EthnicGroup', 'ParentEduc', 'TestPrep', 'PracticeSport', 'IsFirstChild') into numerical format, as machine learning models require numerical data. This method assigns a unique integer to each category, making the data suitable for analysis. Label Encoding is particularly effective for nominal data with no intrinsic order, providing a simple and efficient conversion. By applying this preprocessing step, the dataset is now ready for machine learning models, ensuring accurate model training and analysis.

```
[35] from sklearn.preprocessing import LabelEncoder
     import pandas as pd
     from scipy import stats

     # Load the dataset

     df = pd.read_csv('Original_dataset.csv')
     LE = LabelEncoder()

     columns_encode= ['Gender','EthnicGroup','ParentEduc','TestPrep','PracticeSport','IsFirstChild']

     for column in columns_encode:
       df[column] = LE.fit_transform(df[column])

     df.to_csv('Processed_dataset.csv' , index = False)
     print("DataFrame after Encoding")
     print(df)
```

```
DataFrame after Encoding
       Unnamed: 0  Gender  EthnicGroup  ParentEduc     LunchType  TestPrep  \
0               0       0            5           1      standard         1
1               1       0            2           4      standard         2
2               2       0            1           3      standard         1
3               3       1            0           0  free/reduced         1
4               4       1            2           4      standard         1
...           ...     ...          ...         ...           ...       ...
30636         816       0            3           2      standard         1
30637         890       1            4           2      standard         1
30638         911       0            5           2  free/reduced         0
30639         934       0            3           0      standard         0
30640         960       1            1           4      standard         1

      ParentMaritalStatus  PracticeSport  IsFirstChild  NrSiblings  \
0                 married              1             1         3.0
1                 married              2             1         0.0
2                  single              2             1         4.0
3                 married              0             0         1.0
4                 married              2             1         0.0
...                   ...            ...           ...         ...
30636              single              2             0         2.0
30637              single              1             0         1.0
30638             married              2             0         1.0
30639             married              1             0         3.0
30640             married              0             0         1.0

      TransportMeans WklyStudyHours  IsFirstChildNumeric  Average Score
0         school_bus            < 5                  1.0      72.000000
1                NaN         5 - 10                  1.0      82.333333
2         school_bus            < 5                  1.0      90.333333
3                NaN         5 - 10                  0.0      47.666667
4         school_bus         5 - 10                  1.0      76.333333
...              ...            ...                  ...            ...
30636     school_bus         5 - 10                  0.0      61.666667
30637        private         5 - 10                  0.0      54.000000
30638        private         5 - 10                  0.0      66.000000
30639     school_bus         5 - 10                  0.0      88.333333
30640     school_bus         5 - 10                  0.0      60.666667

[30641 rows x 14 columns]
```

- **Normalization:**

```
       Unnamed: 0  Gender  EthnicGroup  ParentEduc     LunchType  TestPrep  \
0               0       0            5           1      standard         1
1               1       0            2           4      standard         2
2               2       0            1           3      standard         1
3               3       1            0           0  free/reduced         1
4               4       1            2           4      standard         1
...           ...     ...          ...         ...           ...       ...
30636         816       0            3           2      standard         1
30637         890       1            4           2      standard         1
30638         911       0            5           2  free/reduced         0
30639         934       0            3           0      standard         0
30640         960       1            1           4      standard         1

      ParentMaritalStatus  PracticeSport  IsFirstChild  NrSiblings  \
0                 married              1             1         3.0
1                 married              2             1         0.0
2                  single              2             1         4.0
3                 married              0             0         1.0
4                 married              2             1         0.0
...                   ...            ...           ...         ...
30636              single              2             0         2.0
30637              single              1             0         1.0
30638             married              2             0         1.0
30639             married              1             0         3.0
30640             married              0             0         1.0

      TransportMeans WklyStudyHours  IsFirstChildNumeric  Average Score
0         school_bus            < 5                  1.0      72.000000
1                NaN         5 - 10                  1.0      82.333333
2         school_bus            < 5                  1.0      90.333333
3                NaN         5 - 10                  0.0      47.666667
4         school_bus         5 - 10                  1.0      76.333333
...              ...            ...                  ...            ...
30636     school_bus         5 - 10                  0.0      61.666667
30637        private         5 - 10                  0.0      54.000000
30638        private         5 - 10                  0.0      66.000000
30639     school_bus         5 - 10                  0.0      88.333333
30640     school_bus         5 - 10                  0.0      60.666667

[30641 rows x 14 columns]
```

Data preprocessing was applied to ensure the dataset is suitable for machine learning models, specifically through **Decimal Scaling Normalization** of the Average Score column. This technique scales values by dividing them by a power of 10 based on the maximum absolute value, ensuring all values are within a similar range. Normalization was necessary to prevent features with larger magnitudes from dominating the model, improving its stability and performance, especially for distance-based or gradient-based algorithms. Decimal scaling was chosen for its simplicity, efficiency, and suitability when values are within a reasonable range. The processed dataset is now ready for further analysis or model training.

```python
import pandas as pd

df = pd.read_csv('Processed_dataset.csv')
df = pd.DataFrame(df)

# Column to Normalize
columns_to_normalize= ['Average Score']

# Decimal Scaling Normalization
for column in columns_to_normalize:
  max_abs_val = df[column].abs().max()
  df[column] = df[column] / (10 ** len(str(int(max_abs_val))))

df.to_csv('Processed_dataset.csv' , index = False)
print("DataFrame after Decimal Scaling Normalization")
print(df)
```

```
DataFrame after Decimal Scaling Normalization
       Unnamed: 0  Gender  EthnicGroup  ParentEduc    LunchType  TestPrep  \
0               0       0            5           1     standard         1
1               1       0            2           4     standard         2
2               2       0            1           3     standard         1
3               3       1            0           0  free/reduced        1
4               4       1            2           4     standard         1
...           ...     ...          ...         ...          ...       ...
30636         816       0            3           2     standard         1
30637         890       1            4           2     standard         1
30638         911       0            5           2  free/reduced        0
30639         934       0            3           0     standard         0
30640         960       1            1           4     standard         1

      ParentMaritalStatus  PracticeSport  IsFirstChild NrSiblings  \
0                 married              1             1   Moderate
1                 married              2             1        NaN
2                  single              2             1   Moderate
3                 married              0             0        Few
4                 married              2             1        NaN
...                   ...            ...           ...        ...
30636              single              2             0        Few
30637              single              1             0        Few
30638             married              2             0        Few
30639             married              1             0   Moderate
30640             married              0             0        Few

      TransportMeans WklyStudyHours  Average Score
0         school_bus            < 5       0.007200
1                NaN         5 - 10       0.008233
2         school_bus            < 5       0.009033
3                NaN         5 - 10       0.004767
4         school_bus         5 - 10       0.007633
...              ...            ...            ...
30636     school_bus         5 - 10       0.006167
30637        private         5 - 10       0.005400
30638        private         5 - 10       0.006600
30639     school_bus         5 - 10       0.008833
30640     school_bus         5 - 10       0.006067

[30641 rows x 13 columns]
```

- **Aggregation:**

The dataset is loaded from Processed_dataset.csv, which has been preprocessed (e.g., encoding, normalization). The data is grouped by WklyStudyHours and TestPrep, calculating the average score for each combination of these factors using the. mean() function. Grouping allows us to analyze how study hours and test preparation affect academic performance by comparing the average scores across these categories, helping to understand the impact of these factors on overall performance. This technique is useful when exploring how different factors influence outcomes like average scores.

```python
import pandas as pd

df = pd.read_csv('Processed_dataset.csv')
avg_score_ethnic_gender= df.groupby(['WklyStudyHours','TestPrep'])[['Average Score']].mean()

print("average score by ethnic group and gender")
print(avg_score_ethnic_gender)
```

```
average score by ethnic group and gender
                        Average Score
WklyStudyHours TestPrep
5 - 10         0             0.007292
               1             0.006593
               2             0.006905
< 5            0             0.007097
               1             0.006429
               2             0.006686
> 10           0             0.007462
               1             0.006713
               2             0.006880
```

- **Discretization:**

The dataset is loaded from Processed_dataset.csv, and the continuous NrSiblings column is discretized into categorical bins using pd.cut().The bins are defined as [-1, 0, 2, 5, 7], with values categorized as 'None', 'Few', 'Moderate', and 'Many'. Discretization helps transform continuous data into more interpretable categories, which is useful for certain analyses or machine learning models. It simplifies the data, making it easier to analyze or model by grouping continuous values into predefined segments.

```python
import pandas as pd

df = pd.read_csv('Processed_dataset.csv')

df = pd.DataFrame(df)

column_to_Discretize = 'NrSiblings'


Sibling_labels = ['None','Few','Moderate','Many']
bin_edges = [-1, 0, 2, 5, 7]


df[column_to_Discretize] = pd.cut(df[column_to_Discretize], bins=bin_edges , labels= Sibling_labels)

df.to_csv('Processed_dataset.csv' , index = False)
print("DataFrame after Discretization")
print(df)
```

```
DataFrame after Discretization
       Unnamed: 0  Gender  EthnicGroup  ParentEduc     LunchType  TestPrep  \
0               0       0            5           1      standard         1
1               1       0            2           4      standard         2
2               2       0            1           3      standard         1
3               3       1            0           0  free/reduced         1
4               4       1            2           4      standard         1
...           ...     ...          ...         ...           ...       ...
30636         816       0            3           2      standard         1
30637         890       1            4           2      standard         1
30638         911       0            5           2  free/reduced         0
30639         934       0            3           0      standard         0
30640         960       1            1           4      standard         1

      ParentMaritalStatus  PracticeSport  IsFirstChild NrSiblings  \
0                 married              1             1   Moderate
1                 married              2             1       None
2                  single              2             1   Moderate
3                 married              0             0        Few
4                 married              2             1       None
...                   ...            ...           ...        ...
30636              single              2             0        Few
30637              single              1             0        Few
30638             married              2             0        Few
30639             married              1             0   Moderate
30640             married              0             0        Few

      TransportMeans WklyStudyHours  Average Score
0         school_bus            < 5       0.072000
1                NaN         5 - 10       0.082333
2         school_bus            < 5       0.090333
3                NaN         5 - 10       0.047667
4         school_bus         5 - 10       0.076333
...              ...            ...            ...
30636     school_bus         5 - 10       0.061667
30637        private         5 - 10       0.054000
30638        private         5 - 10       0.066000
30639     school_bus         5 - 10       0.088333
30640     school_bus         5 - 10       0.060667

[30641 rows x 13 columns]
```

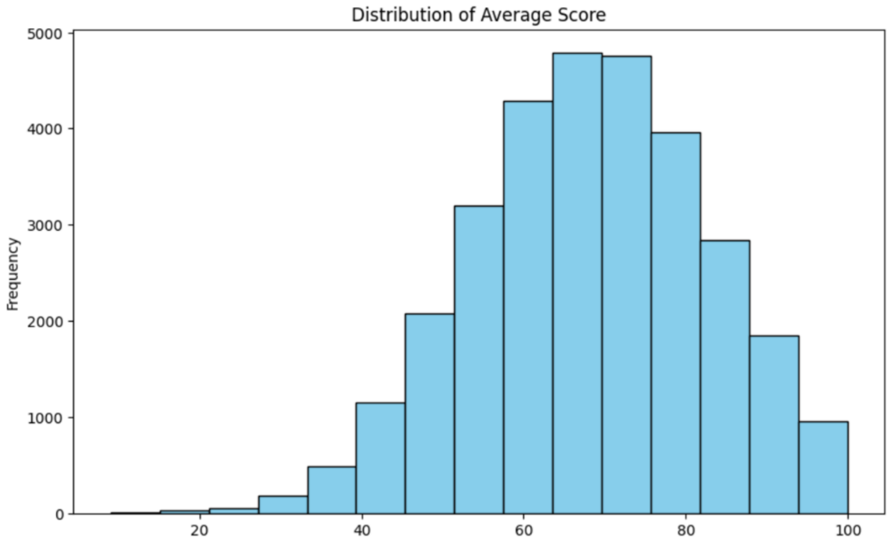
- **Balance Data:**

Before starting the Data Mining Technique, we investigated whether the data was

balanced or not:

```
mean = df['Average Score'].mean()
median = df['Average Score'].median()
std_dev = df['Average Score'].std()

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
plt.figure(figsize=(10, 6))
plt.hist(df['Average Score'], bins=15, color='skyblue', edgecolor='black')
plt.title("Distribution of Average Score")
plt.xlabel("Average Score")
plt.ylabel("Frequency")
plt.show()
```

```
Mean: 68.11818587295889
Median: 68.33333333333333
Standard Deviation: 14.454326705496916
```



Based on the calculated values, the "Average Score" appears to be relatively balanced. The mean score is 68.12, which is very close to the median of 68.33, indicating a central distribution of scores around this value. Additionally, the standard deviation of 14.45 suggests moderate variance among the scores, meaning that scores are not widely spread from the mean. Overall, the "Average Score" can be considered reasonably balanced, with scores distributed around a stable central value without significant dispersion.

5. Data Mining Technique

This project employs **classification** and **clustering** techniques to analyze students' academic performance. The primary objective is to classify students based on their performance and group them into clusters based on shared characteristics. These approaches help uncover patterns and relationships that can be used to improve educational outcomes.

Classification
For classification, the **Decision Tree** algorithm was used due to its transparency and ability to handle both categorical and numerical data. The following steps were implemented:

1. **Splitting the Data:**
   - The dataset was split into training and testing sets using different ratios:
     - **90% Training / 10% Testing**
     - **80% Training / 20% Testing**
     - **70% Training / 30% Testing**
2. **Criteria for Splitting:**
   - **Information Gain (Entropy):** Measures the reduction in uncertainty when splitting the data.
   - **Gini Index:** Evaluates the "impurity" of a dataset split to determine its quality.
3. **Evaluation Metrics:** The model performance was evaluated using:
   - **Accuracy:** Percentage of correct predictions.
   - **Sensitivity (Recall):** Proportion of actual positive cases identified correctly.
   - **Specificity:** Proportion of actual negative cases identified correctly.
   - **Precision:** Proportion of positive predictions that were correct.
   - **Error Rate:** Percentage of incorrect predictions.
4. **Visualization:**
   - **Decision Tree Diagrams** were used to display the hierarchical structure of feature splits.
   - **Confusion Matrices** illustrated the true positives, true negatives, false positives, and false negatives for each split ratio.

The results showed that the **90%-10% split** achieved the best balance between accuracy and specificity when using the **Information Gain (Entropy)** criterion.

Clustering
For clustering, the **K-means** algorithm was applied to group students into clusters based on their shared attributes. This technique provides insights into the distribution and similarities among students.

1. **Preprocessing:**
   - The **target column (Average Score)** was excluded to ensure clustering relied solely on other features.
   - Features were scaled using normalization to ensure consistent scaling across attributes.
2. **Number of Clusters (KKK):**
   - Different values of KKK were tested to identify the optimal number of clusters:
     - **Elbow Method** was used to determine K=5K = 5K=5, where adding more clusters resulted in diminishing returns in terms of compactness.
     - **Silhouette Analysis** revealed that K=9K = 9K=9 achieved the highest silhouette score, indicating the best separation between clusters.
3. **Visualization:**
   - The **Elbow Method Graph** was used to display the relationship between KKK and the within-cluster sum of squares (WSS), identifying K=5K = 5K=5 as the optimal point.
   - The **Silhouette Plot** visualized the cohesion and separation of clusters, demonstrating the quality of clustering at different values of KKK.

Python Packages and Methods

- **Data Preprocessing:**
  - pandas for data manipulation.
  - StandardScaler from sklearn.preprocessing for scaling features.
- **Classification:**
  - DecisionTreeClassifier from sklearn.tree for building and evaluating the decision tree.
  - train_test_split from sklearn.model_selection for splitting the data.
  - plot_tree and matplotlib for visualizing the decision tree.
  - confusion_matrix from sklearn.metrics for evaluating model performance.
- **Clustering:**
  - KMeans from sklearn.cluster for clustering.
  - silhouette_score from sklearn.metrics for evaluating cluster quality.
  - matplotlib for creating the Elbow and Silhouette plots.

6. Evaluation and Comparison

- **Classification:**

Classification [90% training, 10% testing] Information Gain:

Figure (1) (decision tree):

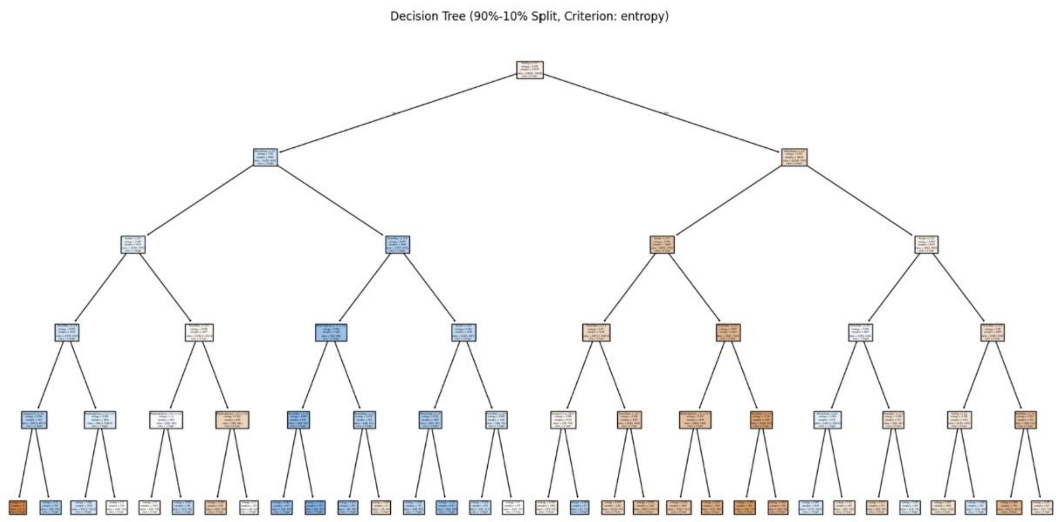
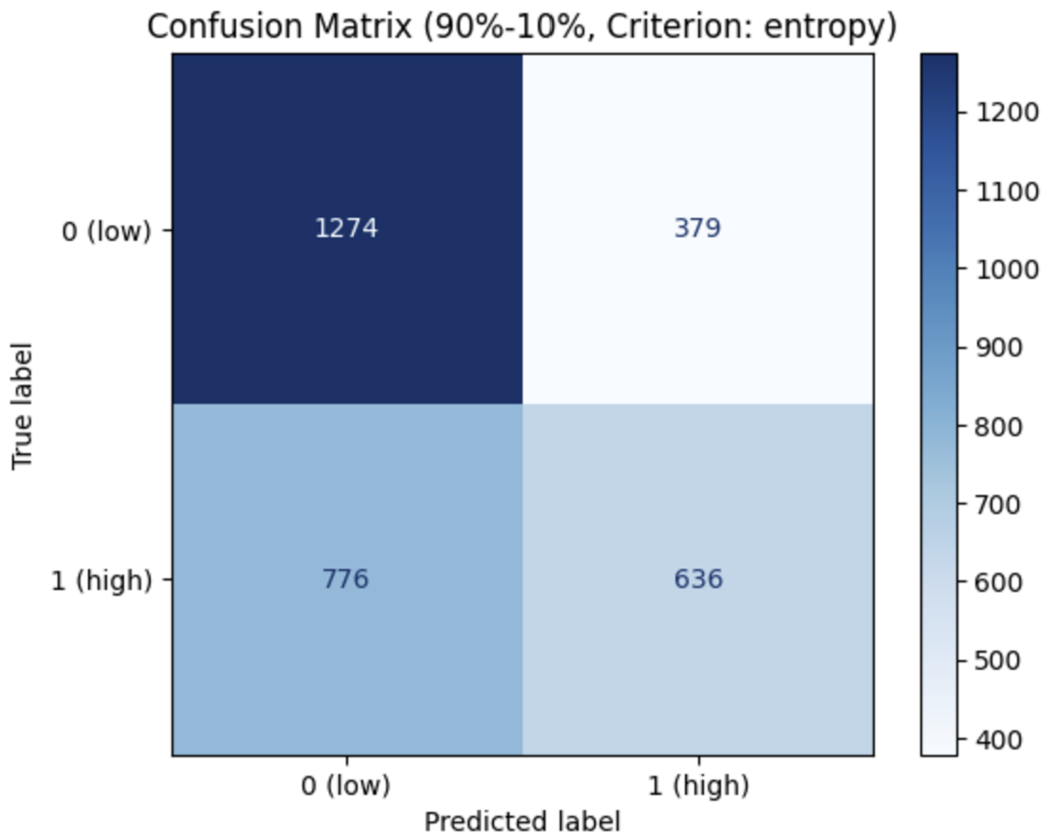


Figure (2) (confusion matrix):



Classification [80% training, 20% testing] Information Gain:

Figure (1) (decision tree):

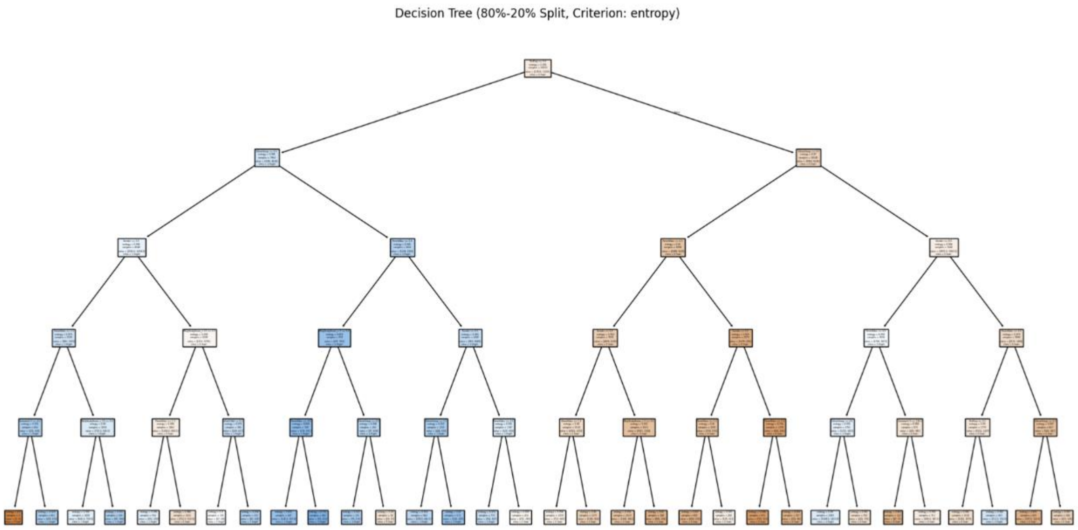
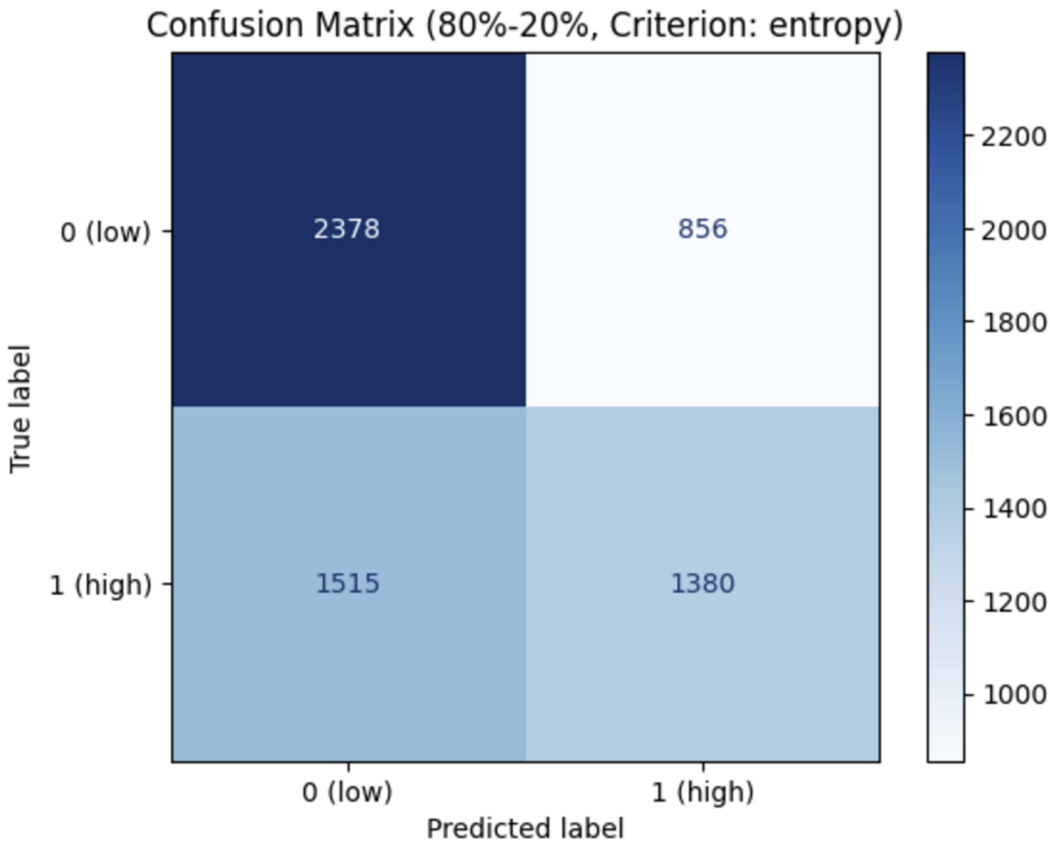


Figure (2) (confusion matrix):



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

Figure (1) (decision tree):

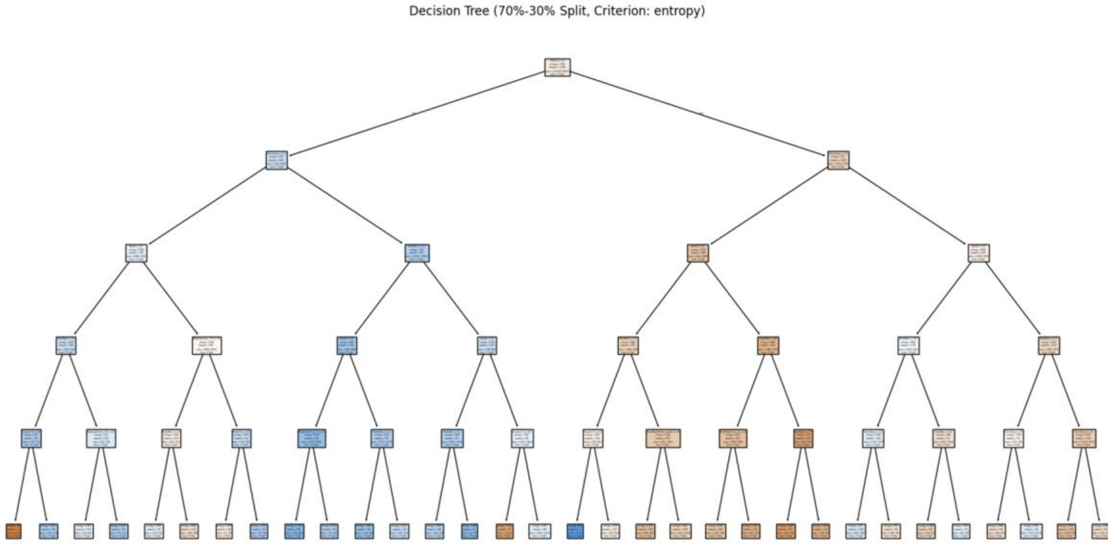
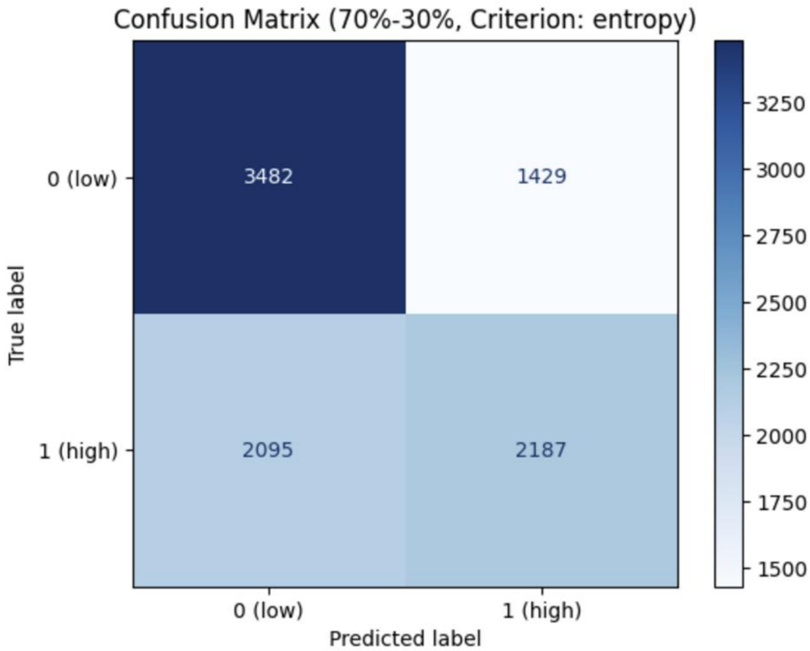


Figure (2) (confusion matrix):



Mining task	Comparison Criteria
Classification for Information Gain	We tried 3 different sizes for dataset splitting to create the decision tree: 90% Training, 10% Test data.

Accuracy	62%
precision	62%
Sensitivity	45%
Specificity	77%
Error Rate	37%

Mining task	Comparison Criteria
Classification for Information Gain	80% Training, 20% Test data

Accuracy	61%
precision	61%
Sensitivity	47%
Specificity	73%
Error Rate	38%

Mining task	Comparison Criteria
Classification for Information Gain	70% Training, 30% Test data.

Accuracy	61%
precision	60%
Sensitivity	51%
Specificity	70%
Error Rate	38%

Classification [90% training, 10% testing] Gini Index:

Figure (1) (decision tree):

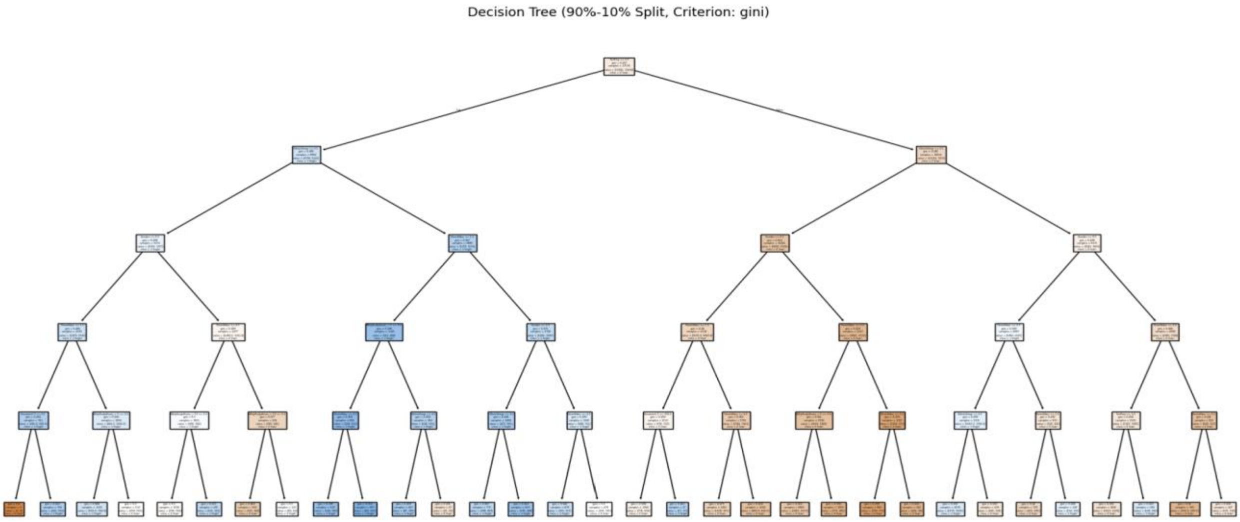
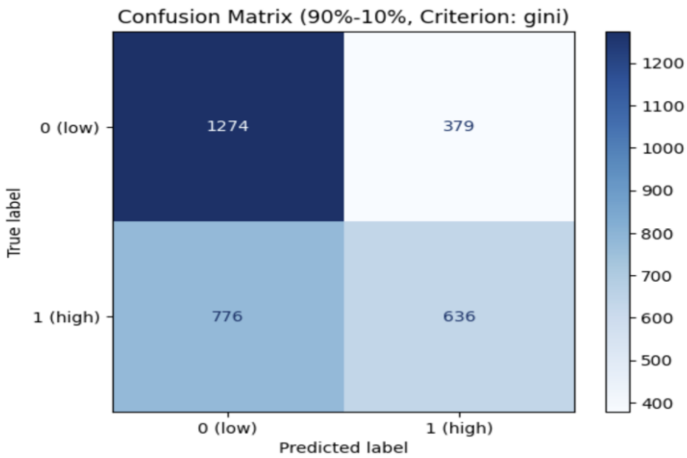


Figure (2) (confusion matrix):



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

Figure (1) (decision tree):

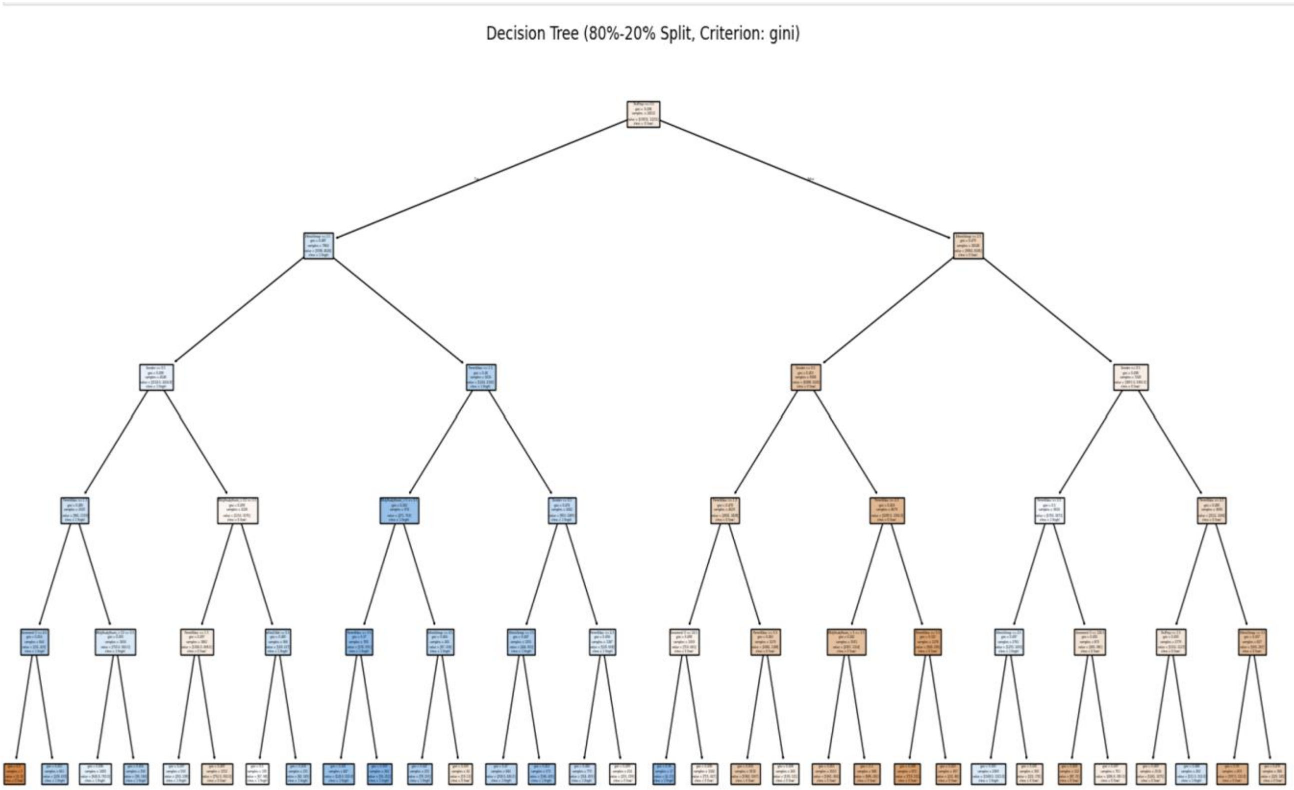
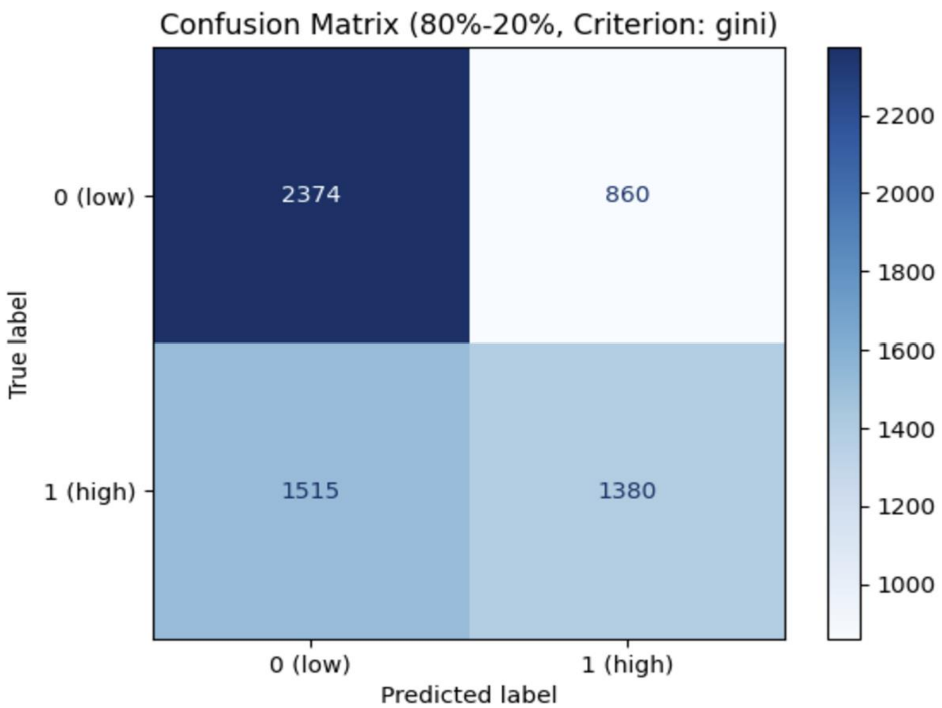


Figure (2) (confusion matrix):



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

Figure (1) (decision tree):

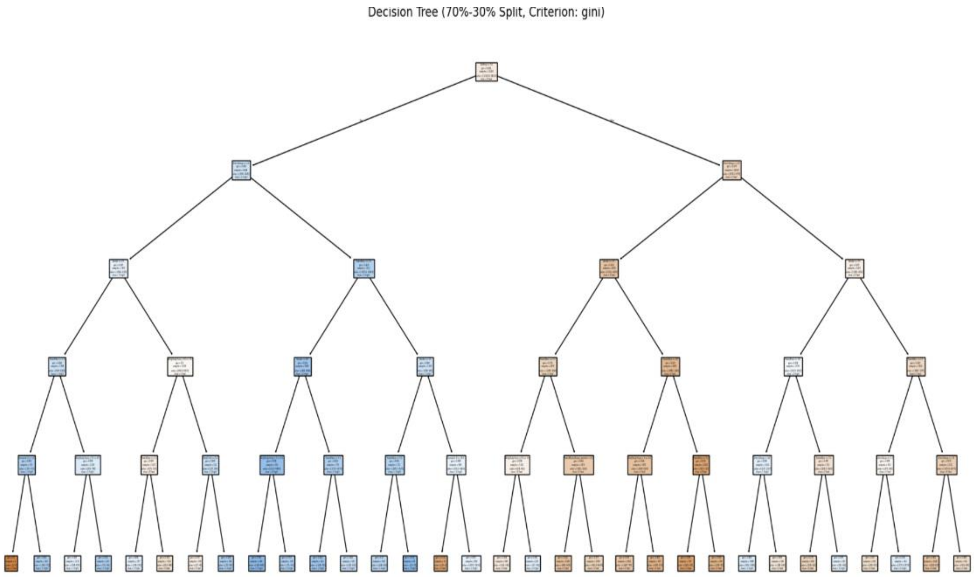
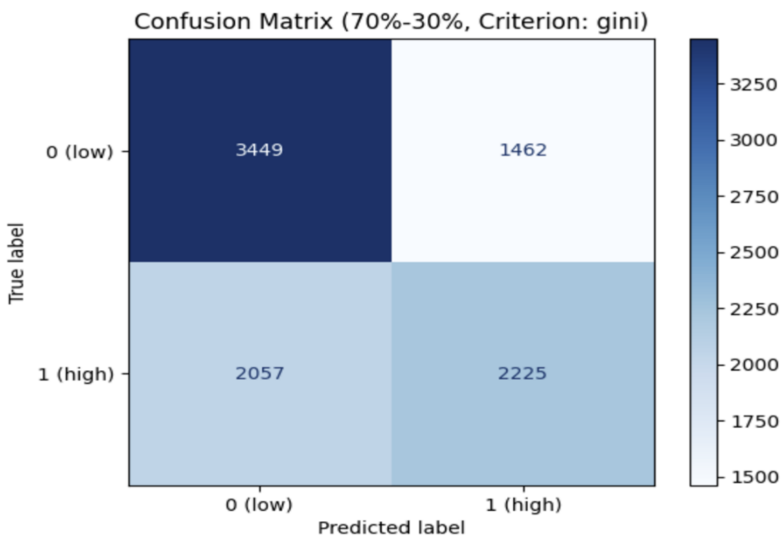


Figure (2) (confusion matrix):



Mining task	Comparison Criteria
Classification for Gini Index	We tried 3 different sizes for dataset splitting to create the decision tree: 90% Training, 10% Test data.

Accuracy	62%
precision	62%
Sensitivity	45%
Specificity	77%
Error Rate	37%

Mining task	Comparison Criteria
Classification for Gini Index	80% Training, 20% Test data

Accuracy	61%
precision	61%
Sensitivity	47%
Specificity	73%
Error Rate	38%

Mining task	Comparison Criteria
Classification for Gini Index	70% Training, 30% Test data.

Accuracy	61%
precision	60%
Sensitivity	51%
Specificity	70%
Error Rate	38%

- **The better partitioning:**

The 90%-10% splits (both Gini and Entropy) generally perform best across metrics such as accuracy, error rate, specificity, and precision, while the 70%-30% split (especially Gini) performs better on sensitivity and true positives. Depending on the importance of each metric for your use case (e.g., prioritizing accuracy over sensitivity), the 90%-10% Gini or Entropy model could be the optimal choice for a balanced performance, while the 70%-30% Gini split might be preferred if capturing positives is more critical.

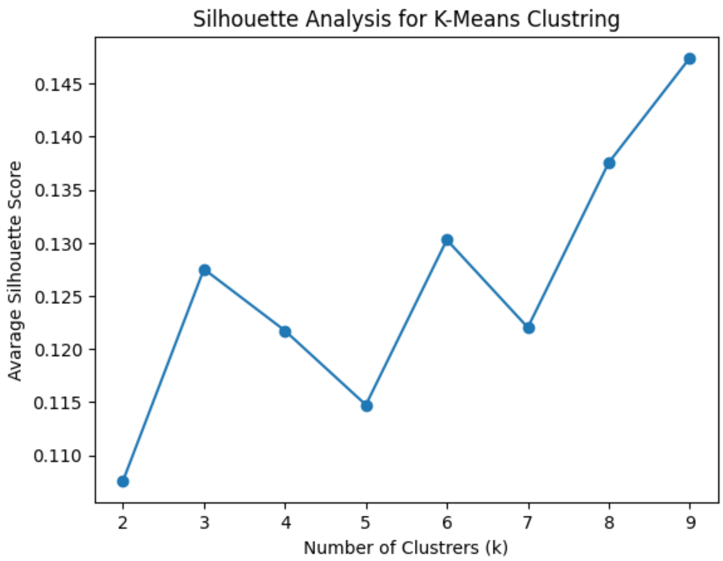
- **Clustering:**

Based on the outcomes of the validation techniques, we select three different sizes [5,8,9], and we then utilize these sizes to calculate the k-means clustering.

1. Silhouette Method

The Silhouette approach assesses clustering quality by comparing how well each point fits into its own cluster vs others. ratings range from -1 to 1, with higher ratings indicating clearly identifiable clusters.

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_samples , silhouette_score
    from sklearn.datasets import load_iris
    from sklearn.preprocessing import StandardScaler

    import pandas as pd

    k_range = range (2, 10)
    silhouette_avg_values = []

    for k in k_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans_result = kmeans.fit_predict(df_scaled)
        silhouette_avg = silhouette_score(df_scaled, kmeans_result)
        silhouette_avg_values.append(silhouette_avg)

    plt.plot(k_range, silhouette_avg_values, marker='o')
    plt.title('Silhouette Analysis for K-Means Clustring')
    plt.xlabel('Number of Clustrers (k)')
    plt.ylabel('Avarage Silhouette Score')
    plt.show()
```

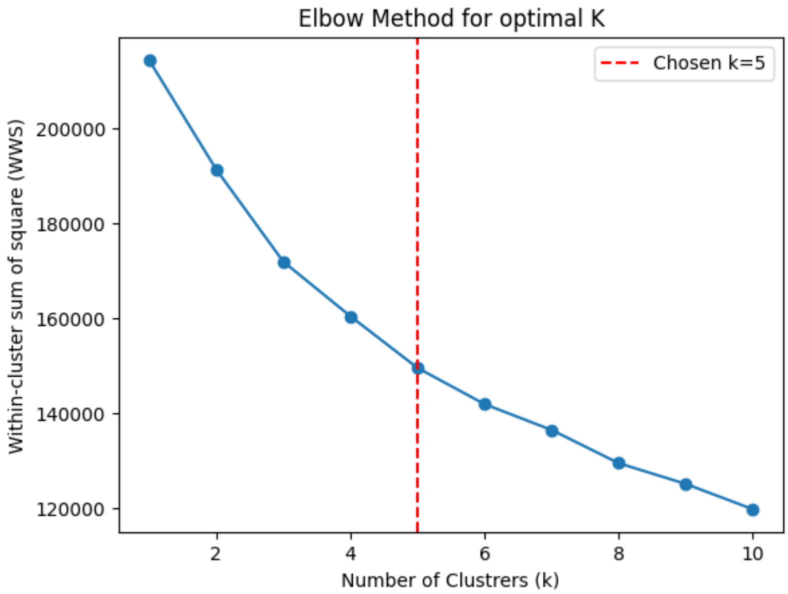


As seen above, we discovered that the ideal number of clusters (k) for maximizing the average Silhouette coefficient is 9, which will be our initial K-means option. and the second highest average Silhouette coefficient is 8.

2. Elbow method

The Elbow Method plots the within-cluster sum of squares (WSS) against (K) to find the ideal number of clusters. The ideal (K) is selected at this elbow point.

```
%pip install Kneed
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
from sklearn.pipeline import make_pipeline
from kneed import KneeLocator

wss_values = []
k_range = range (1, 11)

for k in k_range:
    kmeans = KMeans( n_clusters=k, n_init='auto', random_state=42)
    kmeans.fit(df_scaled)
    wss_values.append(kmeans.inertia_)

knee =  KneeLocator(k_range, wss_values, curve='convex',direction='decreasing')
turning_point = knee.elbow

plt.plot(k_range, wss_values, marker='o')
plt.title('Elbow Method for optimal K')
plt.xlabel('Number of Clustrers (k)')
plt.ylabel('Within-cluster sum of square (WWS)')

plt.axvline(x=turning_point, linestyle='--', color='red',label=f'Chosen k={turning_point}')
plt.legend()
plt.show()
```

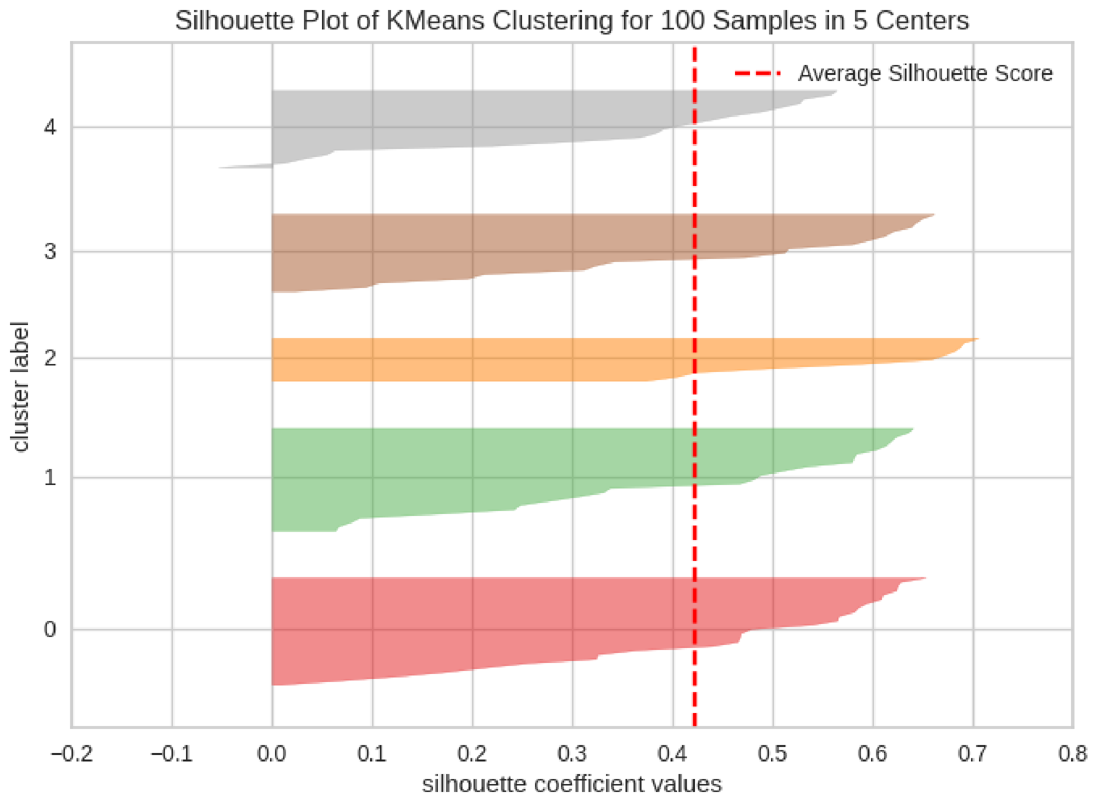


Our analysis of the elbow plot revealed a second turning point at (k = 5). We will choose (k = 5) for the third K-means clustering, as this reflects a new cluster structure. By taking into account this extra turning point, we hope to capture a varied spectrum of cluster forms while maximizing potential clustering performance.

- Trail 1 : Silhouette scores [K =5]

```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale

np.random.seed(42)

kmeans = KMeans( n_clusters=5, random_state=42,n_init='auto')
kmeans.fit(df_scaled)

print("Cluster Centers:")
print(kmeans.cluster_centers_)
print("\nCluster Labels:")
print(kmeans.labels_)
```

```python
[ ] %pip install yellowbrick
    from yellowbrick.cluster import SilhouetteVisualizer
    from sklearn.cluster import KMeans

    kmeans = KMeans( n_clusters=5,n_init='auto')
    visualizer = SilhouetteVisualizer(kmeans, color="yellowbrick")
    visualizer.fit(df_scaled)
    visualizer.show()
```



the majority of the silhouette scores are positive indicates that the samples are well-matched to their clusters and are separated from nearby clusters. This shows that the clustering algorithm successfully divided the data into discrete, well-defined groups.

```
[ ] from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_score
    import pandas as pd

    kmeans = KMeans( n_clusters=5, random_state=42,n_init='auto')
    kmeans.fit(df_scaled)
    labels = kmeans.labels_

    wss = kmeans.inertia_

    silhouette_avg = silhouette_score(df_scaled, labels)

    print("WSS:", wss)
    print("Avarage Silhouette Score:" ,silhouette_avg )
```

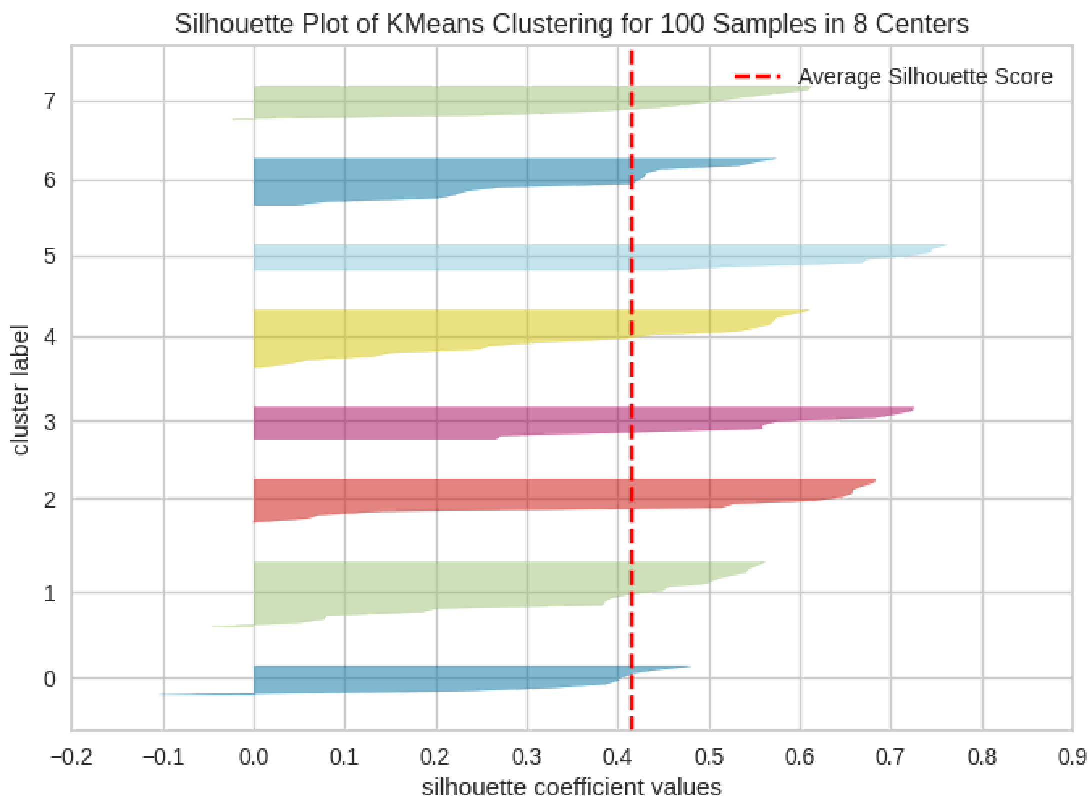
```
WSS: 35.587184050480836
Avarage Silhouette Score: 0.42217536279583234
```

The WSS value of 35.5872 represents the total variation within the clusters, with lower values indicating tighter clusters. While this value is not very low, it suggests that the clusters are somewhat separated but still may have room for improvement in terms of compactness.
The Average Silhouette Score of 0.4222 is relatively good, indicating that the clusters are fairly well differentiated. However, there is still some overlap or ambiguity between the clusters, suggesting that the clustering quality might be further improved.

- Trail 2: Silhouette scores [K =8]

```
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Assuming df_scaled is your scaled dataset
# Perform K-means clustering with K=8
kmeans = KMeans(n_clusters=8, n_init='auto', random_state=42)  # Set n_clusters=8
visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')

# Fit the visualizer to your scaled data
visualizer.fit(df_scaled)

# Show the silhouette plot
visualizer.show()
```

```
[ ] import numpy as np
    from sklearn.cluster import KMeans
    from sklearn.preprocessing import scale

    np.random.seed(42)

    kmeans = KMeans( n_clusters=8, random_state=42,n_init='auto')
    kmeans.fit(df_scaled)

    print("Cluster Centers:")
    print(kmeans.cluster_centers_)
    print("\nCluster Labels:")
    print(kmeans.labels_)
```



Most of the silhouette scores with a positive value reinforce 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.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import pandas as pd

# Assuming df_scaled is your scaled data
X = df_scaled  # Make sure df_scaled is defined before using it

# Perform k-means clustering with k=8
kmeans = KMeans(n_clusters=8, random_state=42, n_init='auto')  # Set k=8
kmeans.fit(X)  # Use X (which is df_scaled) for fitting
labels = kmeans.labels_

# Compute the WSS (Within-Cluster Sum of Squares)
wss = kmeans.inertia_

# Compute the Average Silhouette Score
silhouette_avg = silhouette_score(X, labels)

# Print the evaluation metrics
print("WSS:", wss)
print("Average Silhouette Score:", silhouette_avg)
```

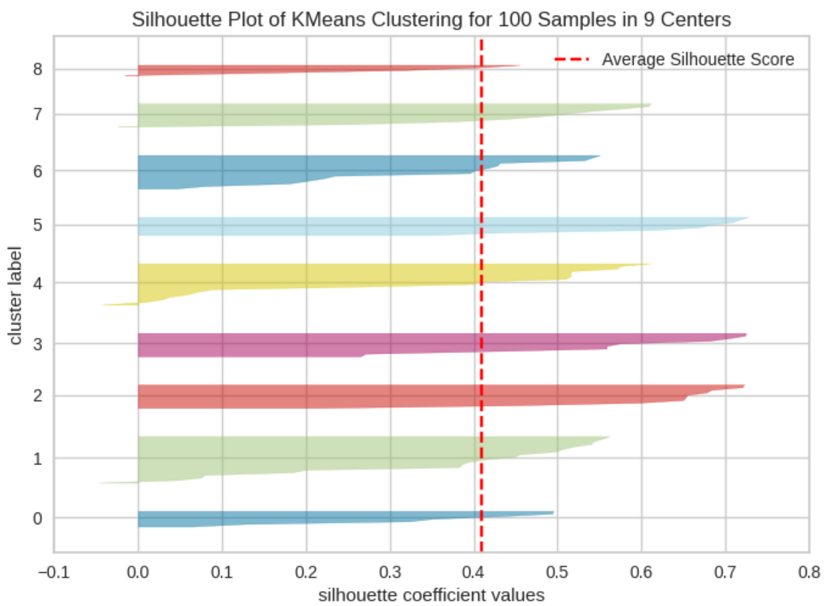
```
WSS: 20.09324711660112
Average Silhouette Score: 0.41622939498931744
```

WSS: 20.09324711660112 - The relatively low WSS value suggests that the clusters are reasonably well-separated and compact, though not as optimal as some lower WSS values might indicate. The clusters are still relatively tight but could potentially benefit from some refinement.
Average Silhouette Score: 0.41622939498931744 - The moderately high score indicates that there is still some degree of overlap or ambiguity in the cluster assignments, but the clusters are generally well-defined and distinct. While not perfect, the clusters appear to exhibit acceptable separation overall.

- Trail 3: Silhouette scores [K =9]

```
[ ]  import numpy as np
     from sklearn.cluster import KMeans
     from sklearn.preprocessing import scale

     np.random.seed(42)

     kmeans = KMeans( n_clusters=9, random_state=42,n_init='auto')
     kmeans.fit(df_scaled)

     print("Cluster Centers:")
     print(kmeans.cluster_centers_)
     print("\nCluster Labels:")
     print(kmeans.labels_)
```

```
[ ]  from yellowbrick.cluster import SilhouetteVisualizer
     from sklearn.cluster import KMeans
     import matplotlib.pyplot as plt

     # Assuming df_scaled is your scaled dataset
     # Perform K-means clustering with K=9
     kmeans = KMeans(n_clusters=9, n_init='auto', random_state=42)  # Set n_clusters=9
     visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')

     # Fit the visualizer to your scaled data
     visualizer.fit(df_scaled)

     # Show the silhouette plot
     visualizer.show()
```



the fact that most of the silhouette scores have positive values is indeed a positive indicator. Positive silhouette scores suggest that the samples are well-matched to their clusters and are relatively distant from neighboring clusters. This reinforces the notion that the clustering solution has successfully separated the data points into distinct and well-defined clusters.

```
[ ] from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_score
    import pandas as pd

    # Assuming df_scaled is your scaled data
    X = df_scaled  # Make sure df_scaled is defined with your scaled dataset

    # Perform k-means clustering with k=9
    kmeans = KMeans(n_clusters=9, random_state=42, n_init='auto')  # Set k=9
    kmeans.fit(X)  # Use X (which is df_scaled) for fitting
    labels = kmeans.labels_

    # Compute the WSS (Within-Cluster Sum of Squares)
    wss = kmeans.inertia_

    # Compute the Average Silhouette Score
    silhouette_avg = silhouette_score(X, labels)

    # Print the evaluation metrics
    print("WSS:", wss)
    print("Average Silhouette Score:", silhouette_avg)

WSS: 18.096000876015545
Average Silhouette Score: 0.4086975323167754
```

WSS: 18.096000876015545 - The WSS value indicates that the cluster separation and compactness are reasonable, though not as strong as expected for ideal clustering.
Average Silhouette Score: 0.4086975323167754 - The relatively high score suggests that there is some overlap or ambiguity in the cluster assignments, but overall the clusters are reasonably well-defined and distinc.

<table>
<tr><th>Mining task</th><th>Comparison Criteria</th></tr>
<tr><td>Clustering</td><td>We used 3 sizes of K,
K=5, K=8, K=9
<table>
<tr><th></th><th>K=5
(best)</th><th>K=8</th><th>K=9</th></tr>
<tr><td>Average Silhouette width</td><td>0.42218</td><td>0.41623</td><td>0.40870</td></tr>
<tr><td>Total within-cluster sum of square</td><td>35.587184050480836</td><td>20.09324711660112</td><td>18.096000876015545</td></tr>
</table>
</td></tr>
</table>

Conclusion: The K=5 model demonstrates superior clustering performance compared to the K=8 and K=9 models. It effectively separates the data into distinct and well-defined clusters, making it the most suitable choice for this particular dataset.

7. Findings

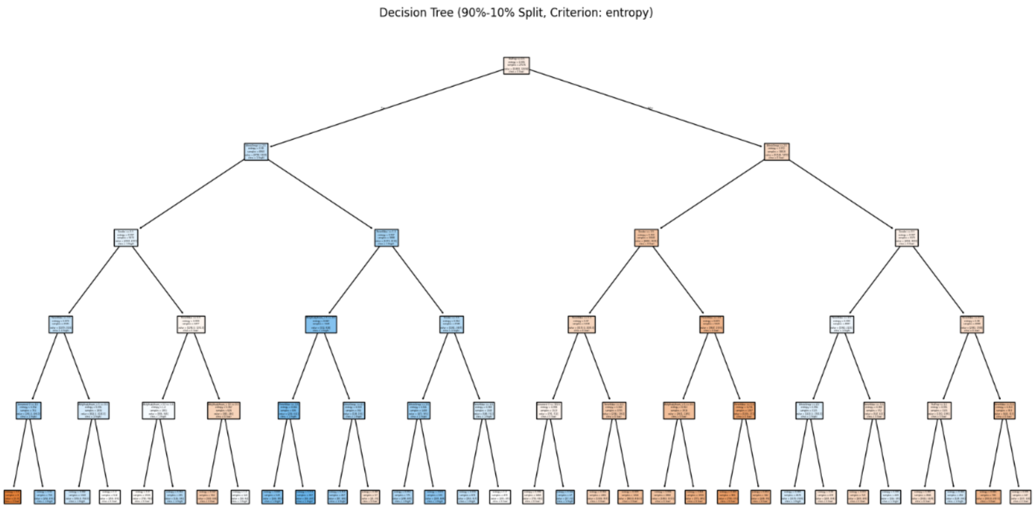
The goal of this analysis was to classify students based on their performance and to identify clusters of students with similar characteristics. **Classification** was carried out using the **Decision Tree algorithm**, tested with both **Information Gain (Entropy)** and **Gini Index** criteria. For **clustering**, the **K-means algorithm** was used to group students based on shared features, with different values of KKK evaluated to determine the optimal number of clusters. The results are discussed in detail below.

Classification Results

The **Decision Tree algorithm** was applied with varying criteria and training/testing splits. The performance of the models was evaluated using confusion matrices and metrics such as accuracy, sensitivity, specificity, and precision. The following observations were made:
Using Information Gain (Entropy):

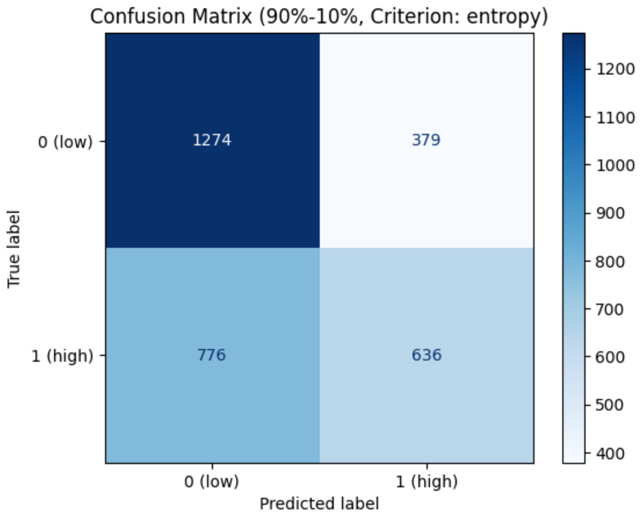
1. **Model Performance**:
   - The **90%-10% training/testing split** achieved a good balance between specificity and precision.
   - The confusion matrix revealed:
     - **True Positives (TP):** 636 (high-performing students correctly classified).
     - **True Negatives (TN):** 1274 (low-performing students correctly identified).
     - **False Positives (FP):** 379 (low performers misclassified as high performers).
     - **False Negatives (FN):** 776 (high performers misclassified as low performers).
2. **Feature Importance**:
   - The **Decision Tree diagram** highlighted **Test Preparation** as the most decisive feature, followed by **Parental Education** and **Weekly Study Hours**.

- **Decision Tree Diagram (Entropy)**:



"This Decision Tree illustrates the hierarchical structure of decision-making based on the Entropy criterion. Key features such as **Test Preparation** and **Parental Education** are crucial for predicting student performance."

- **Confusion Matrix (90%-10%, Entropy)**:

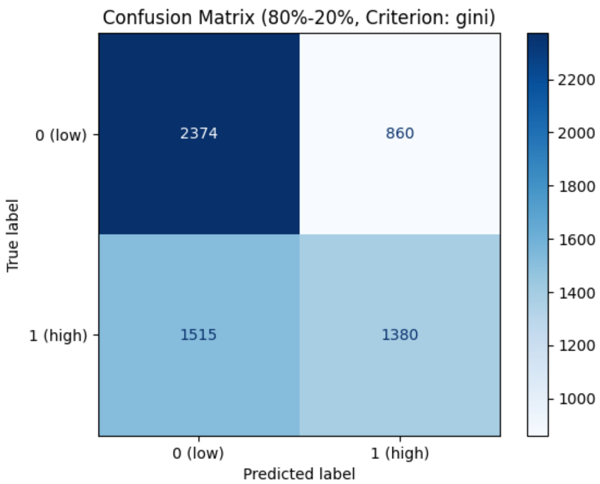


"The confusion matrix displays the classification performance for the Decision Tree using Information Gain, highlighting true/false positives and negatives."

Using Gini Index:

1. **Model Performance**:
   - The **80%-20% training/testing split** showed slightly improved sensitivity, making it more effective for detecting high-performing students.
   - The confusion matrix for this split revealed:
     - **True Positives (TP):** A higher count compared to the Entropy criterion.
     - **False Negatives (FN):** Reduced misclassification of high performers.
2. **Feature Importance**:
   - Similar to Entropy, **Test Preparation** and **Parental Education** were significant, but the tree structure was more compact, leading to faster decision-making.

- **Confusion Matrix (80%-20%, Gini)**:



"This confusion matrix showcases the results for the Decision Tree model using the Gini Index, with improved sensitivity for high-performing students."

The Best Model Between Information Gain and Gini Index

After analyzing the performance of both Information Gain (Entropy) and Gini Index, the Decision Tree using Information Gain with a 90%-10% split was selected as the best model for this dataset. The reasons for this choice are:

1. Balanced Performance:
   - While both criteria achieved similar accuracy, Information Gain provided a better balance between specificity (correctly identifying low performers) and precision (correctly predicting high performers).
   - This makes it more suitable for datasets where accurately identifying key groups (e.g., high performers) is critical.
2. Feature Insights:
   - The model using Information Gain highlighted Test Preparation, Parental Education, and Weekly Study Hours as the most influential features, which aligns with practical expectations and provides actionable insights.
3. Model Interpretability:
   - The splits generated by Information Gain were clearer and more intuitive compared to Gini Index, making it easier to explain the decision-making process to stakeholders.

Conclusion: The Decision Tree with Information Gain was selected as the best model due to its superior balance of performance metrics and its ability to provide interpretable results. This model is particularly useful for identifying at-risk students and designing targeted interventions to improve academic outcomes.

Best Model for Classification:
After evaluating both criteria, the **Decision Tree using Information Gain (Entropy)** with a **90%-10% split** was chosen as the best model. Its ability to balance specificity and precision makes it suitable for addressing the problem.

Clustering Results

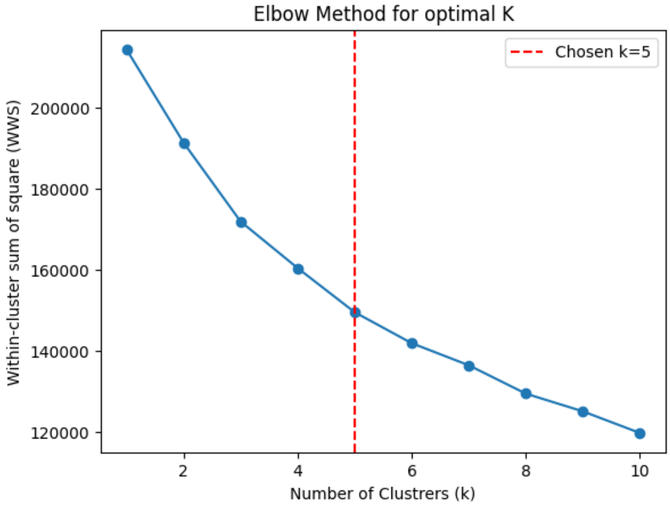
For Clustering, we used K-means algorithm with 3 different K to find the optimal number of clusters, we calculated the average silhouette width for each K, and we concluded the following results:

Model	WSS	Average Silhouette Score
K=8	20.09324711660112	0.41623
K=9	18.096000876015545	0.40870
K=5	35.587184050480836	0.42218

The **K-means algorithm** was applied to group students into clusters based on their shared characteristics. Several KKK values were tested to determine the optimal number of clusters.
Optimal Clustering:

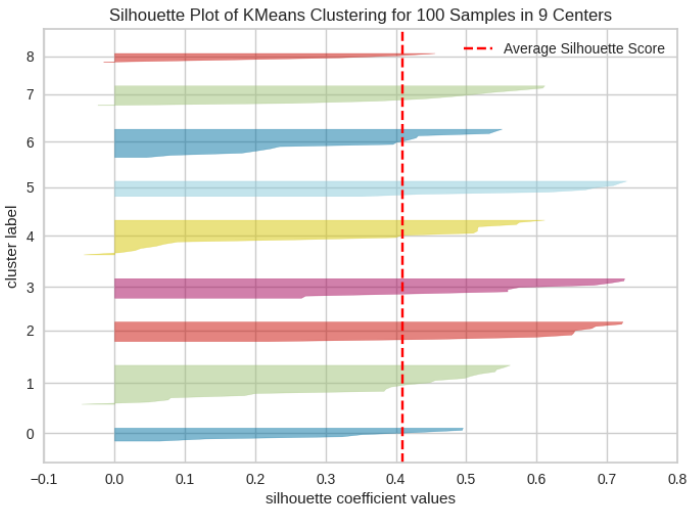
1. **Elbow Method**:
   - The **Elbow Method** identified K=5K = 5K=5 as the optimal number of clusters, balancing compactness and simplicity.
   - This ensures well-defined clusters with minimal overlap.
2. **Silhouette Score**:
   - The highest **Silhouette Score** was observed at K=9K = 9K=9, indicating the clusters are well-separated and cohesive.

- **Elbow Method Graph**:



"The Elbow Method graph identifies K=5K = 5K=5 as the optimal number of clusters, where adding more clusters no longer significantly reduces the Within-Cluster Sum of Squares (WSS)."

- **Silhouette Plot**:

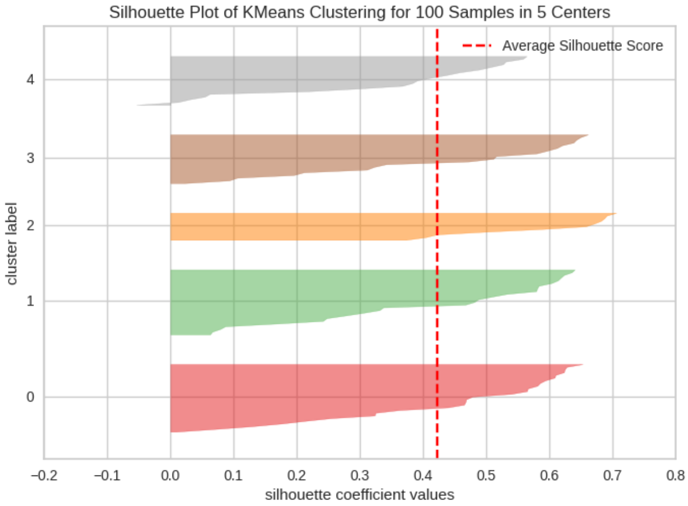


"The Silhouette Plot evaluates clustering quality, with K=9K = 9K=9 achieving the highest average score, indicating well-separated clusters."

Cluster Insights:

1. **Cluster Characteristics**:
    - **High-Performing Clusters**:
        - Students with consistent test preparation and high weekly study hours.
    - **Low-Performing Clusters**:
        - Students with limited parental education and lower test preparation efforts.
2. **Patterns Revealed**:
    - The clustering analysis revealed distinct groups of students based on socio-economic and study-related factors, enabling targeted interventions.

- **Cluster Distribution Scatter Plot (K = 5)**:



"This scatter plot visualizes the distribution of students across clusters for K=5K = 5K=5, highlighting shared characteristics such as **Test Preparation** and **Parental Education**."

Problem Solutions

Based on the findings, the following solutions are proposed:

1. **Targeted Interventions**:
    - Focus on students in low-performing clusters by providing additional support for test preparation and improving study habits.
2. **Policy Recommendations**:
    - Encourage schools to provide resources for parental engagement, as **Parental Education** plays a significant role in student outcomes.
3. **Use of Decision Trees**:
    - The Decision Tree model offers a transparent framework for identifying students needing assistance and allocating resources effectively.

Conclusion

The analysis demonstrates the effectiveness of combining **classification** and **clustering** techniques to address the problem. The **Decision Tree using Information Gain (Entropy)** and **K-means clustering with K=5K = 5K=5** is identified as the best models for solving the problem under study. These methods not only predict student performance but also uncover actionable patterns for improving academic outcomes.

8. References

[1] D. Geb, "Students Exam Scores," *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/desalegngeb/students-exam-scores. [Accessed: Nov. 30, 2024].