University of Barishal

Lab Report

Course Title: Machine Learning Lab; Course Code: CSE-4225
[Machine Learning Classification and Clustering Techniques]
(4th Year 2nd Semester)

Submitted To:

Dr. Tania Islam
Assistant Professor
Department of Computer Science and Engineering

Proposed By:

Obaydul Hasan Nayeem
Class Roll: 19CSE014
Session: 2018-19

Submission Date:

2 Oct 2024

Department of Computer Science and Engineering

Faculty of Science and Engineering, University of Barishal

Table of Contents

1. Introduction **3**
2. Dataset Description **3**
2.1 Overview 3
2.2 Column Descriptions 3
3. Algorithms Overview **4**
3.1 Naive Bayes Algorithm 4
3.2 K-Nearest Neighbors (KNN) 5
3.3 Decision Tree Algorithm 7
3.4 Support Vector Machine (SVM) 8
3.5 K-Means Clustering 10
3.6 Confusion Matrix 11
Example: 15
4. Performance Analysis and Result Interpretation **15**
4.1 Summary of Results 15
4.2 Explanation of Results 15
4.3 Analysis of Performance 15
5. Comparison of All Algorithms **16**
Explanation: 16
6. Conclusion **17**

1. Introduction

Machine learning has become an essential tool in data analysis and prediction across various domains, including education, healthcare, finance, and more. This lab report explores the application of several machine learning algorithms to classify student performance based on multiple features, such as demographics and previous academic records. We aim to understand how different algorithms perform in this context and identify the best approaches for predicting student grades.

2. Dataset Description

2.1 Overview

The dataset - 'Higher Education Students Performance Evaluation' used in this report consists of various features that influence student performance, such as:

- **Demographic Information**: Age, gender, etc.
- **Previous Academic Records**: Grades from previous semesters.
- **Behavioral Factors**: Attendance, participation in extracurricular activities, etc.

The target variable is the `GRADE`, which represents the final grade achieved by students. The `STUDENTID` column serves as a unique identifier and is excluded from analysis, as it does not contribute to the prediction. The dataset was preprocessed by encoding categorical features into numerical format and splitting it into training and testing sets (80% training, 20% testing).

- **Total Entries**: 30
- **Total Columns**: 8
- **Data Completeness**: The dataset is complete with no missing values across all columns.

Dataset Link: Higher Education Students Performance Evaluation (kaggle.com)

Source Code:
https://colab.research.google.com/drive/1bIQIgyFDzM82lwFWerkISG7cWWF7hw5A?usp=sharing

2.2 Column Descriptions

1. STUDENTID: A unique identifier for each student (not used for analysis).
2. AGE: The age of the student.
3. GENDER: The gender of the student (e.g., Male, Female).
4. PREVIOUS_GRADES: Grades from previous academic terms (numerical value).
5. ATTENDANCE: Percentage of attendance in previous terms.
6. PARTICIPATION: Engagement in extracurricular activities (numerical value).
7. HOURS_STUDIED: Average hours studied per week (numerical value).

8. GRADE: The final grade achieved by the student (target variable).

The dataset consists of various features that influence student performance, such as demographic information, previous academic records, and behavioral factors. The `GRADE` column represents the target variable that the models will aim to predict.

3. Algorithms Overview

3.1 Naive Bayes Algorithm

Introduction: Naive Bayes is a family of probabilistic algorithms based on Bayes' theorem. It assumes independence among predictors, which simplifies the computation.

Working Procedure:

- Calculate the prior probabilities of each class from the training data.
- Calculate the likelihood of the features given to each class.
- Use Bayes' theorem to compute the posterior probability for each class.

Pros:

- Simple and fast to compute.
- Works well with large datasets.
- Effective for categorical data.

Cons:

- Assumes independence among features, which may not hold true.
- May perform poorly with correlated features.

Limitations: Not suitable for datasets where feature independence is violated.

Results Analysis: The Naive Bayes classifier achieved an accuracy of **13.79%**, indicating poor performance across most classes.

Accuracy	13.79%
Classification Report	{'0': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '1': {'precision': 0.3333333333333333, 'recall': 0.3333333333333333, 'f1-score': 0.3333333333333333, 'support': 3.0}, '2': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0,

	'support': 4.0}, '3': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 6.0}, '4': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '6': {'precision': 0.3333333333333333, 'recall': 0.5, 'f1-score': 0.4, 'support': 4.0}, '7': {'precision': 0.125, 'recall': 0.5, 'f1-score': 0.2, 'support': 2.0}, 'accuracy': 0.13793103448275862, 'macro avg': {'precision': 0.09895833333333333, 'recall': 0.16666666666666666, 'f1-score': 0.11666666666666667, 'support': 29.0}, 'weighted avg': {'precision': 0.08908045977011493, 'recall': 0.13793103448275862, 'f1-score': 0.10344827586206896, 'support': 29.0}}
Final Accuracy	13.79%
Final Precision	8.91%
Final Recall	13.79%
Final F1 Score	10.34%

3.2 K-Nearest Neighbors (KNN)

Introduction: KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on the majority class among its k-nearest neighbors.

Working Procedure:

- Calculate the distance between the test instance and all training instances.
- Identify the k nearest neighbors.
- Classify the test instance based on the majority class among these neighbors.

Pros:

- Simple and intuitive.
- No training phase, making it easy to implement.

Cons:

- Computationally expensive during the prediction phase.
- Sensitive to the scale of data and irrelevant features.

Limitations: Performance degrades with high-dimensional data.

Results Analysis: KNN yielded an accuracy of **34.48%**, which is an improvement over Naive Bayes but still suggests significant challenges in classifying the dataset.

Accuracy	34.48%
Classification Report	{'0': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '1': {'precision': 0.21428571428571427, 'recall': 1.0, 'f1-score': 0.35294117647058826, 'support': 3.0}, '2': {'precision': 0.5, 'recall': 0.5, 'f1-score': 0.5, 'support': 4.0}, '3': {'precision': 0.5, 'recall': 0.3333333333333333, 'f1-score': 0.4, 'support': 6.0}, '4': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '5': {'precision': 0.5, 'recall': 0.3333333333333333, 'f1-score': 0.4, 'support': 3.0}, '6': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '7': {'precision': 0.5, 'recall': 1.0, 'f1-score': 0.6666666666666666, 'support': 2.0}, 'accuracy': 0.3448275862068966, 'macro avg': {'precision': 0.2767857142857143, 'recall': 0.3958333333333333, 'f1-score': 0.28995098039215683, 'support': 29.0}, 'weighted avg': {'precision': 0.28078817733990147, 'recall': 0.3448275862068966, 'f1-score': 0.27559161595672754, 'support': 29.0}}
Final Accuracy	34.48%

Final Precision	28.08%
Final Recall	34.48%
Final F1 Score	27.56%

3.3 Decision Tree Algorithm

Introduction: Decision Trees are a supervised learning method used for classification and regression. They create a model based on a tree-like structure of decisions.

Working Procedure:

- Split the dataset into subsets based on feature values that maximize information gain or minimize entropy.
- Continue splitting until a stopping criterion is met (e.g., depth of the tree).

Pros:

- Easy to interpret and visualize.
- Can handle both numerical and categorical data.

Cons:

- Prone to overfitting, especially with deep trees.
- Sensitive to noisy data.

Limitations: May not generalize well to unseen data if not pruned properly.

Results Analysis: The Decision Tree algorithm achieved an accuracy of **27.59%**, indicating moderate performance but still struggles with certain classes.

Accuracy	27.59%
Classification Report	{'0': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '1': {'precision': 0.125, 'recall': 0.3333333333333333, 'f1-score':

	0.18181818181818182, 'support': 3.0}, '2': {'precision': 0.2, 'recall': 0.25, 'f1-score': 0.2222222222222222, 'support': 4.0}, '3': {'precision': 0.5, 'recall': 0.3333333333333333, 'f1-score': 0.4, 'support': 6.0}, '4': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '5': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '6': {'precision': 0.5, 'recall': 0.5, 'f1-score': 0.5, 'support': 4.0}, '7': {'precision': 0.5, 'recall': 1.0, 'f1-score': 0.6666666666666666, 'support': 2.0}, 'accuracy': 0.27586206896551724, 'macro avg': {'precision': 0.228125, 'recall': 0.3020833333333333, 'f1-score': 0.24633838383838383, 'support': 29.0}, 'weighted avg': {'precision': 0.24741379310344827, 'recall': 0.27586206896551724, 'f1-score': 0.24716126785092302, 'support': 29.0}}
Final Accuracy	27.59%
Final Precision	24.74%
Final Recall	27.59%
Final F1 Score	24.72%

3.4 Support Vector Machine (SVM)

Introduction: SVM is a supervised learning model that constructs a hyperplane or set of hyperplanes in a high-dimensional space for classification.

Working Procedure:

- Find the hyperplane that maximizes the margin between different classes.
- Utilize kernel functions to handle non-linear separability.

Pros:

- Effective in high-dimensional spaces.
- Works well with a clear margin of separation.

Cons:

- Memory-intensive and not suitable for large datasets.
- Requires careful tuning of parameters.

Limitations: Can be sensitive to noise and outliers.

Results Analysis: SVM demonstrated an accuracy of **20.69%**, reflecting its limitations in this dataset.

Accuracy	20.69%
Classification Report	{'0': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '1': {'precision': 0.13636363636363635, 'recall': 1.0, 'f1-score': 0.24, 'support': 3.0}, '2': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '3': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 6.0}, '4': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 3.0}, '5': {'precision': 1.0, 'recall': 0.3333333333333333, 'f1-score': 0.5, 'support': 3.0}, '6': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 4.0}, '7': {'precision': 0.3333333333333333, 'recall': 1.0, 'f1-score': 0.5, 'support': 2.0}, 'accuracy': 0.20689655172413793, 'macro avg': {'precision': 0.18371212121212122, 'recall': 0.29166666666666663, 'f1-score': 0.155, 'support': 29.0}, 'weighted avg': {'precision': 0.1405433646812957, 'recall': 0.20689655172413793, 'f1-score': 0.11103448275862068, 'support': 29.0}}
Final Accuracy	20.69%

Final Precision	14.05%
Final Recall	20.69%
Final F1 Score	11.10%

3.5 K-Means Clustering

Introduction: K-Means is an unsupervised clustering algorithm that partitions data into K-distinct clusters based on feature similarity.

Working Procedure:

- Randomly initialize K centroids.
- Assign data points to the nearest centroid.
- Update centroids based on the mean of the assigned points.
- Repeat until convergence.

Pros:

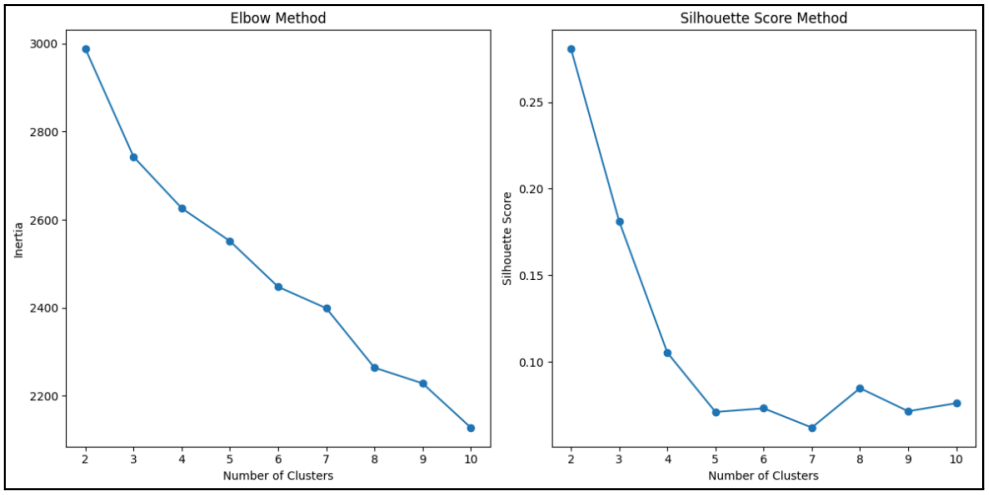
- Simple and easy to implement.
- Scalable to large datasets.

Cons:

- Requires the number of clusters to be specified in advance.
- Sensitive to the initial placement of centroids.

Limitations: Does not perform well with non-spherical clusters or outliers.

Results Analysis: K-Means clustering results were used to group students based on performance, though specific performance metrics are not included in this report as it is an unsupervised learning method.




```
   STUDENTID  AGE  GENDER  HS_TYPE  SCHOLARSHIP  WORK  ACTIVITY  PARTNER  \
0   STUDENT1    2       2        3            3     1         2        2
1   STUDENT2    2       2        3            3     1         2        2
2   STUDENT3    2       2        2            3     2         2        2
3   STUDENT4    1       1        1            3     1         2        1
4   STUDENT5    2       2        1            3     2         2        1

   SALARY  TRANSPORT  ...  PREP_EXAM  NOTES  LISTENS  LIKES_DISCUSS  \
0       1          1  ...          1      3        2              1
1       1          1  ...          1      3        2              3
2       2          4  ...          1      2        2              1
3       2          1  ...          2      3        2              2
4       3          1  ...          1      2        2              2

   CLASSROOM  CUML_GPA  EXP_GPA  COURSE ID  GRADE  Cluster
0          2         1        1          1      1        2
1          2         2        3          1      1        2
2          1         2        2          1      1        2
3          1         3        2          1      1        0
4          1         2        2          1      1        2

[5 rows x 34 columns]
```

3.6 Confusion Matrix

Introduction:

A **Confusion Matrix** is a performance evaluation tool for classification models. It displays the actual versus predicted classifications made by a model, providing detailed insight into the accuracy and types of errors made. It is a square matrix with rows representing actual classes and columns representing predicted classes.

Working Procedure:

1. **True Positive (TP)**: Correctly predicted positive class.
2. **True Negative (TN)**: Correctly predicted negative class.

3. **False Positive (FP)**: Incorrectly predicted as positive when it's actually negative (Type I error).
4. **False Negative (FN)**: Incorrectly predicted as negative when it's actually positive (Type II error).
5. The confusion matrix is generated by counting these occurrences for each class and is structured in a matrix to compare actual and predicted classifications.

Pros:

- Provides a detailed breakdown of correct and incorrect predictions.
- Highlights the type of errors made by the classifier (false positives/negatives).
- Useful for multi-class classification as it shows performance across all classes.

Cons:

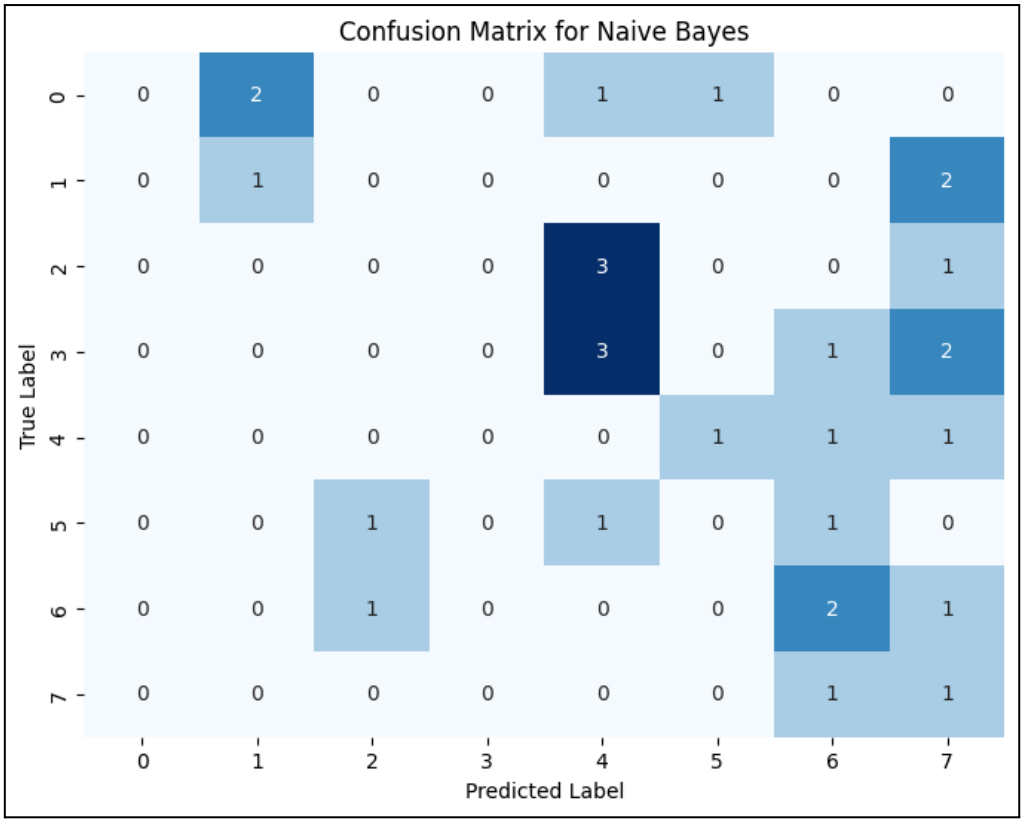
- Can be complex to interpret when the number of classes increases.
- Requires a large enough sample size to capture meaningful trends.
- Does not provide a single performance metric like accuracy but gives more detailed insights.

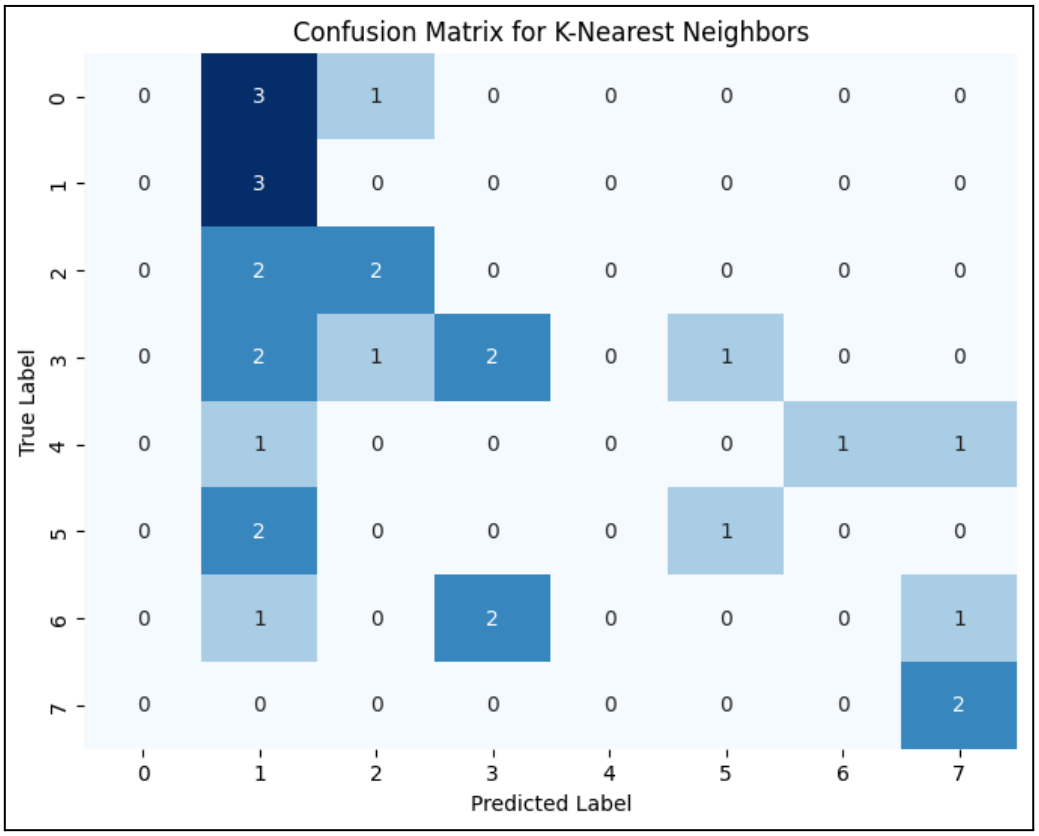
Limitations:

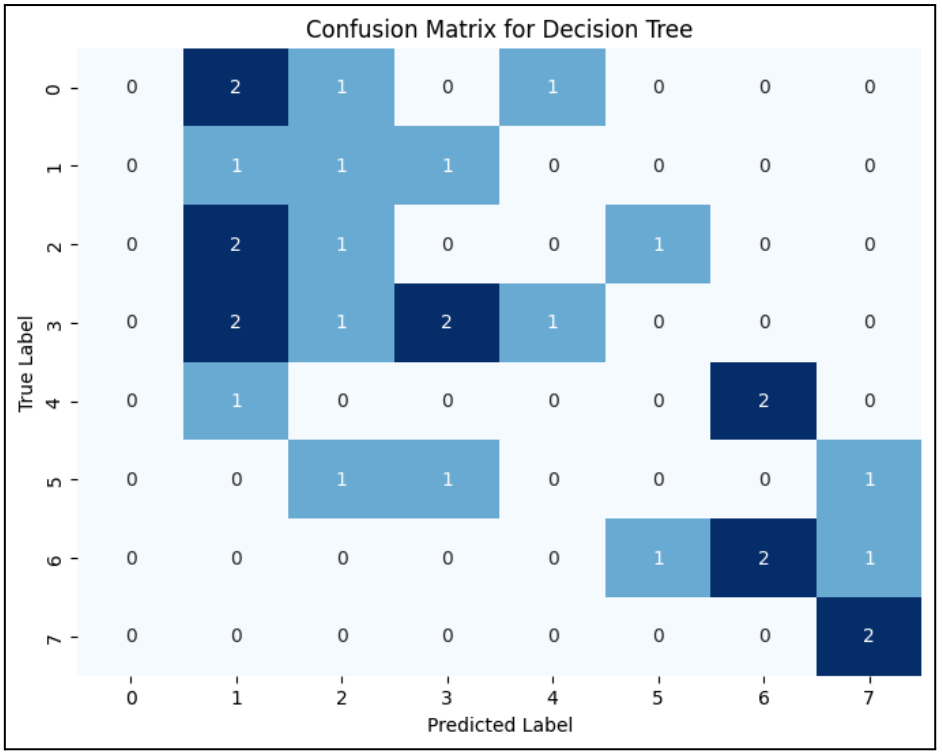
- **Class Imbalance**: If one class is heavily represented, the confusion matrix can be misleading without additional metrics like precision, recall, or F1-score.
- **Doesn't work for unsupervised learning algorithms**: Confusion matrices are only applicable to supervised learning algorithms where actual labels are known.

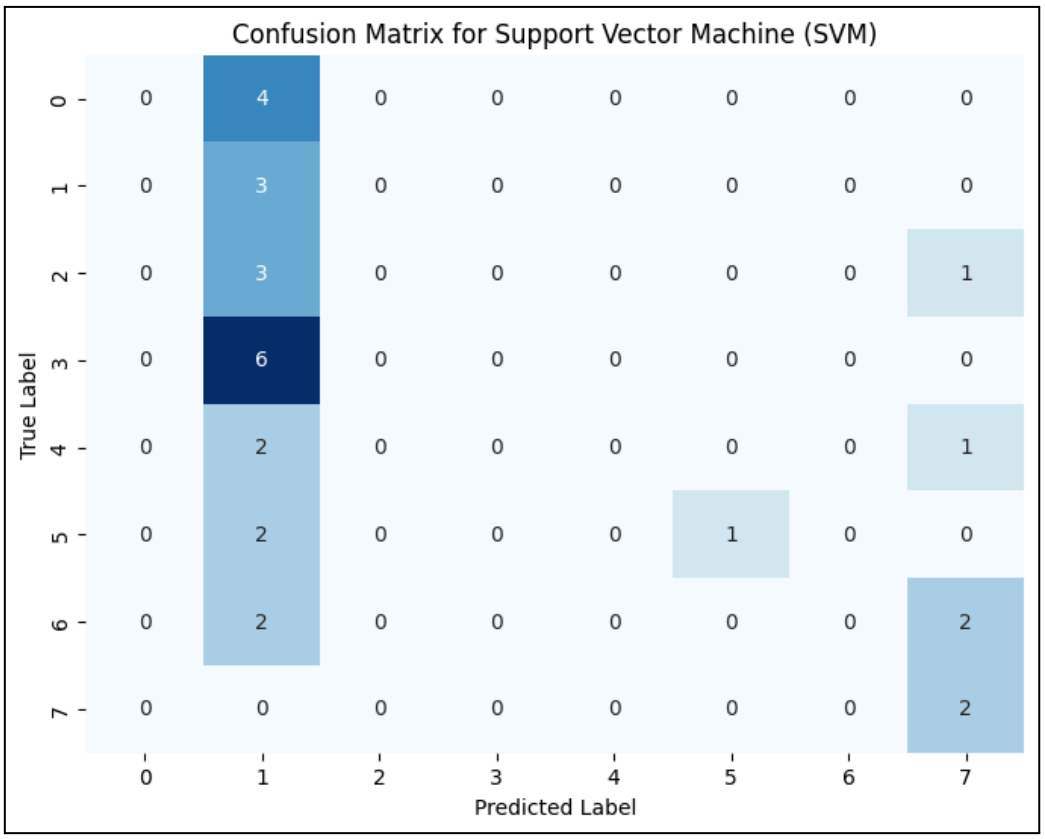
Results Analysis:

- The confusion matrix gives insight into how well the model predicts each class, identifying misclassified instances.
- It highlights if the model is overpredicting certain classes (leading to false positives) or underpredicting others (leading to false negatives).
- It helps assess model performance beyond just accuracy, especially in cases with imbalanced classes where other metrics (like precision, recall, and F1-score) are more informative.









Example:

In our experiments with models like K-Nearest Neighbors and Decision Trees, confusion matrices provided insight into specific class misclassifications, showing where the model struggled to distinguish between certain grades or outcomes. This highlighted areas for model improvement and emphasized the importance of using a range of evaluation metrics.

4. Performance Analysis and Result Interpretation

4.1 Summary of Results

- **Naive Bayes**: 13.79% accuracy; performed poorly across most classes, with many classes showing zero precision and recall.
- **KNN**: 34.48% accuracy; showed some ability to classify certain classes correctly, particularly class '1' and '2'.
- **Decision Tree**: 27.59% accuracy; some performance in specific classes but overall limited.
- **SVM**: 20.69% accuracy; struggled with class separation and demonstrated low precision for most classes.

4.2 Explanation of Results

The low accuracy and performance across various algorithms suggest a potential class imbalance in the dataset, with some classes being underrepresented. The algorithms struggled to classify these minority classes effectively.

4.3 Analysis of Performance

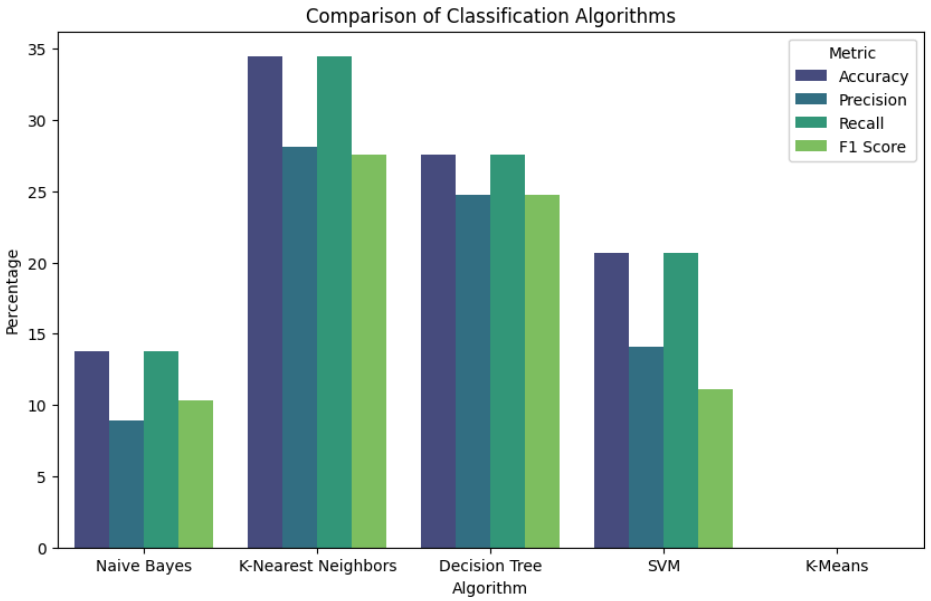
- The **KNN** algorithm outperformed the others, indicating its suitability for this specific dataset. However, its performance still highlights the challenges of classifying data with inherent noise and possible overlap among classes.
- The **Naive Bayes** and **SVM** algorithms displayed significantly lower accuracy, suggesting a misalignment between their assumptions (independence and linear separability) and the characteristics of the dataset.
- **Decision Trees** provided moderate performance but are prone to overfitting, as reflected in their results.

5. Comparison of All Algorithms

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Naive Bayes	13.79	8.91	13.79	10.34
K-Nearest Neighbors	34.48	28.08	34.48	27.56
Decision Tree	27.59	24.74	27.59	24.72
Support Vector Machine	20.69	14.05	20.69	11.10

From the comparison table, it's evident that KNN achieved the highest accuracy, while Naive Bayes performed the worst. This disparity highlights the importance of choosing the right algorithm based on dataset characteristics and the nature of the problem.

To visually compare the performance of different algorithms based on key metrics like accuracy, precision, recall, and F1-score, we can plot a bar chart.



Explanation:

1. **Data Setup**: A dictionary stores the `accuracy`, `precision`, `recall`, and `F1-score` for each algorithm.
2. **Visualization**: We use `seaborn`'s `barplot` to create a grouped bar chart, where each bar represents a different algorithm and color groups represent different evaluation metrics.
3. **Unsupervised K-Means**: Since K-Means clustering is unsupervised, accuracy-related metrics are not applicable, and its values are set to `0` for comparison purposes.

6. Conclusion

This lab provided a detailed exploration of several machine-learning algorithms applied to a student performance dataset. While K-Nearest Neighbors emerged as the most effective classifier, all models exhibited challenges in achieving high accuracy and classifying certain groups.

The findings emphasize the necessity of careful algorithm selection and dataset preparation in the application of machine learning techniques.