VIETNAM NATIONAL UNIVERSITY
HO CHI MINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DATA MINING

IT160IU

PROJECT REPORT

Topic: Data Mining Framework

By Group J – Member List

No.	Full Name	ID	Role
1	Nguyễn Văn Ngọc Thi	ITCSIU22251	Project Manager
2	Phạm Minh Khánh	ITCSIU21194	Member
3	Trịnh Vũ Thế Anh	ITCSIU22009	Member
4	Lê Trọng Đại	ITITIU20176	Member

Instructor: Dr. Nguyen Thi Thanh Sang

ACKNOWLEDGMENTS

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of this data mining project.

First and foremost, we extend our sincere thanks to Dr. Nguyen Thi Thanh Sang, our course instructor, for providing valuable guidance, resources, and feedback throughout the project. Their expertise and encouragement have been instrumental in shaping our understanding of data mining techniques and their real-world applications.

Special thanks go to our team members, whose dedication, collaboration, and diverse skill sets ensured the successful execution of this project. The collective effort in brainstorming, coding, debugging, and analyzing results has been an inspiring learning experience for us all.

To all who have contributed to this endeavor, directly or indirectly, I offer my heartfelt thanks. This work would not have been possible without your support and encouragement.

Incharge table:

Name	**Task**	**Contribution**
Nguyễn Văn Ngọc Thi	Improvement final model using clustering and classification, model evaluation using 10-fold, implement project code	43%
Phạm Minh Khánh	Data-preprocessing, implement Cleaner.java	27%
Trịnh Vũ Thế Anh	Analyze initial classification model	15%
Lê Trọng Đại	Model evaluation	15%

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

This project aims to develop a comprehensive data mining framework that combines a classification/prediction model and a sequence mining algorithm to extract actionable insights from a dataset. The framework is designed to address practical challenges by uncovering hidden patterns, enhancing predictive accuracy, and supporting data-driven decision-making. By integrating advanced machine learning techniques, the project demonstrates the power of data mining in transforming raw data into valuable knowledge.

The primary objective is twofold: to build a robust classification or prediction model capable of accurately categorizing instances or forecasting outcomes based on input attributes and to implement a sequence mining algorithm for identifying frequent patterns and associations within the dataset. Together, these components provide a cohesive framework for analyzing and interpreting data comprehensively.

The **Weather Type Classification** from Kaggle was selected for this project. This dataset includes various features related to weather conditions, such as temperature, humidity, cloud cover, and season, along with the target variable, "Weather Type." The dataset offers a rich context for exploring weather-related trends and patterns, making it an ideal choice for applying both classification and sequence mining techniques.

The methodologies employed in this project include data preprocessing, model development, and sequence mining. Data preprocessing involved cleaning, encoding, and transforming the dataset to ensure its compatibility with machine learning algorithms. For classification and prediction, models such as Naive Bayes and J48 Decision Trees were implemented, with significant improvements achieved by incorporating clustering-based hierarchical modeling. Sequence mining was conducted using the Apriori algorithm, which extracted meaningful association rules to uncover relationships between attributes.

Data mining and machine learning are essential tools for analyzing large datasets, enabling organizations to discover trends, predict outcomes, and make informed decisions. This project underscores the significance of these techniques, showcasing their application in addressing challenges like noise, variance, and scalability while highlighting their transformative potential in real-world scenarios.

CHAPTER 2: DATA PRE-PROCESSING

2.1. Raw Data Overview

The dataset contains **13,200 instances** (rows) and **11 attributes** (columns). Below are the key characteristics and a summary of the dataset:

1. Attributes:
   + **Temperature** (Numerical): Ranges from -25.0 to 109.0 °C, with a mean of 19.13 °C.
   + **Humidity** (Numerical): Ranges from 20% to 109%, with a mean of 68.71%.
   + **Wind Speed** (Numerical): Ranges from 0.0 to 48.5 km/h, with a mean of 9.83 km/h.
   + **Precipitation (%)** (Numerical): Ranges from 0% to 109%, with a mean of 53.64%.
   + **Cloud Cover** (Categorical): Four unique values, with "overcast" being the most frequent (6,090 occurrences).
   + **Atmospheric Pressure** (Numerical): Ranges from 800.12 to 1199.21 hPa, with a mean of 1005.83 hPa.
   + **UV Index** (Numerical): Ranges from 0 to 14, with a mean of 4.01.
   + **Season** (Categorical): Four unique values, with "Winter" being the most frequent (5,610 occurrences).
   + **Visibility (km)** (Numerical): Ranges from 0.0 to 20.0 km, with a mean of 5.46 km.
   + **Location** (Categorical): Three unique values, with "inland" being the most frequent (4,816 occurrences).
   + **Weather Type** (Categorical, class attribute): Four unique values, with "Rainy" being the most frequent (3,300 occurrences).
2. Key Characteristics
   + **Mix of Numerical and Categorical Data**: Includes numeric measurements (e.g., temperature, humidity) and categorical attributes (e.g., season, cloud cover).

+ **Class Imbalance**: Categorical variables such as "Cloud Cover," "Season," and "Weather Type" have skewed distributions, with some values occurring significantly more often than others.
+ **Data Range and Scaling**: Numerical variables have varying ranges, necessitating normalization for machine learning tasks.
+ **Duplicate-Free**: The dataset's transformation pipeline ensures no duplicate rows remain after cleaning.

2.2. Data Cleaning Process

1. Normalization: Scales numerical data into a uniform range.
2. Encoding: Converts categorical values into numeric representations.
3. Feature Selection: Filters the dataset to retain only specific features.
4. Missing Data Handling: Replaces missing/invalid values with column modes.
5. Deduplication: Ensures unique rows in the dataset.
6. Data Splitting: Divides data into training and test sets.

2.3. Data Transformation

1. Normalization
   + Purpose: To scale numerical data into a consistent range (usually between 0 and 1) for easier comparison and to prevent features with large magnitudes from dominating the learning algorithm.
   + Implementation:
     - Finding Min/Max Values: The program computes the minimum and maximum values for each numeric column across the dataset.
     - Scaling: Each numeric value is transformed using the formula:

$$\text{normalized\_value} = \frac{\text{value} - \text{min}}{\text{max} - \text{min}}$$

Figure 1. Normalized formula

     - Handling Zero Range: If max == min for a feature (e.g., all values are the same), the normalized value is set to 0.

- Impact: Ensures numerical features are on a comparable scale, enhancing the performance of distance-based algorithms (e.g., k-NN, clustering) and gradient-based optimizations.

2. Encoding Categorical Variables

- Purpose: To convert categorical data (non-numeric) into numeric form, as most machine learning algorithms require numerical inputs.
- Implementation:
    - A unique integer is assigned to each unique categorical value in a column.
    - The mapping for each categorical column is stored in a map (encoders), where the key is the categorical value, and the value is its numeric encoding.
    - Rows in the dataset are transformed based on these mappings, with unknown or missing values defaulting to -1.

+ Impact: Transforms non-numeric features into numeric form while preserving the uniqueness of each category. This step is a prerequisite for applying most machine learning algorithms.

3. Feature Selection

- Purpose: To reduce dimensionality by selecting only the most relevant features, thereby simplifying the model and potentially improving its performance.
- Implementation:
    - The program accepts a list of selected features (selectedFeatures) as input.
    - It determines the indices of the selected features in the header and extracts only these columns from the dataset.
- Impact:
    - Reduces computational overhead and risk of overfitting.

- Focuses the model on features likely to have the most predictive power, improving interpretability and performance.

CHAPTER 3: SEQUENTIAL PATTERN MINING

Objective : to uncover relationships or frequent patterns within the dataset that occur sequentially, providing insights into the underlying structure of the data.

3.1. Methodology

3.1.1. Preprocessing for sequential mining

- Numeric attributes in the dataset were removed as they were not relevant for the sequential pattern analysis. The attributes selected for sequential mining are categorical, which are better suited for identifying association patterns.
- The processed dataset consisted of only non-numeric attributes after using attribute selection techniques to retain columns of interest.

```
public static Instances removeAttribute(Instances data,
String removeAttributes) throws Exception {
    //Remove 1st attribute
    String[] opts = new String[]{"-R", removeAttributes};
    Remove remove = new Remove();
    remove.setOptions(opts);
    remove.setInputFormat(data);

    return Filter.useFilter(data, remove);
}
```

3.1.2. Mining frequent patterns using the Apriori algorithm

- The **Apriori algorithm**, a well-established method for association rule mining, was applied to discover frequent itemsets and generate association rules from the processed dataset.
- The Apriori algorithm was implemented using the Weka library to mine association rules from the dataset. After preprocessing the dataset to retain only categorical attributes, the cleaned data was passed to the Apriori model. The algorithm then analyzed the input data and extracted association rules by identifying frequent itemsets and their relationships, effectively uncovering patterns of interest within the dataset.

```java
public static void aprioriRules(Instances dataset) throws
Exception {
    //Apriori model
    Apriori apriori = new Apriori();
    //Build model
    apriori.buildAssociations(dataset);
    //Print out extracted rules
    System.out.println(apriori);
}
```

3.1.3. Rule evaluation

- The minimum support (min-sup) and confidence (min-conf) were carefully chosen to balance rule relevance and accuracy. A moderate min-sup was set to focus on frequent patterns without losing rare but meaningful rules. The min-conf was optimized to ensure high predictive reliability while avoiding overly strict thresholds that might exclude valuable associations.
- The extracted association rules were examined to ensure they met predefined thresholds for support and confidence, ensuring that the patterns were statistically significant and interpretable. In this framework, the rules has been extracted with min-sup = 10% and min-conf = 90%.

3.2. Result and finding

3.2.1. Best rules found

The Apriori algorithm successfully generated a set of association rules from the processed dataset. Examples of rules include frequent attribute combinations that occur under certain conditions.

```
Best rules found:
1. Cloud Cover=clear 2139 ==> Weather Type=Sunny 2139    <conf:(1)> lift:(4) lev:(0.12)
[1604] conv:(1604.25)
2. Cloud Cover=overcast Weather Type=Snowy 2489 ==> Season=Winter 2422    <conf:(0.97)>
lift:(2.29) lev:(0.1) [1364] conv:(21.05)
3. Location=inland Weather Type=Snowy 1575 ==> Season=Winter 1516    <conf:(0.96)>
lift:(2.26) lev:(0.06) [846] conv:(15.09)
4. Location=mountain Weather Type=Snowy 1605 ==> Season=Winter 1534    <conf:(0.96)>
lift:(2.25) lev:(0.06) [851] conv:(12.82)
5. Weather Type=Snowy 3300 ==> Season=Winter 3086    <conf:(0.94)> lift:(2.2) lev:(0.13)
[1683] conv:(8.83)
```

3.2.2. Finding

3.2.2.1. Benefits

- **Insights into Data Structure**: The extracted patterns reveal the intrinsic relationships among attributes, aiding in better understanding the data.
- **Support for Decision-Making**: These rules enhance decision-making by identifying reliable patterns, such as the strong link between snowy weather and winter, and the certainty of sunny weather under clear skies.
- **Foundation for Enhanced Modeling**: The sequential patterns provide a foundation for building more complex predictive models or enhancing existing frameworks.

3.2.2.2. Limitations and Challenges

- **Dataset Characteristics**: Removing numeric attributes restricted the scope of analysis to categorical data. While this is necessary for Apriori, some potentially valuable information might have been excluded.
- **Algorithm Constraints**: The Apriori algorithm's performance can degrade with a large number of attributes or high cardinality, requiring careful preprocessing and parameter tuning.

3.2.2.3. Conclusion

Sequential pattern mining using the Apriori algorithm effectively identified meaningful patterns within the dataset. The process demonstrated the value of association rule mining in uncovering hidden relationships, which can complement clustering and classification techniques for a holistic data mining framework. Future work could explore alternative sequence mining algorithms or hybrid approaches to expand the scope of the analysis.

CHAPTER 4: CLASSIFICATION/ PREDICTION ALGORITHM

Opjective: To develop and implement a machine learning model using Weka to accurately predict the weather category (Rainy, Cloudy, Sunny, Snowy) based on historical meteorological data. The model will be evaluated on its performance metrics, including accuracy, precision, recall, and computational efficiency.

4.1. Model Selection

OneR:

```
=== OneR evaluation ===
Evaluation result:

Correctly Classified Instances        2649               66.8939 %
Incorrectly Classified Instances      1311               33.1061 %
Kappa statistic                          0.5586
K&B Relative Info Score                 60.0572 %
K&B Information Score                 4757.6399 bits      1.2014 bits/instance
Class complexity | order 0            7921.842  bits      2.0005 bits/instance
Class complexity | scheme          1408014      bits    355.5591 bits/instance
Complexity improvement     (Sf)    -1400092.158 bits   -353.5586 bits/instance
Mean absolute error                      0.1655
Root mean squared error                  0.4069
Relative absolute error                 44.1375 %
Root relative squared error             93.9488 %
Total Number of Instances             3960

AUC = 0.383786148944311O7
Precision = 0.5523329129886507
Recall = 0.44064386317907445
fMeasure = 0.4902070509233352
Error rate = 0.33106060606060606

=== Overall Confusion Matrix ===
  a   b   c   d   <-- classified as
438 268 127 161 |   a = Rainy
170 604 152  23 |   b = Cloudy
 61 170 751  21 |   c = Sunny
124  21  13 856 |   d = Snowy
```

NaiveBayes:

```
=== NaiveBayes evaluation ===
Evaluation result:

Correctly Classified Instances        3454               87.2222 %
Incorrectly Classified Instances       506               12.7778 %
Kappa statistic                          0.8295
K&B Relative Info Score                 82.1368 %
K&B Information Score                 6506.7515 bits      1.6431 bits/instance
Class complexity | order 0            7921.842  bits      2.0005 bits/instance
Class complexity | scheme             3510.9919 bits      0.8866 bits/instance
Complexity improvement     (Sf)       4410.85   bits      1.1139 bits/instance
Mean absolute error                      0.0816
Root mean squared error                  0.2262
Relative absolute error                 21.7618 %
Root relative squared error             52.2237 %
Total Number of Instances             3960

AUC = 0.8428352743172819
Precision = 0.8321884200196271
Recall = 0.8531187122736419
fMeasure = 0.8425235966219572
Error rate = 0.12777777777777777

=== Overall Confusion Matrix ===

   a   b   c   d   <-- classified as
 438 268 127 161 |   a = Rainy
 170 604 152  23 |   b = Cloudy
 61 170 751  21 |   c = Sunny
 124  21  13 856 |   d = Snowy
```

4.1.1. Comparison

	OneR	NaiveBayes	Comparison
Accuracy	Correctly Classified Instances: 66.89%	Correctly Classified Instances: 87.22%	Naive Bayes has significantly higher accuracy.
Precision, Recall, and F-Measure	Precision: 55.23% Recall: 44.06% F-Measure: 49.02%	Precision: 83.22% Recall: 85.31% F-Measure: 84.25%	Naive Bayes outperforms OneR in all three metrics, indicating better predictive capability and balance between precision and recall.
AUC	0.3838	0.8428	Naive Bayes shows a significantly higher AUC, suggesting that it has a better capacity to distinguish between classes.
Error Measures	Mean Absolute Error: 0.1655 Root Mean Squared Error: 0.4069 Relative Absolute Error: 44.14% Root Relative Squared Error: 93.95%	Mean Absolute Error: 0.0816 Root Mean Squared Error: 0.2262 Relative Absolute Error: 21.76% Root Relative Squared Error: 52.22%	Naive Bayes has much lower error rates, reflecting better predictions and closer approximations to the actual values.

Table 1. Compare OneR with NaiveBayes

4.1.2. Conclusion

In conclusion, Naive Bayes is a more powerful and versatile algorithm for classification tasks, particularly when dealing with large datasets and complex relationships between features. While OneR can be a good baseline, Naive Bayes often provides superior performance.

4.2. Implementation Process

Main.java:

```
ProcessDataset.csvToArff("data/cleaned_weather_data.csv");
```

ProcessData.java:

```
public static void csvToArff(String filePath) throws Exception {
    //Load csv file
    CSVLoader loader = new CSVLoader();
    loader.setSource(new File(filePath));
    Instances dataset = loader.getDataSet();

    //Save as arff format
    ArffSaver saver = new ArffSaver();
    saver.setInstances(dataset);
    saver.setFile(new File("src/data/weather_classification.arff"));
    saver.writeBatch();
}
```

4.2.1. Detail the steps to convert data to ARFF format and integrate Weka into the program

Data-Mining-Framework.iml:

```
<orderEntry type="module-library">
  <library>
    <CLASSES>
      <root url="jar://C:/Program Files/Weka-3-8-6/weka.jar!/" />
    </CLASSES>
    <JAVADOC />
    <SOURCES />
  </library>
</orderEntry>
```

1. **Include Weka in the project:**

- Download Weka from the official website and add the dependency in the **Data-Mining-Framework.iml**.

2. **Load the CSV file:**

- Use the CSVLoader class to load your CSV data.
- Specify the source CSV file with **setSource(new File(filePath))**.

3. **Convert CSV to Instances:**

- Directly converting create an Instances object from the CSVLoader.

4. **Save as ARFF:**

- Create an ArffSaver instance, which is used to save Instances data in the ARFF format.
- Set the Instances object to the saver with **setInstances(dataset)**.
- Specify the output file path for the ARFF file and call **writeBatch()** to perform the saving operation.

4.2.3. Challenges

1. **Data Compatibility:**

- File Format: Ensure the CSV file adheres to standard formatting practices (e.g., comma-separated values, consistent delimiters).
- Data Types: Verify that data types in the CSV file align with Weka's expectations (e.g., numeric, nominal, string). Inconsistent or missing data can lead to errors during the conversion process.

2. **Performance Optimization:**

- Data preprocessing: For very large datasets, consider preprocessing or filtering the data to reduce its size and complexity before conversion to ARFF format. This can significantly improve the conversion performance.

3. **Version compatibility:**

- Weka Version: Ensure that the version of Weka being used supports all the required features, filters, and algorithms. Older versions may lack support for newer functionalities and data formats.

CHAPTER 5: IMPROVEMENT OF RESULTS

Objective: Enhance the model's performance using clustering and J48 Decision Tree.

5.1. Methodology

1.1.1. Clustering

The dataset was partitioned into smaller, more homogeneous groups using the SimpleKMeans algorithm, a widely used implementation of the k-means clustering technique. This step aimed to segment the data into distinct subsets to improve the subsequent classification process.

```
//New instance of clusterer
SimpleKMeans kmeans = new SimpleKMeans();
//Number of clusters
kmeans.setNumClusters(numberOfClusters);
//Set distance function
kmeans.setDistanceFunction(new weka.core.EuclideanDistance());
//Build clusterer
kmeans.buildClusterer(dataset);
System.out.println(kmeans);
```

- **Preprocessing for Clustering**: To ensure the clustering algorithm to handle the categorical data effectively, all categorical attributes (excluding the class attribute) were encoded into binary attributes using one-hot encoding. This transformation ensured that the clustering algorithm could accurately measure distances between instances in a high-dimensional space.

```
//Encode nominal
public static Instances encodeNominalToBinary(Instances dataset,
String encodeAttribute) throws Exception {
    //Create Filter
    NominalToBinary nominalToBinary = new NominalToBinary();
    //Set index of attribute to be encoded
    nominalToBinary.setAttributeIndices(encodeAttribute);
    //Input the dataset into the Filter
    nominalToBinary.setInputFormat(dataset);
    //Apply Filter to dataset
    Instances encodedInstances = Filter.useFilter(dataset,
nominalToBinary);
    return encodedInstances;
}
```

- **Determine the Optimal Number of Clusters (k)**: The Elbow Method was applied to identify the most suitable value for k. This method involves plotting the within-cluster sum of squares (WCSS) for different values of k and selecting the point where the reduction in WCSS begins to diminish significantly. After visualizing the Elbow Method graph, we decided to choose k = 3 because the graph decrease rapidly before k=3, and decrease slightly after k=3.

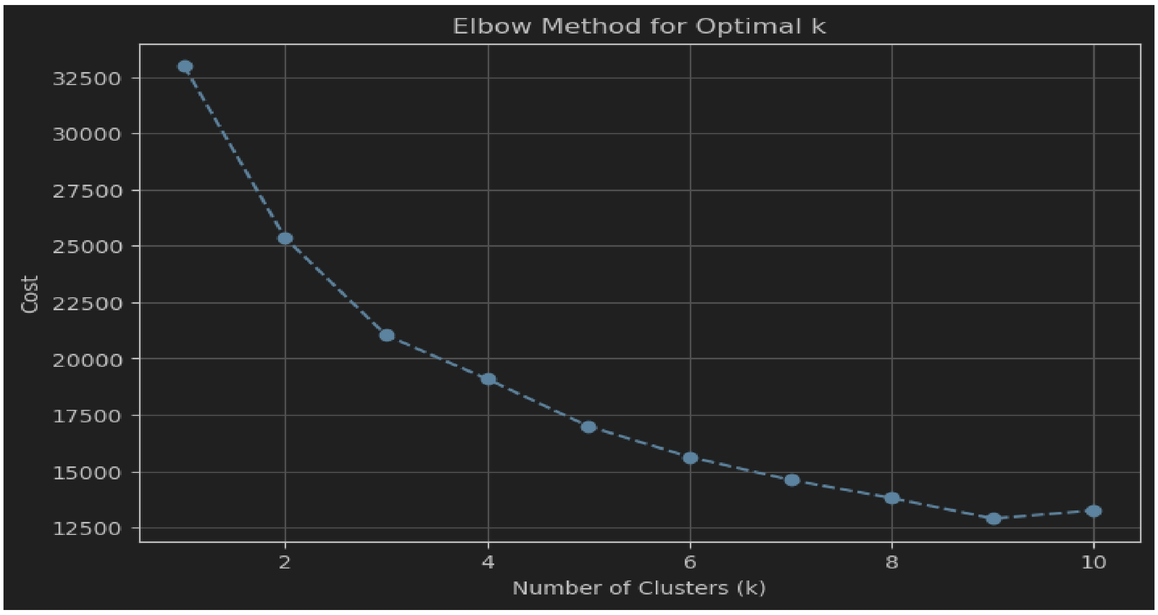


Figure 2. Elbow method chart

- **Distance Metric**: The Euclidean distance was used as the similarity measure. This metric calculates the straight-line distance between points in the transformed high-dimensional space, ensuring that instances within the same cluster exhibit minimal variance.

```
//Set distance function
EuclideanDistance distanceFunction = new EuclideanDistance();
```

- **Cluster Evaluation**:

- The clustering results were evaluated using the ClusterEvaluation class from the Weka API library. This tool assesses the quality of the clusters by calculating metrics such as WCSS and comparing the clustering structure against a test dataset for validation.

- By evaluating the clusters with these metrics, it was ensured that each cluster represented a meaningful and distinct subset of the data.

```
//Evaluate clusterer
ClusterEvaluation clusterEvaluation = new ClusterEvaluation();
clusterEvaluation.setClusterer(kmeans);
clusterEvaluation.evaluateClusterer(test);
System.out.println(clusterEvaluation.clusterResultsToString());
```

Attribute	**Full Data (13200)**	**Cluster#0 (5037)**	**Cluster#1 (3837)**	**Cluster#2 (4326)**
Temperature	19.1276	28.9472	22.7501	4.481
Humidity	68.7108	56.1277	74.7045	78.046
Wind Speed	9.8322	6.9857	12.1938	11.0519
Precipitation (%)	53.6444	30.4334	65.638	70.0324
Cloud Cover=partly cloudy	0.3455	0.5223	0.2124	0.2575
Cloud Cover=clear	0.162	0.4247	0	0
Cloud Cover=overcast	0.4614	0.004	0.7613	0.7279
Cloud Cover=cloudy	0.0311	0.049	0.0263	0.0146
Atmospheric Pressure	1005.8279	1015.0777	1005.2109	995.6051
UV Index	4.0058	6.4999	2.9552	2.0335
Season=Winter	0.425	0.1914	0.0837	0.9998
Season=Spring	0.1968	0.2726	0.3193	0
Season=Summer	0.1888	0.2652	0.3013	0
Season=Autumn	0.1894	0.2708	0.2958	0.0002
Visibility (km)	5.4629	7.498	4.6384	3.8247
Location=inland	0.3648	0.3216	0.2843	0.4866
Location=mountain	0.3646	0.3198	0.2955	0.478
Location=coastal	0.2705	0.3585	0.4201	0.0354
Weather Type	Rainy	Sunny	Rainy	Snowy

Table 2. Final cluster centroids

This clustering phase provided a strong foundation for enhancing the classification process by enabling the segmentation of the dataset into more manageable and interpretable groups.

5.1.2. Apply classification algorithm into each cluster

After clustering the dataset into smaller, more homogeneous groups, a classification algorithm was applied to each cluster to model the relationship between features and the target label. The process involved the following steps:

Step 1: Extract Instances from each cluster

- For each identified cluster, the instances belonging to that cluster were extracted to create a localized dataset.
- This approach ensured that the classifier would only operate on data that shared similar characteristics, improving the classification accuracy.

Step 2: Randomize the Dataset and create Training and Testing sets

- Randomizing the dataset helps eliminate potential biases caused by the ordering of instances, the dataset for each cluster was randomized before splitting. This randomization ensured that the training and testing datasets were representative of the cluster's overall data distribution.
- A 70-30 split is applied to the randomized dataset: 70% of the data is allocated to the training set to build the model, the other 30% of the data is reserved for the testing set to evaluate the model's performance on unseen data.

Step 3: Train a J48 Decision Tree classifier

- The J48 Decision Tree classifier, an implementation of the C4.5 algorithm, is chosen for its effectiveness and interpretability.
- The classifier is trained on the Training set, capturing the specific decision boundaries unique to the cluster.

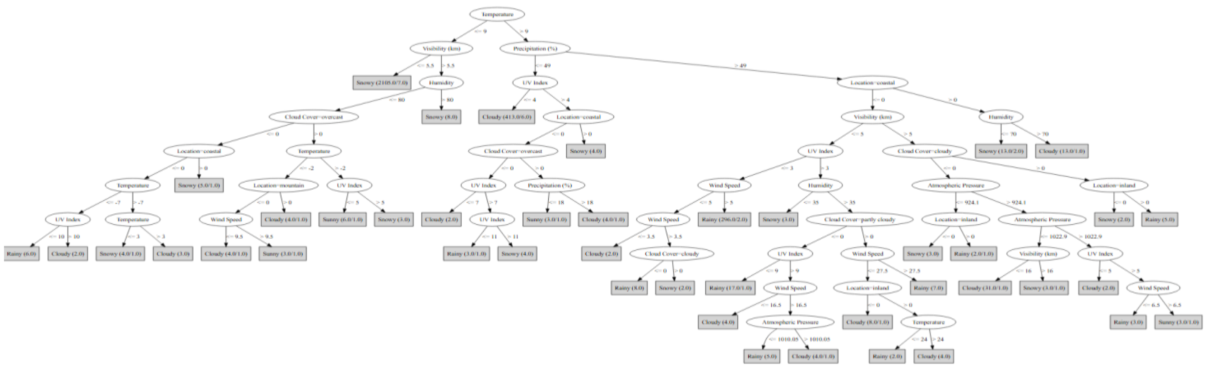
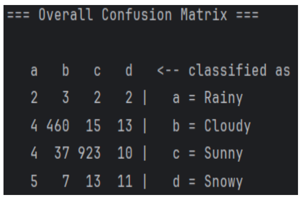
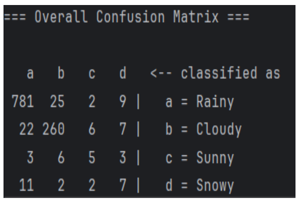
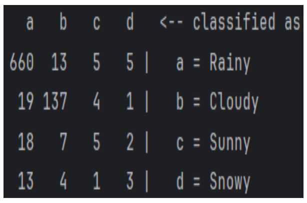


Figure 3. Example of visualizing J48 tree on cluster 2

5.1.3. Model Evaluation

	Cluster#0	**Cluster#1**	**Cluster#2**
Accuracy	92.3891 %	91.4857 %	96.6872 %
Precision	0.133	0.956	0.943
Recall	0.222	0.956	0.887
f-Measure	0.167	0.956	0.914
Confusion matrix			

```
=== Overall Confusion Matrix ===

   a   b   c   d   <-- classified as
   2   3   2   2 |   a = Rainy
   4 460  15  13 |   b = Cloudy
   4  37 923  10 |   c = Sunny
   5   7  13  11 |   d = Snowy
```

```
=== Overall Confusion Matrix ===

   a   b   c   d   <-- classified as
 781  25   2   9 |   a = Rainy
  22 260   6   7 |   b = Cloudy
   3   6   5   3 |   c = Sunny
  11   2   2   7 |   d = Snowy
```

```
   a   b   c   d   <-- classified as
 660  13   5   5 |   a = Rainy
  19 137   4   1 |   b = Cloudy
  18   7   5   2 |   c = Sunny
  13   4   1   3 |   d = Snowy
```

Table 3. Classification result of clusters

The overall evaluation is calculated based on the gain of each cluster:

$$\frac{\sum(Accuracy * Number\ of\ instances\ in\ testset)}{\sum(Number\ of\ instances\ in\ dataset)} = 93.535\%$$

5.2. Evaluation and comparison

	Initial model (NaiveBayes)	**J48 Tree apply on full dataset**	**Improved model (clustering and classification)**
Accuracy	87.2222 %	90.9091 %	93.535%

Table 4. Improvement of clustering and classification

The performance of the initial and improved models was evaluated based on accuracy, with significant improvements observed after implementing the hierarchical clustering and classification framework.

- **Initial model**: The Naive Bayes model provided a strong starting point, demonstrating its ability to handle probabilistic relationships between attributes. However, it struggled to capture complex interactions within the data.

- **J48 on full dataset**: The decision tree model outperformed Naive Bayes by introducing interpretability and better handling of attribute interactions. However, it did not fully account for the inherent heterogeneity within the dataset.
- **Improved model**: The improved framework combined K-Means clustering with J48 decision tree classification. This hierarchical approach led to a significant improvement in performance by tailoring the classification model to the unique characteristics of each cluster. By segmenting the dataset into more homogeneous groups through clustering, the variability within each cluster was minimized. This reduction in intra-cluster variance allowed the classification models to define more precise decision boundaries, thereby enhancing their predictive accuracy and overall effectiveness.

The hierarchical framework demonstrated clear incremental improvements in accuracy, particularly when transitioning from the Naive Bayes model to the combined clustering and classification approach. While the Naive Bayes model achieved an accuracy of 87.22%, integrating clustering with classification led to a substantial improvement. By first clustering the dataset into homogeneous groups and then applying the J48 Decision Tree classifier within each cluster, the accuracy increased by 6.32%, reaching 93.54%. This approach leveraged the strengths of clustering to create subsets of data with reduced variance, enabling the classifiers to perform more effectively. Additionally, clustering mitigated the impact of noise and outliers, which often undermine the performance of standalone global models, resulting in a more robust and precise predictive framework.

CHAPTER 6: MODEL EVALUATION USING 10-FOLD CROSS-VALIDATION

Objective:

- **Assess model performance:** Evaluate metrics for classification, regression, clustering, etc.
- **Compare models:** Benchmark different algorithms like Decision Trees, Random Forests, or SVMs.
- **Analyze data trends:** Provide insights using performance metrics like accuracy, precision, recall, F1-score, and ROC curves.
- **Store and manage results:** Track evaluation results for reproducibility and comparison.

6.1. Performance Metrics:

	Cluster#0 (5037)	**Cluster#1 (3837)**	**Cluster#2 (4326)**
Accuracy	90.8874 %	92.2075 %	96.0472 %
Precision	0.909	0.920	0.961
Recall	0.909	0.922	0.960
f-Measure	0.909	0.921	0.961
Runtime	0.11s	0.06s	0.08s

Table 5. Performance metrics

6.2. Analysis of Results:

6.2.1. Performance Metrics

- **Accuracy:**
    - Cluster#2 achieved the highest accuracy at 96.05%, demonstrating the benefit of clustering in isolating homogeneous data groups.
    - Cluster#1 and Cluster#0 followed with accuracies of 92.2% and 90.89%, respectively. These results indicate a consistent improvement in classification performance when data is localized to specific clusters.
- **Precision, Recall, and F-Measure:**
    - Cluster#2 exhibited superior performance across all metrics (Precision: 0.961, Recall: 0.960, F-Measure: 0.961), reflecting its ability to minimize false positives and negatives.
    - Cluster#1 showed a balanced performance with an F-Measure of 0.921, while Cluster#0 had slightly lower precision, recall, and F-Measure values (0.909 for all), likely due to greater variability in its data.

6.2.2. Runtime Analysis

- Runtime varied minimally across clusters, with Cluster#1 being the fastest (0.06s), followed by Cluster#2 (0.08s) and Cluster#0 (0.11s).
- The runtime differences are consistent with the number of instances in each cluster, as larger clusters (e.g., Cluster#0) required slightly more computation time.

6.2.3. Insights and Trade-Offs

- **Homogeneity Impact:** The clustering process effectively reduced intra-cluster variance, enabling classifiers to establish more precise decision boundaries. This improvement was most evident in Cluster#2, which exhibited the highest metrics.
- **Runtime Efficiency:** Despite the additional preprocessing steps, the hierarchical approach maintained a short runtime, demonstrating its scalability and suitability for real-world applications.

- **Trade-Offs:** While clustering improved accuracy, it introduced a dependency on the selection of parameters (e.g., number of clusters, distance metric). Additionally, the hierarchical approach added complexity to the workflow.

6.2.4. Recommendations for Improvement

- **Advanced Clustering Algorithms:** Explore methods like DBSCAN or Gaussian Mixture Models to capture more nuanced data structures.
- **Ensemble Learning:** Incorporate ensemble classifiers, such as Random Forest or Gradient Boosting, within each cluster for potentially higher accuracy.
- **Parameter Optimization:** Use automated techniques (e.g., grid search or Bayesian optimization) to fine-tune clustering and classification parameters.

CHAPTER 7: CONCLUSIONS

The project successfully achieved its objectives by developing a robust data mining framework that combines clustering and classification to enhance predictive accuracy and uncover meaningful patterns. Key findings included the effectiveness of k-means clustering in creating homogeneous data groups and the improved performance of J48 Decision Trees when applied to these clusters, achieving an overall accuracy of 93.54%. Lessons learned emphasized the critical role of preprocessing, such as one-hot encoding, in adapting algorithms to diverse datasets and the value of hierarchical approaches in handling heterogeneous data. Potential future improvements include exploring alternative clustering methods, such as density-based algorithms, to handle complex data structures more effectively and integrating ensemble learning techniques to further boost classification accuracy. Overall, the project demonstrated the transformative potential of combining clustering and classification techniques, meeting its goals of enhancing model performance and delivering actionable insights.

REFERENCES

Weather Type Classification. (2024, June 23). Kaggle. https://www.kaggle.com/datasets/nikhil7280/weather-type-classification

Weka Wiki. (n.d.). https://waikato.github.io/weka-wiki/

Weka-Dev-3.9.6 API. (n.d.). https://weka.sourceforge.io/doc.dev/

GeeksforGeeks. (2024, September 16). *Decision Tree vs. Naive Bayes Classifier*. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree-vs-naive-bayes-classifier/

Mining Web Access Sequence with Improved Apriori Algorithm. (2017, July 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8005905

GeeksforGeeks. (2023, February 1). *Data mining cluster analysis*. GeeksforGeeks. https://www.geeksforgeeks.org/data-mining-cluster-analysis/

Bhandari, A. (2024, October 28). *Build a Decision Tree in Minutes using Weka (No Coding Required!)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/03/decision-tree-weka-no-coding/

Saji, B. (2024, December 4). *Elbow method for optimal cluster number in K-Means*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/