Experiment 1: Evaluate Information Gain of Attributes in Student Dataset

---

Aim:

To evaluate the **Information Gain (IG)** of each attribute in the student dataset and identify the **best attribute** to split for predicting **Result**.

Theory:

- **Information Gain (IG)** measures how much an attribute **reduces uncertainty** about the class (Result).
- The attribute with the **highest IG** is chosen for splitting in **decision tree algorithms** like ID3 or J48.

Dataset (student.arff)

```
@relation student

@attribute Attendance numeric
@attribute InternalMarks numeric
@attribute AssignmentScore numeric
@attribute SemesterMarks numeric
@attribute Result {Pass, Fail}

@data
80,75,70,85,Pass
60,65,60,70,Pass
50,55,50,45,Fail
90,80,85,90,Pass
70,60,65,60,Pass
45,50,55,50,Fail
85,75,80,88,Pass
```

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.

2. Click **Open File** → select **student.arff**.
3. Go to **Select Attributes tab**.
4. Choose **Attribute Evaluator → InfoGainAttributeEval**.
5. Choose **Search Method → Ranker**.
6. Click **Start** → WEKA calculates **Information Gain** for all attributes.
7. Identify the attribute with **highest IG** → best for splitting.

Result (Sample / Expected):

Attribute	**Information Gain**
Attendance	0.42
InternalMarks	0.56
AssignmentScore	0.35
SemesterMarks	0.60

- **Highest IG:** SemesterMarks → **best attribute to split** for predicting Result.

Conclusion:

- **SemesterMarks** is the most informative attribute for predicting **Pass/Fail**.
- Using the attribute with highest IG improves **classification accuracy** in decision trees.
- WEKA provides a **quick and easy way** to calculate Information Gain.

Experiment 2: Classification Using J48 Decision Tree Algorithm

---

Aim:

To demonstrate **classification** using the **J48 Decision Tree algorithm** on the Weather dataset and predict **Play** for a given test instance.

Theory:

- **J48** is a decision tree algorithm in WEKA (implementation of **C4.5**).
- It selects the **best attribute** to split at each node using **Information Gain**.
- Helps in **predicting class labels** based on input attributes.

Dataset (weather.arff)

@relation weather

@attribute Outlook {Sunny, Overcast, Rain}

@attribute Temperature numeric

@attribute Humidity numeric

@attribute Windy {True, False}

@attribute Play {Yes, No}

@data

Sunny,85,85,False,No

Sunny,80,90,True,No

Overcast,83,78,False,Yes

Rain,70,96,False,Yes

Rain,68,80,False,Yes

Rain,65,70,True,No

Overcast,64,65,True,Yes

Sunny,72,95,False,No

Sunny,69,70,False,Yes

Rain,75,80,False,Yes

Sunny,75,70,True,Yes

Overcast,72,90,True,Yes

Overcast,81,75,False,Yes

Rain,71,91,True,No

Class Attribute: Play (Yes/No)

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **weather.arff**.
3. Go to **Classify tab**.
4. Choose **Classifier → trees → J48**.
5. Click **Start** to build the decision tree.
6. Observe output:
   - Generated decision tree
   - Correctly classified instances
   - Confusion matrix
7. Predict **Play** for a new test instance (e.g., Outlook = Sunny, Temperature = 72, Humidity = 90, Windy = False) using the tree.

Result (Sample / Expected):

Generated J48 Decision Tree (Simplified):

Outlook = Sunny

| Humidity <= 75 : Yes

| Humidity > 75 : No

Outlook = Overcast : Yes

Outlook = Rain

| Windy = False : Yes

| Windy = True : No

Prediction Example:

- Test instance: Outlook = Sunny, Temperature = 72, Humidity = 90, Windy = False → **Play = No**

Conclusion:

- J48 Decision Tree effectively classifies the Weather dataset.
- The model can predict **Play** for new instances accurately.
- Using WEKA, building and visualizing decision trees is **quick and easy**.

Experiment 3: Classification Using ID3 Decision Tree Algorithm

---

Aim:

To demonstrate **classification** using the **ID3 Decision Tree algorithm** on the Weather dataset and derive **decision rules** for **Play**.

Theory:

- **ID3** is a decision tree algorithm that uses **Information Gain** to select the best attribute at each node.
- Suitable for **nominal/categorical attributes**.
- The resulting tree can be **converted into decision rules** for classification.

Dataset (weather.arff)

@relation weather

@attribute Outlook {Sunny, Overcast, Rain}

@attribute Temperature {Hot, Mild, Cool}

@attribute Humidity {High, Normal}

@attribute Windy {True, False}

@attribute Play {Yes, No}

@data

Sunny,Hot,High,False,No

Sunny,Hot,High,True,No

Overcast,Hot,High,False,Yes

Rain,Mild,High,False,Yes

Rain,Cool,Normal,False,Yes

Rain,Cool,Normal,True,No

Overcast,Cool,Normal,True,Yes

Sunny,Mild,High,False,No

Sunny,Cool,Normal,False,Yes

Rain,Mild,Normal,False,Yes

Sunny,Mild,Normal,True,Yes

Overcast,Mild,High,True,Yes

Overcast,Hot,Normal,False,Yes

Rain,Mild,High,True,No

Class Attribute: Play (Yes/No)

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **weather.arff**.
3. Ensure **all attributes are nominal**.
4. Go to **Classify tab**.
5. Choose **Classifier → trees → ID3**.
6. Click **Start** to build the decision tree.
7. Observe output:
    - Generated decision tree
    - Correctly classified instances
    - Confusion matrix
8. Derive **decision rules** from the tree for **Play**.

Result (Sample / Expected):

Generated ID3 Decision Tree (Simplified):

Outlook = Sunny

| Humidity = High : No

| Humidity = Normal : Yes

Outlook = Overcast : Yes

Outlook = Rain

| Windy = False : Yes

| Windy = True : No

Derived Decision Rules:

1. If Outlook = Sunny AND Humidity = High → Play = No
2. If Outlook = Sunny AND Humidity = Normal → Play = Yes

3. If Outlook = Overcast → Play = Yes
4. If Outlook = Rain AND Windy = False → Play = Yes
5. If Outlook = Rain AND Windy = True → Play = No

Conclusion:

- ID3 effectively classifies the Weather dataset with all nominal attributes.
- Decision tree can be easily converted into **decision rules**.
- WEKA simplifies the **tree generation and rule extraction** process.

Experiment 4: Prediction Using k-Nearest Neighbor (k-NN) Classification

---

Aim:

To predict the **Result** of a new student record using **k-Nearest Neighbor (k-NN)** and evaluate the model accuracy.

Theory:

- **k-NN** is a **lazy supervised learning algorithm**.
- Predicts the class of a new instance based on the **majority class of its k nearest neighbors**.
- Distance metric (e.g., **Euclidean distance**) is used to find nearest neighbors.
- Accuracy can be evaluated using **cross-validation**.

Dataset (student.arff)

@relation student

@attribute Attendance numeric

@attribute InternalMarks numeric

@attribute AssignmentScore numeric

@attribute SemesterMarks numeric

@attribute Result {Pass, Fail}

@data

80,75,70,85,Pass

60,65,60,70,Pass

50,55,50,45,Fail

90,80,85,90,Pass

70,60,65,60,Pass

45,50,55,50,Fail

85,75,80,88,Pass

Class Attribute: Result (Pass/Fail)

New Test Instance: 65, 70, 60, 68, ?

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **student.arff**.
3. Go to **Classify tab**.
4. Choose **Classifier → lazy → IBk (k-NN)**.
5. Set **k = 3** (or any suitable value).
6. Click **Start** to train the model and evaluate accuracy.
7. Use the trained model to **predict Result** for the new test instance.

Result (Sample / Expected):

- **Predicted Result:** Pass
- **Model Accuracy:** 100% (all instances correctly classified in this dataset)

Conclusion:

- k-NN accurately predicted the Result of a new student based on **nearest neighbors**.
- WEKA provides a **quick way** to implement and test k-NN classification.
- Accuracy depends on **k value** and dataset distribution.

Experiment 5: Prediction Using Bayesian Classification

Aim:

To predict the **species** of a new Iris sample using **Bayesian classification** and display the **probability of each class**.

Theory:

- **Bayesian classification** (Naive Bayes) is a **probabilistic classifier**.
- Assumes **attribute independence** and computes the probability of each class for a given instance.
- The class with the **highest probability** is selected as the predicted class.
- Useful for **quick and accurate classification** of datasets.

Dataset (iris.arff)

@relation iris

@attribute SepalLength numeric

@attribute SepalWidth numeric

@attribute PetalLength numeric

@attribute PetalWidth numeric

@attribute Species {Setosa, Versicolor, Virginica}

@data

5.1,3.5,1.4,0.2,Setosa

4.9,3.0,1.4,0.2,Setosa

6.2,3.4,5.4,2.3,Virginica

5.9,3.0,5.1,1.8,Virginica

6.0,2.2,4.0,1.0,Versicolor

5.5,2.3,4.0,1.3,Versicolor

...

New Test Sample: SepalLength=6.1, SepalWidth=2.9, PetalLength=4.7, PetalWidth=1.4

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **iris.arff**.
3. Go to **Classify tab**.
4. Choose **Classifier → bayes → NaiveBayes**.
5. Click **Start** to train the model.
6. Use **Supplied test instance** to predict **Species**.
7. Observe **probability distribution** for each class.

Result (Sample / Expected):

- **Predicted Species:** Versicolor
- **Probability of Each Class:**
    - Setosa: 0.01
    - Versicolor: 0.85
    - Virginica: 0.14

Conclusion:

- Bayesian classification predicted the **species** based on computed probabilities.
- Naive Bayes is effective for **numerical and categorical data**.
- WEKA provides **easy computation of class probabilities** and predictions.

Experiment 6: Feature Selection to Improve Model Performance

Aim:

To select **prominent feature subsets** from the Iris dataset to improve **classification model performance**.

Theory:

- **Feature selection** identifies the **most relevant attributes** for predicting the class.
- Reduces **overfitting**, improves **accuracy**, and decreases **training time**.
- WEKA provides **attribute selection tools** like **InfoGainAttributeEval + Ranker**.

Dataset (iris.arff)

```
@relation iris

@attribute SepalLength numeric
@attribute SepalWidth numeric
@attribute PetalLength numeric
@attribute PetalWidth numeric
@attribute Species {Setosa, Versicolor, Virginica}

@data
5.1,3.5,1.4,0.2,Setosa
4.9,3.0,1.4,0.2,Setosa
6.2,3.4,5.4,2.3,Virginica
5.9,3.0,5.1,1.8,Virginica
6.0,2.2,4.0,1.0,Versicolor
5.5,2.3,4.0,1.3,Versicolor
...
```

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.

2. Click **Open File** → select **iris.arff**.
3. Go to **Select Attributes tab**.
4. Choose **Attribute Evaluator → InfoGainAttributeEval**.
5. Choose **Search Method → Ranker**.
6. Click **Start** → WEKA ranks attributes based on importance.
7. Identify the **most prominent features** for classification.

Result (Sample / Expected):

Attribute	**Importance (Info Gain)**
PetalLength	0.9
PetalWidth	0.88
SepalLength	0.5
SepalWidth	0.3

- **Prominent Features:** PetalLength and PetalWidth → most useful for predicting Species.

Conclusion:

- Selecting **PetalLength and PetalWidth** improves model accuracy and reduces complexity.
- Feature selection helps in **better performance and faster training**.
- WEKA provides a simple interface to **rank and select attributes**

Experiment 7: Data Pre-processing on Customer Dataset

Aim:

To perform **pre-processing** on the Customer dataset, including handling missing values, normalization, discretization, standardization, removing unnecessary attributes, and encoding categorical attributes.

Theory:

- **Data pre-processing** improves dataset quality for data mining tasks.
- Common steps:
  - Handle **missing values**
  - **Normalize** numerical attributes
  - **Discretize** continuous values into categories
  - **Standardize** attributes for uniform scale
  - Remove **irrelevant attributes**
  - Encode **categorical attributes** to numerical values

Dataset (customer.arff)

@relation customer

@attribute CustomerID numeric

@attribute Age numeric

@attribute Gender {Male, Female}

@attribute AnnualIncome numeric

@attribute SpendingScore numeric

@attribute Segment {High, Medium, Low}

@data

101,25,Male,50000,70,Medium

102,30,Female,60000,60,High

103,22,Male,35000,40,Low

104,28,Female,58000,80,High

105,35,Male,45000,50,Medium

106,40,Female,62000,30,Low

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **customer.arff**.
3. Go to **Preprocess tab**.
4. **Handle missing values:** Use **Filter → unsupervised → attribute → ReplaceMissingValues**.
5. **Normalize numerical attributes:** Filter → unsupervised → attribute → **Normalize**.
6. **Discretization:** Filter → unsupervised → attribute → **Discretize**.
7. **Standardization:** Filter → unsupervised → attribute → **Standardize**.
8. **Remove unnecessary attributes:** Filter → unsupervised → attribute → **Remove** (e.g., CustomerID).
9. **Encode categorical attributes:** Filter → unsupervised → attribute → **NominalToBinary**.
10. Apply filters step by step and **save processed dataset** if needed.

Result (Sample / Expected):

- All **missing values handled**.
- Numerical attributes (**Age, AnnualIncome, SpendingScore**) **normalized and standardized**.
- Continuous attributes **discretized** into categories.
- **CustomerID removed**.
- Categorical attributes (**Gender, Segment**) encoded numerically.

Conclusion:

- Pre-processing ensures **clean and consistent dataset**.
- Improves **model accuracy and performance**.
- WEKA provides **easy tools** for all pre-processing tasks.

Experiment 8: Data Pre-processing on Iris Dataset

Aim:

To apply **data pre-processing techniques** on the Iris dataset, including handling missing values, normalization, and encoding categorical data.

Theory:

- **Data pre-processing** improves dataset quality for data mining and machine learning.
- Steps include:
    - **Handling missing values** to avoid errors
    - **Normalizing** numerical attributes for uniform scale
    - **Encoding categorical attributes** (e.g., Species) into numerical values

---

Dataset (iris.arff)

@relation iris

@attribute SepalLength numeric

@attribute SepalWidth numeric

@attribute PetalLength numeric

@attribute PetalWidth numeric

@attribute Species {Setosa, Versicolor, Virginica}

@data

5.1,3.5,1.4,0.2,Setosa

4.9,3.0,1.4,0.2,Setosa

6.2,3.4,5.4,2.3,Virginica

5.9,3.0,5.1,1.8,Virginica

6.0,2.2,4.0,1.0,Versicolor

5.5,2.3,4.0,1.3,Versicolor

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **iris.arff**.
3. Go to **Preprocess tab**.
4. **Handle missing values:** Filter → unsupervised → attribute → **ReplaceMissingValues**.
5. **Normalize numerical attributes:** Filter → unsupervised → attribute → **Normalize**.
6. **Encode categorical attribute (Species):** Filter → unsupervised → attribute → **NominalToBinary**.
7. Apply filters step by step and **save the pre-processed dataset**.

Result (Sample / Expected):

- All missing values **handled**.
- Numerical attributes (**SepalLength, SepalWidth, PetalLength, PetalWidth**) **normalized**.
- Categorical attribute (**Species**) **encoded numerically**.

Conclusion:

- Pre-processing improves **dataset quality and model performance**.
- WEKA makes it **easy to handle missing values, normalize data, and encode categories**.
- Processed dataset is ready for **classification or clustering** tasks.

Experiment 9: Back Propagation Neural Network on Student Performance Dataset

---

Aim:

To implement a **Back Propagation Neural Network (BPNN)** to train and update weights and biases for predicting **Result (Pass/Fail)**.

Theory:

- **Back Propagation Neural Network (BPNN)** is a **supervised learning model**.
- It consists of **input, hidden, and output layers**.
- Weights and biases are **updated iteratively** using **error correction** to minimize prediction error.
- Useful for **numerical prediction and classification**.

Dataset (student_performance.arff)

@relation student_performance

@attribute Attendance numeric

@attribute InternalMarks numeric

@attribute AssignmentScore numeric

@attribute SemesterMarks numeric

@attribute Result {Pass, Fail}

@data

80,75,70,85,Pass

60,65,60,70,Pass

50,55,50,45,Fail

90,80,85,90,Pass

70,60,65,60,Pass

45,50,55,50,Fail

85,75,80,88,Pass

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **student_performance.arff**.
3. Go to **Classify tab**.
4. Choose **Classifier → functions → MultilayerPerceptron**.
5. Configure network parameters:
    - **Learning rate** (e.g., 0.3)
    - **Momentum** (e.g., 0.2)
    - **Training epochs** (e.g., 500)
6. Click **Start** → WEKA trains the neural network.
7. Observe **weight and bias updates** in the output.
8. Predict **Result** for new student records.

Result (Sample / Expected):

- Neural Network trained successfully.
- Weights and biases **updated iteratively** to minimize error.
- Predicted Result for new instance (Attendance=65, InternalMarks=70, AssignmentScore=60, SemesterMarks=68) → **Pass**
- Training Accuracy: 100% (for this small dataset)

Conclusion:

- Back Propagation Neural Network successfully predicted student Result.
- Updating weights and biases improves model **learning and accuracy**.
- WEKA simplifies **neural network training and prediction** without coding.

Experiment 10: Customer Segmentation Using k-Means Clustering

---

Aim:

To apply the **k-Means clustering algorithm** on the Customer Segmentation dataset to group customers based on **similar spending behavior**.

Theory:

- **k-Means** is an **unsupervised learning algorithm** used for clustering.
- It partitions data into **k clusters** based on similarity (e.g., distance metric like Euclidean).
- Each cluster has a **centroid** representing the cluster's mean.
- Useful for **market segmentation and customer analysis**.

Dataset (customer_segmentation.arff)

@relation customer_segmentation

@attribute Age numeric

@attribute AnnualIncome numeric

@attribute SpendingScore numeric

@data

25,50000,70

30,60000,60

22,35000,40

28,58000,80

35,45000,50

40,62000,30

32,52000,65

26,48000,75

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **customer_segmentation.arff**.
3. Go to **Cluster tab**.
4. Choose **Clusterer → SimpleKMeans**.
5. Set **number of clusters (k = 3)**.
6. Click **Start** → WEKA performs clustering.
7. Observe **cluster centroids and instance assignments**.
8. Visualize clusters using **Visualize panel**.

Result (Sample / Expected):

- Customers grouped into **3 clusters**:
  - Cluster 0: Young, high spending
  - Cluster 1: Middle-aged, medium spending
  - Cluster 2: Older, low spending
- Centroids show **average Age, AnnualIncome, SpendingScore** per cluster.

Conclusion:

- k-Means successfully grouped customers with **similar spending behavior**.
- Clustering helps in **market segmentation and targeted marketing**.
- WEKA provides an **easy interface for clustering and visualization**.

Experiment 11: Frequent Itemsets and Association Rules Using Apriori Algorithm

---

Aim:

To use the **Apriori algorithm** on the Market Basket dataset to identify **frequent itemsets** and generate **strong association rules**.

Theory:

- **Apriori** is an **unsupervised data mining algorithm** for **association rule mining**.
- Identifies **frequent itemsets** that appear together in transactions.
- Generates **strong association rules** based on **support** and **confidence** thresholds.
- Useful for **market basket analysis** and **sales strategy**.

Dataset (market_basket.arff)

@relation market_basket

@attribute Milk {Yes, No}

@attribute Bread {Yes, No}

@attribute Butter {Yes, No}

@attribute Eggs {Yes, No}

@attribute Jam {Yes, No}

@data

Yes,Yes,No,Yes,No

Yes,No,Yes,No,Yes

No,Yes,Yes,Yes,No

Yes,Yes,Yes,No,No

No,No,Yes,Yes,Yes

Yes,Yes,No,No,Yes

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **market_basket.arff**.
3. Go to **Associate tab**.
4. Choose **Associate → Apriori**.
5. Set parameters:
    - Minimum **support** (e.g., 0.2)
    - Minimum **confidence** (e.g., 0.7)
6. Click **Start** → WEKA finds **frequent itemsets** and **association rules**.
7. Observe the **frequent itemsets** and **generated rules**.

Result (Sample / Expected):

Frequent Itemsets:

- {Milk, Bread}
- {Butter, Eggs}

Strong Association Rules:

1. Milk → Bread (Support: 0.4, Confidence: 0.8)
2. Eggs → Butter (Support: 0.3, Confidence: 0.75)

Conclusion:

- Apriori identifies **items frequently bought together**.
- Strong association rules help in **marketing strategies and product placement**.
- WEKA provides **easy generation and visualization** of frequent patterns and rules.

Experiment 12: Frequent Patterns and Association Rules Using FP-Growth Algorithm

---

Aim:

To apply the **FP-Growth algorithm** on the Retail Transactions dataset to find **frequent patterns** and generate **association rules**.

Theory:

- **FP-Growth** is an **efficient algorithm** for mining **frequent itemsets** without candidate generation.
- Generates **association rules** using **support** and **confidence** thresholds.
- Useful for **market basket analysis** and identifying **co-purchased items**.

Dataset (retail_transactions.arff)

@relation retail_transactions

@attribute Rice {Yes, No}

@attribute Wheat {Yes, No}

@attribute Oil {Yes, No}

@attribute Sugar {Yes, No}

@attribute Salt {Yes, No}

@data

Yes,Yes,No,Yes,No

Yes,No,Yes,No,Yes

No,Yes,Yes,Yes,No

Yes,Yes,Yes,No,No

No,No,Yes,Yes,Yes

Yes,Yes,No,No,Yes

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.

2. Click **Open File** → select **retail_transactions.arff**.
3. Go to **Associate tab**.
4. Choose **Associate → FPGrowth**.
5. Set parameters:
    - Minimum **support** (e.g., 0.2)
    - Minimum **confidence** (e.g., 0.7)
6. Click **Start** → WEKA finds **frequent patterns** and **association rules**.
7. Observe the **patterns** and **generated rules**.

Result (Sample / Expected):

Frequent Patterns:

- {Rice, Wheat}
- {Oil, Sugar}

Strong Association Rules:

1. Rice → Wheat (Support: 0.4, Confidence: 0.85)
2. Sugar → Oil (Support: 0.3, Confidence: 0.75)

Conclusion:

- FP-Growth efficiently finds **frequent patterns** and generates **strong rules**.
- Helps in **market analysis, product placement, and promotions**.
- WEKA provides **fast computation** compared to Apriori for large datasets.

Experiment 13: Performance Comparison of Classifiers on Agricultural Dataset

---

Aim:

To compare the performance of **Decision Tree (J48), k-NN, and Naive Bayes classifiers** on the Agricultural dataset using **accuracy, precision, and recall** metrics.

Theory:

- **Decision Tree (J48):** Supervised classifier using **Information Gain** for splitting.
- **k-NN:** Lazy learning algorithm predicting class based on **nearest neighbors**.
- **Naive Bayes:** Probabilistic classifier assuming **attribute independence**.
- **Performance metrics:**
    - **Accuracy:** Correct predictions / Total predictions
    - **Precision:** Correct positive predictions / Total predicted positive
    - **Recall:** Correct positive predictions / Total actual positive

---

Dataset (agricultural.arff)

@relation agricultural

@attribute Temperature numeric

@attribute Rainfall numeric

@attribute SoilMoisture numeric

@attribute FertilizerUsed numeric

@attribute CropYield {Low, Medium, High}

@data

30,200,60,50,High

25,180,55,45,Medium

28,150,50,40,Medium

35,220,65,60,High

22,100,40,30,Low

27,160,52,45,Medium

...

Procedure (Using WEKA):

1. Open **WEKA → Explorer**.
2. Click **Open File** → select **agricultural.arff**.
3. Go to **Classify tab**.

For each classifier:
4. **Decision Tree (J48):** Classifier → trees → J48 → Start
5. **k-NN:** Classifier → lazy → IBk → Start
6. **Naive Bayes:** Classifier → bayes → NaiveBayes → Start

7. Observe **accuracy, precision, recall, and confusion matrix** for each classifier.
8. Compare metrics to determine **best performing classifier**.

Result (Sample / Expected):

Classifier	Accuracy	Precision	Recall
J48	92%	0.90	0.92
k-NN	88%	0.85	0.87
Naive Bayes	85%	0.82	0.84

- **Best performing classifier:** J48 Decision Tree

Conclusion:

- J48 outperforms k-NN and Naive Bayes on this dataset.
- Performance comparison helps in **selecting the appropriate classifier**.
- WEKA provides **easy computation of multiple metrics** for evaluation

Experiment 14: Installation and Basic Usage of WEKA Tool

Aim:

To install and configure the **WEKA tool**, load a dataset, explore available algorithms, and perform a **simple classification task** to verify successful installation.

Theory:

- **WEKA** is an open-source tool for **data mining and machine learning**.
- Provides **Explorer, Experimenter, KnowledgeFlow** interfaces.
- Supports **classification, clustering, association rules, and pre-processing**.
- Verifying installation ensures the tool is ready for **lab experiments**.

Procedure:

Step 1: Download and Install WEKA

1. Open browser and go to WEKA Download Page.
2. Download the **latest stable version** for your operating system (Windows/Mac/Linux).
3. Run the **installer file** and follow instructions:
    - Accept license agreement
    - Choose installation location
    - Complete installation

Step 2: Launch WEKA

1. Open the installed WEKA application.
2. The **WEKA GUI Chooser** window appears with options: Explorer, Experimenter, KnowledgeFlow.

Step 3: Load a Dataset

1. Click **Explorer → Open File**.
2. Select a dataset (e.g., **iris.arff**).
3. Dataset attributes and instances appear in the Preprocess tab.

Step 4: Explore Algorithms

1. Go to **Classify tab → Classifier**.
2. Browse categories:
    - Trees (e.g., J48)

    - Bayes (e.g., NaiveBayes)
    - Lazy (e.g., k-NN / IBk)

Step 5: Perform Simple Classification

1. Select **J48** classifier.
2. Click **Start**.
3. Observe the **decision tree output**, **accuracy**, and **confusion matrix**.

Step 6: Verify Installation

- Successful dataset load, algorithm execution, and metric display confirm **WEKA is installed and working correctly**.

Result (Sample / Expected):

- Dataset loaded successfully.
- Classifier executed successfully.
- Accuracy example: 97%
- Confusion matrix displayed
- WEKA installation verified.

Conclusion:

- WEKA installed and configured successfully on the system.
- Dataset can be loaded, algorithms explored, and classification performed.
- Tool ready for **data mining experiments and learning tasks**.