CHP-3

04 September 2024 20:43

Decision Trees

Why is tree pruning useful in decision tree induction?

Tree pruning is an essential technique in decision tree induction for several reasons:

1. Reduces Overfitting

- **Overfitting**: A decision tree that is too complex may fit the training data very well but perform poorly on unseen data because it models the noise and minor variations in the training set.
- **Pruning**: By removing branches that have little importance or contribute to overfitting, pruning simplifies the model and helps it generalize better to new data.

2. Improves Generalization

- **Generalization**: A pruned tree often performs better on unseen data because it captures the general trends in the data rather than the specific details.
- **Balancing Act**: Pruning balances the trade-off between model complexity and performance, leading to a model that is both accurate and generalizable.

3. Enhances Interpretability

- **Simpler Models**: Pruned trees are less complex and easier to understand and interpret. A simpler decision tree with fewer branches is often more insightful and can be used more effectively for decision-making.
- **Clarity**: Reducing the number of nodes and branches helps in presenting the decision-making process in a clearer manner.

4. Reduces Computational Costs

- **Efficiency**: A smaller tree with fewer nodes requires less memory and computational power to execute. This makes predictions faster and reduces the cost of deploying the model in a production environment.

5. Prevents Growth of Irrelevant Subtrees

- **Irrelevant Details**: Some branches may be based on very specific cases or noise in the training data. Pruning removes these branches, ensuring that only relevant and significant patterns are retained.

Methods of Tree Pruning

1. **Pre-pruning (Early Stopping)**
   - **Technique**: Stops the tree from growing further based on criteria like maximum depth, minimum number of samples per leaf, or minimum impurity decrease.
   - **Advantage**: Reduces the likelihood of overfitting by preventing the tree from becoming too complex in the first place.
2. **Post-pruning (Cost Complexity Pruning)**
   - **Technique**: Grows the full tree and then prunes it by removing branches that have little impact on the overall accuracy. This is often done using techniques like Reduced Error Pruning or Minimum Description Length (MDL) pruning.
   - **Advantage**: Allows the tree to fully grow and then simplifies it by removing less important branches.

Explain classification and its application

Classification

Definition: Classification is a type of supervised machine learning task where the goal is to assign a category or label to new observations based on a set of input features. It involves training a model on a labeled dataset (where the outcomes are known) so that the model can predict the class of new, unseen data.

How It Works:

1. **Training**: The model is trained on a dataset with known labels. During training, the model learns the relationships between input features and their corresponding labels.
2. **Prediction**: For new data, the model uses the learned relationships to predict the class or category.
3. **Evaluation**: The performance of the model is evaluated using metrics such as accuracy, precision, recall, and F1 score.

Examples of Classification Algorithms:

- **Decision Trees**: Tree-based models that split the data based on feature values to classify the data.
- **Naive Bayes**: A probabilistic model based on Bayes' theorem with strong (naive) independence assumptions between features.
- **Support Vector Machines (SVM)**: A model that finds the hyperplane that best separates different classes in the feature space.
- **K-Nearest Neighbors (KNN)**: A model that classifies data based on the majority class among its nearest neighbors.
- **Neural Networks**: Complex models that can capture non-linear relationships between features and classes.

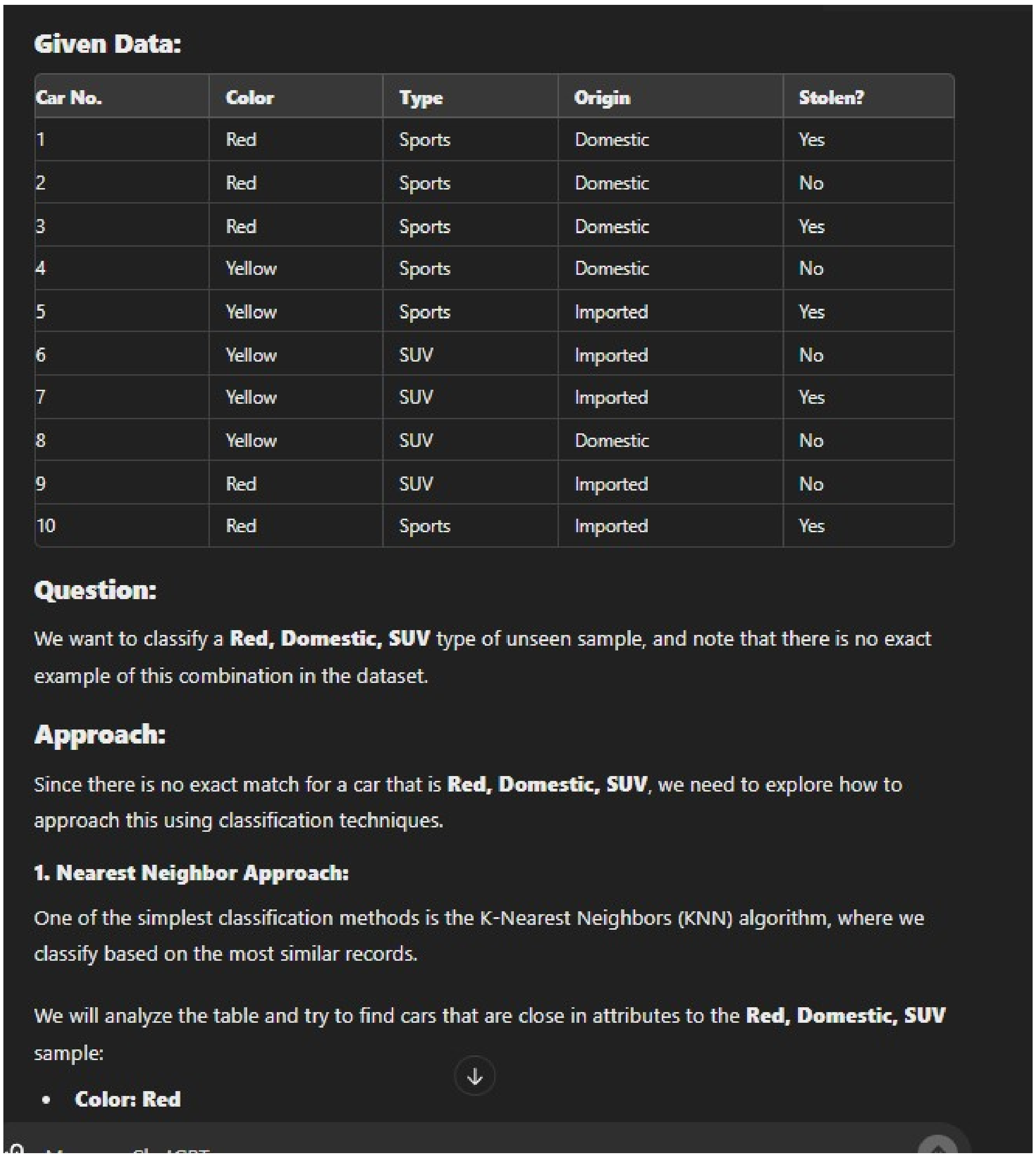
Applications of Classification in DWM (Data Warehouse Management)

In the context of Data Warehouse Management (DWM), classification can be applied in various ways:

1. **Customer Segmentation**:
    - **Application**: Classify customers into different segments based on their purchasing behavior, demographics, or transaction history.
    - **Benefit**: Helps in targeted marketing, personalized recommendations, and improved customer service.
2. **Fraud Detection**:
    - **Application**: Classify transactions or activities as fraudulent or legitimate based on historical data.
    - **Benefit**: Enhances security and reduces financial losses by identifying suspicious activities early.
3. **Sales Forecasting**:
    - **Application**: Classify products or regions into categories based on sales patterns to predict future sales trends.
    - **Benefit**: Assists in inventory management, supply chain optimization, and strategic planning.
4. **Anomaly Detection**:
    - **Application**: Identify unusual patterns or outliers in data that may indicate potential issues or opportunities.
    - **Benefit**: Helps in early detection of problems, such as data quality issues or operational inefficiencies.
5. **Data Quality Management**:
    - **Application**: Classify data records based on their quality, such as completeness, accuracy, or consistency.
    - **Benefit**: Ensures that the data used for analysis and reporting is reliable and valid.
6. **Report Generation**:
    - **Application**: Automatically categorize and generate reports based on different criteria or business needs.
    - **Benefit**: Streamlines the reporting process and ensures that stakeholders receive relevant and timely information.

In summary, classification is a powerful tool in machine learning with numerous applications in Data Warehouse Management, from customer segmentation to fraud detection and beyond. It helps in making informed decisions, improving operational efficiency, and enhancing overall business intelligence.

# Problem based on Naive Bayesian



Given Data:

Car No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Question:

We want to classify a **Red, Domestic, SUV** type of unseen sample, and note that there is no exact example of this combination in the dataset.

Approach:

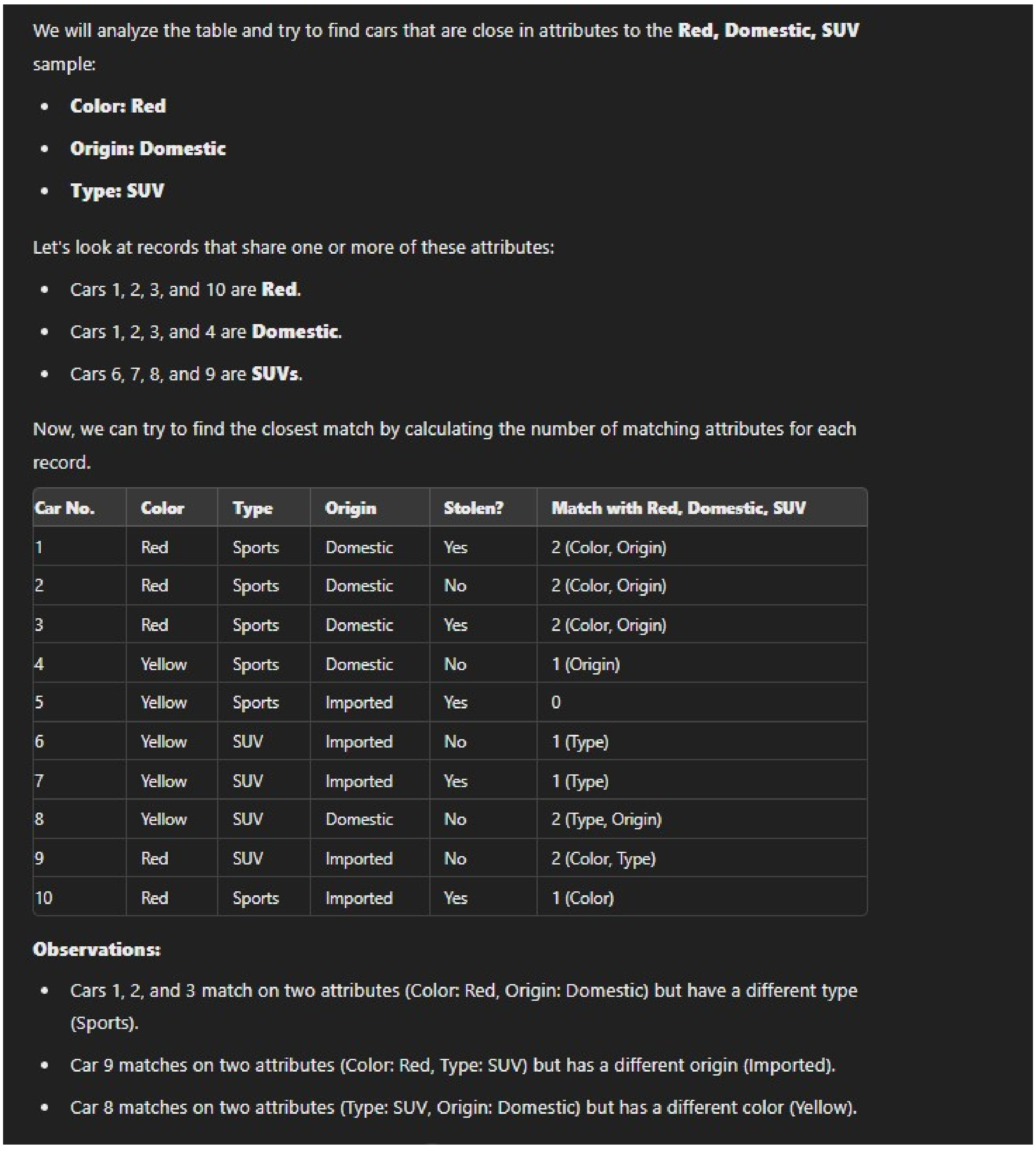
Since there is no exact match for a car that is **Red, Domestic, SUV**, we need to explore how to approach this using classification techniques.

1. Nearest Neighbor Approach:

One of the simplest classification methods is the K-Nearest Neighbors (KNN) algorithm, where we classify based on the most similar records.

We will analyze the table and try to find cars that are close in attributes to the **Red, Domestic, SUV** sample:

- **Color: Red**


We will analyze the table and try to find cars that are close in attributes to the **Red, Domestic, SUV** sample:

- **Color: Red**
- **Origin: Domestic**
- **Type: SUV**

Let's look at records that share one or more of these attributes:

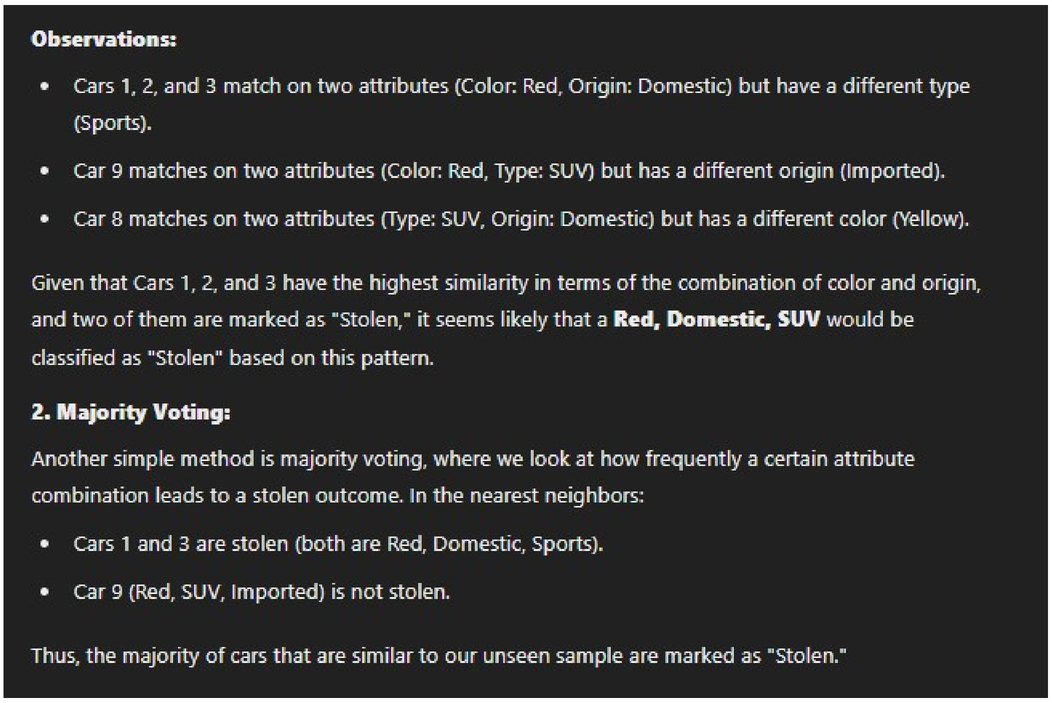
- Cars 1, 2, 3, and 10 are **Red**.
- Cars 1, 2, 3, and 4 are **Domestic**.
- Cars 6, 7, 8, and 9 are **SUVs**.

Now, we can try to find the closest match by calculating the number of matching attributes for each record.

Car No.	Color	Type	Origin	Stolen?	Match with Red, Domestic, SUV
1	Red	Sports	Domestic	Yes	2 (Color, Origin)
2	Red	Sports	Domestic	No	2 (Color, Origin)
3	Red	Sports	Domestic	Yes	2 (Color, Origin)
4	Yellow	Sports	Domestic	No	1 (Origin)
5	Yellow	Sports	Imported	Yes	0
6	Yellow	SUV	Imported	No	1 (Type)
7	Yellow	SUV	Imported	Yes	1 (Type)
8	Yellow	SUV	Domestic	No	2 (Type, Origin)
9	Red	SUV	Imported	No	2 (Color, Type)
10	Red	Sports	Imported	Yes	1 (Color)

Observations:

- Cars 1, 2, and 3 match on two attributes (Color: Red, Origin: Domestic) but have a different type (Sports).
- Car 9 matches on two attributes (Color: Red, Type: SUV) but has a different origin (Imported).
- Car 8 matches on two attributes (Type: SUV, Origin: Domestic) but has a different color (Yellow).


Observations:

- Cars 1, 2, and 3 match on two attributes (Color: Red, Origin: Domestic) but have a different type (Sports).
- Car 9 matches on two attributes (Color: Red, Type: SUV) but has a different origin (Imported).
- Car 8 matches on two attributes (Type: SUV, Origin: Domestic) but has a different color (Yellow).

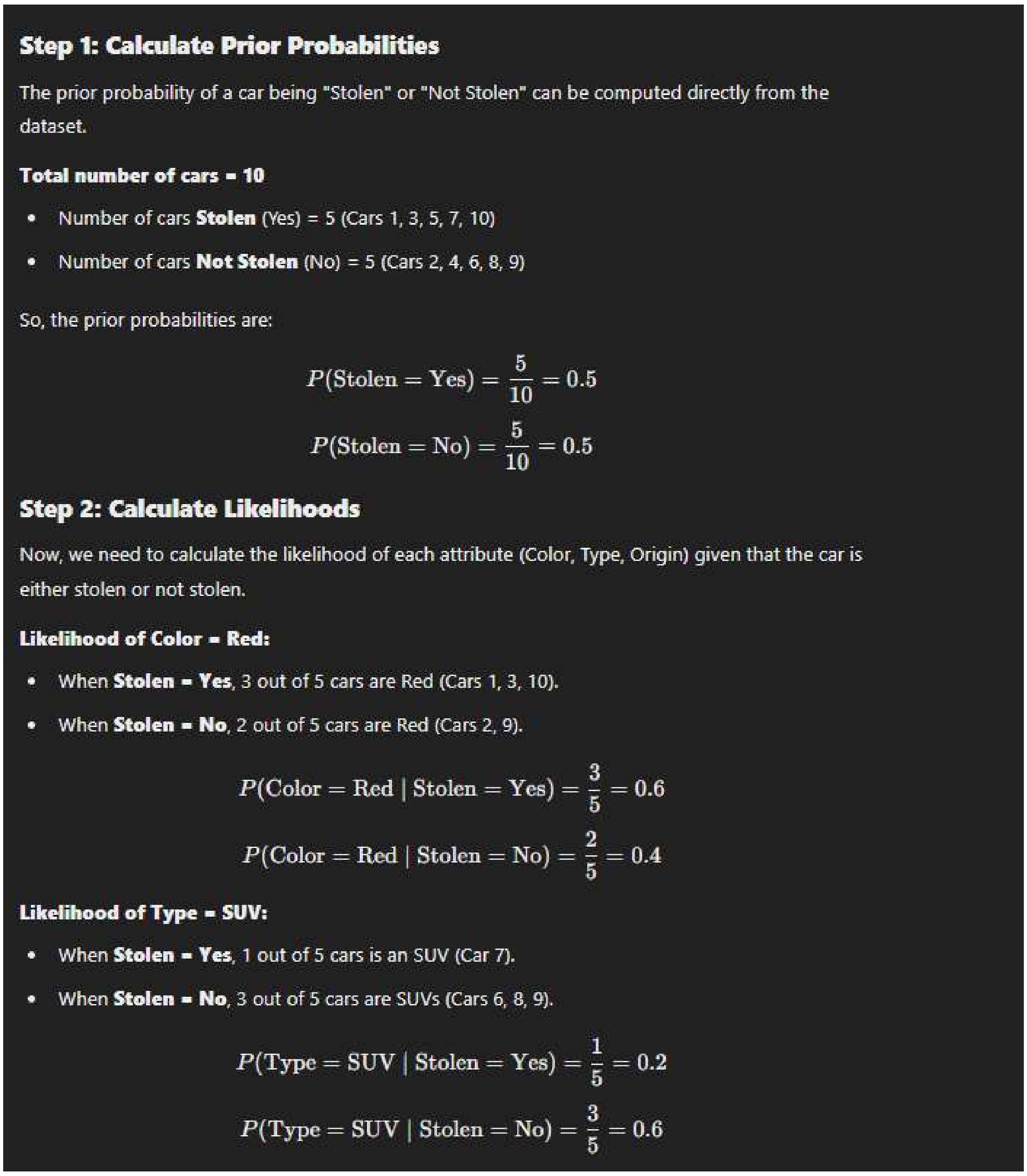
Given that Cars 1, 2, and 3 have the highest similarity in terms of the combination of color and origin, and two of them are marked as "Stolen," it seems likely that a **Red, Domestic, SUV** would be classified as "Stolen" based on this pattern.

2. Majority Voting:

Another simple method is majority voting, where we look at how frequently a certain attribute combination leads to a stolen outcome. In the nearest neighbors:

- Cars 1 and 3 are stolen (both are Red, Domestic, Sports).
- Car 9 (Red, SUV, Imported) is not stolen.

Thus, the majority of cars that are similar to our unseen sample are marked as "Stolen."



Step 1: Calculate Prior Probabilities

The prior probability of a car being "Stolen" or "Not Stolen" can be computed directly from the dataset.

Total number of cars = 10

- Number of cars **Stolen** (Yes) = 5 (Cars 1, 3, 5, 7, 10)
- Number of cars **Not Stolen** (No) = 5 (Cars 2, 4, 6, 8, 9)

So, the prior probabilities are:

$$P(\text{Stolen} = \text{Yes}) = \frac{5}{10} = 0.5$$

$$P(\text{Stolen} = \text{No}) = \frac{5}{10} = 0.5$$

Step 2: Calculate Likelihoods

Now, we need to calculate the likelihood of each attribute (Color, Type, Origin) given that the car is either stolen or not stolen.

Likelihood of Color = Red:

- When **Stolen = Yes**, 3 out of 5 cars are Red (Cars 1, 3, 10).
- When **Stolen = No**, 2 out of 5 cars are Red (Cars 2, 9).

$$P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{Yes}) = \frac{3}{5} = 0.6$$

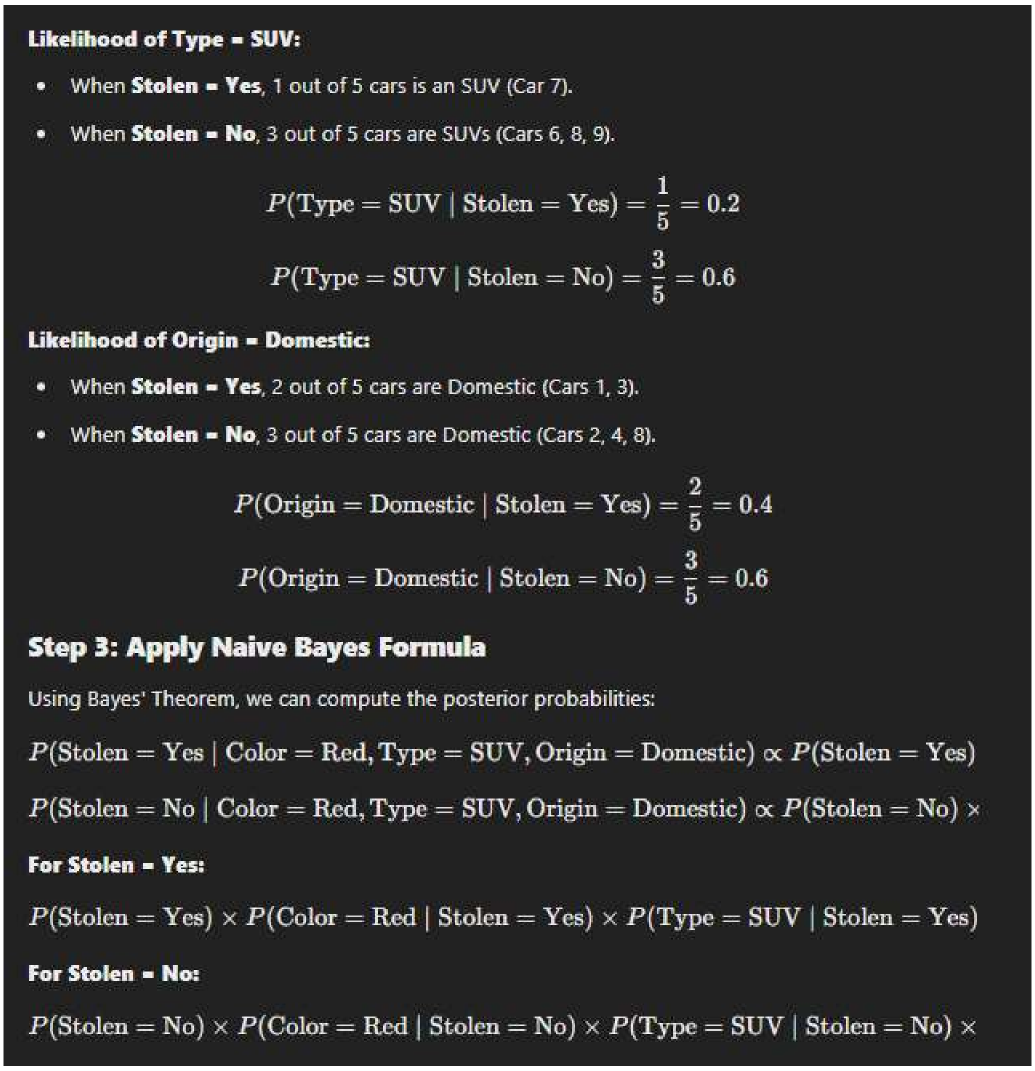
$$P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{No}) = \frac{2}{5} = 0.4$$

Likelihood of Type = SUV:

- When **Stolen = Yes**, 1 out of 5 cars is an SUV (Car 7).
- When **Stolen = No**, 3 out of 5 cars are SUVs (Cars 6, 8, 9).

$$P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{Yes}) = \frac{1}{5} = 0.2$$

$$P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{No}) = \frac{3}{5} = 0.6$$



Likelihood of Type = SUV:

- When **Stolen = Yes**, 1 out of 5 cars is an SUV (Car 7).
- When **Stolen = No**, 3 out of 5 cars are SUVs (Cars 6, 8, 9).

$$P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{Yes}) = \frac{1}{5} = 0.2$$

$$P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{No}) = \frac{3}{5} = 0.6$$

Likelihood of Origin = Domestic:

- When **Stolen = Yes**, 2 out of 5 cars are Domestic (Cars 1, 3).
- When **Stolen = No**, 3 out of 5 cars are Domestic (Cars 2, 4, 8).

$$P(\text{Origin} = \text{Domestic} \mid \text{Stolen} = \text{Yes}) = \frac{2}{5} = 0.4$$

$$P(\text{Origin} = \text{Domestic} \mid \text{Stolen} = \text{No}) = \frac{3}{5} = 0.6$$

Step 3: Apply Naive Bayes Formula

Using Bayes' Theorem, we can compute the posterior probabilities:

$P(\text{Stolen} = \text{Yes} \mid \text{Color} = \text{Red}, \text{Type} = \text{SUV}, \text{Origin} = \text{Domestic}) \propto P(\text{Stolen} = \text{Yes})$

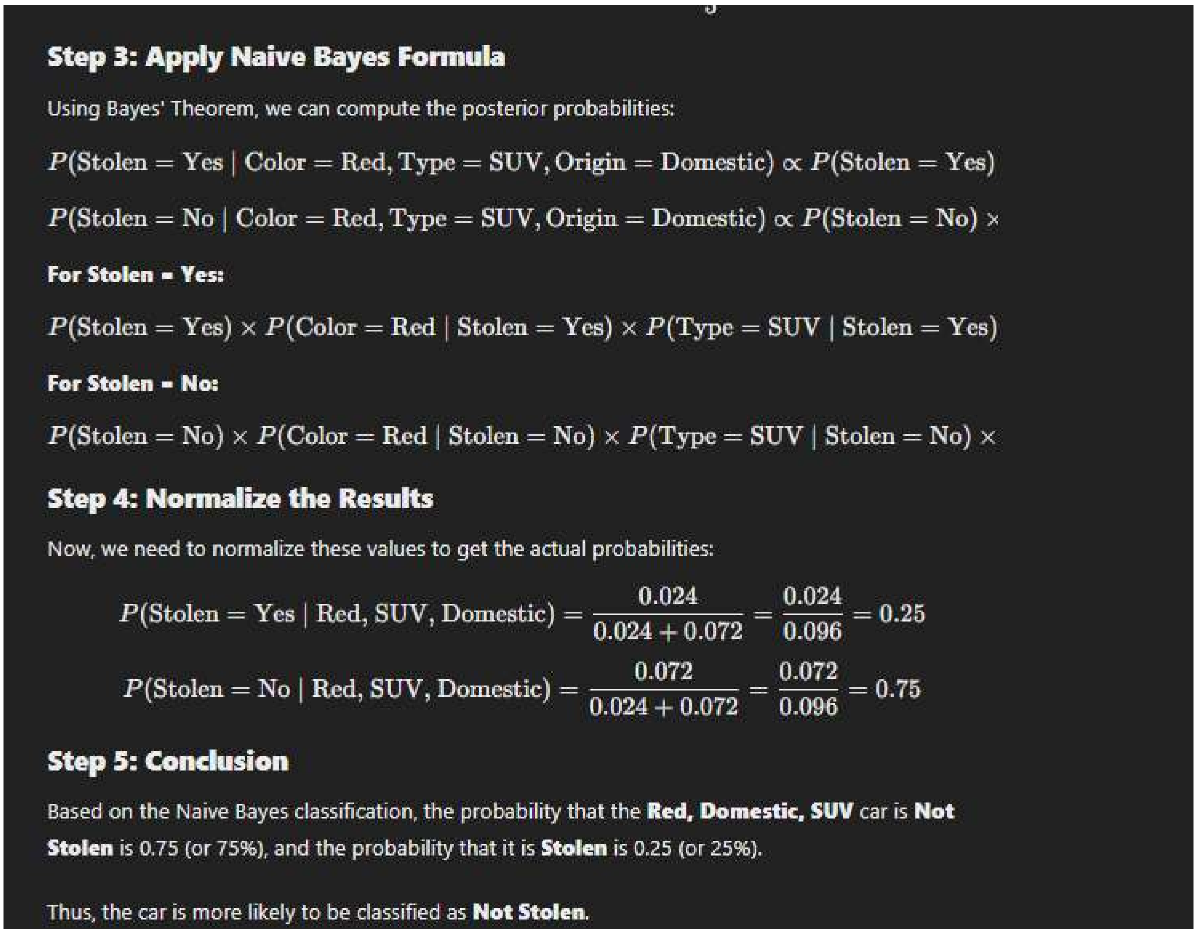
$P(\text{Stolen} = \text{No} \mid \text{Color} = \text{Red}, \text{Type} = \text{SUV}, \text{Origin} = \text{Domestic}) \propto P(\text{Stolen} = \text{No}) \times$

For Stolen = Yes:

$P(\text{Stolen} = \text{Yes}) \times P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{Yes}) \times P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{Yes})$

For Stolen = No:

$P(\text{Stolen} = \text{No}) \times P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{No}) \times P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{No}) \times$



Step 3: Apply Naive Bayes Formula

Using Bayes' Theorem, we can compute the posterior probabilities:

$P(\text{Stolen} = \text{Yes} \mid \text{Color} = \text{Red}, \text{Type} = \text{SUV}, \text{Origin} = \text{Domestic}) \propto P(\text{Stolen} = \text{Yes})$

$P(\text{Stolen} = \text{No} \mid \text{Color} = \text{Red}, \text{Type} = \text{SUV}, \text{Origin} = \text{Domestic}) \propto P(\text{Stolen} = \text{No}) \times$

For Stolen = Yes:

$P(\text{Stolen} = \text{Yes}) \times P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{Yes}) \times P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{Yes})$

For Stolen = No:

$P(\text{Stolen} = \text{No}) \times P(\text{Color} = \text{Red} \mid \text{Stolen} = \text{No}) \times P(\text{Type} = \text{SUV} \mid \text{Stolen} = \text{No}) \times$

Step 4: Normalize the Results

Now, we need to normalize these values to get the actual probabilities:

$$P(\text{Stolen} = \text{Yes} \mid \text{Red, SUV, Domestic}) = \frac{0.024}{0.024 + 0.072} = \frac{0.024}{0.096} = 0.25$$

$$P(\text{Stolen} = \text{No} \mid \text{Red, SUV, Domestic}) = \frac{0.072}{0.024 + 0.072} = \frac{0.072}{0.096} = 0.75$$

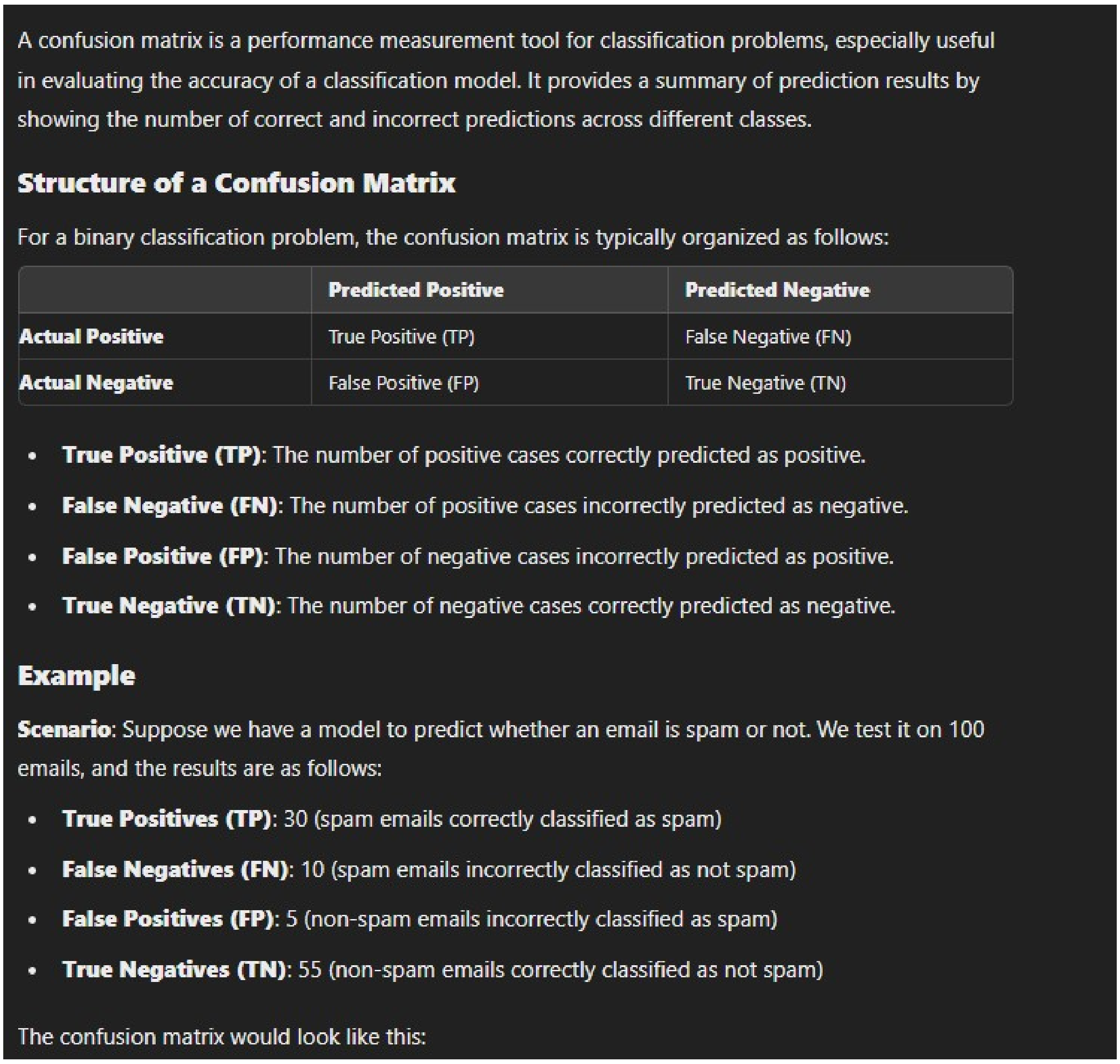
Step 5: Conclusion

Based on the Naive Bayes classification, the probability that the **Red, Domestic, SUV** car is **Not Stolen** is 0.75 (or 75%), and the probability that it is **Stolen** is 0.25 (or 25%).

Thus, the car is more likely to be classified as **Not Stolen**.

Problem based on Decision tree

Confusion Matrix



A confusion matrix is a performance measurement tool for classification problems, especially useful in evaluating the accuracy of a classification model. It provides a summary of prediction results by showing the number of correct and incorrect predictions across different classes.

Structure of a Confusion Matrix

For a binary classification problem, the confusion matrix is typically organized as follows:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

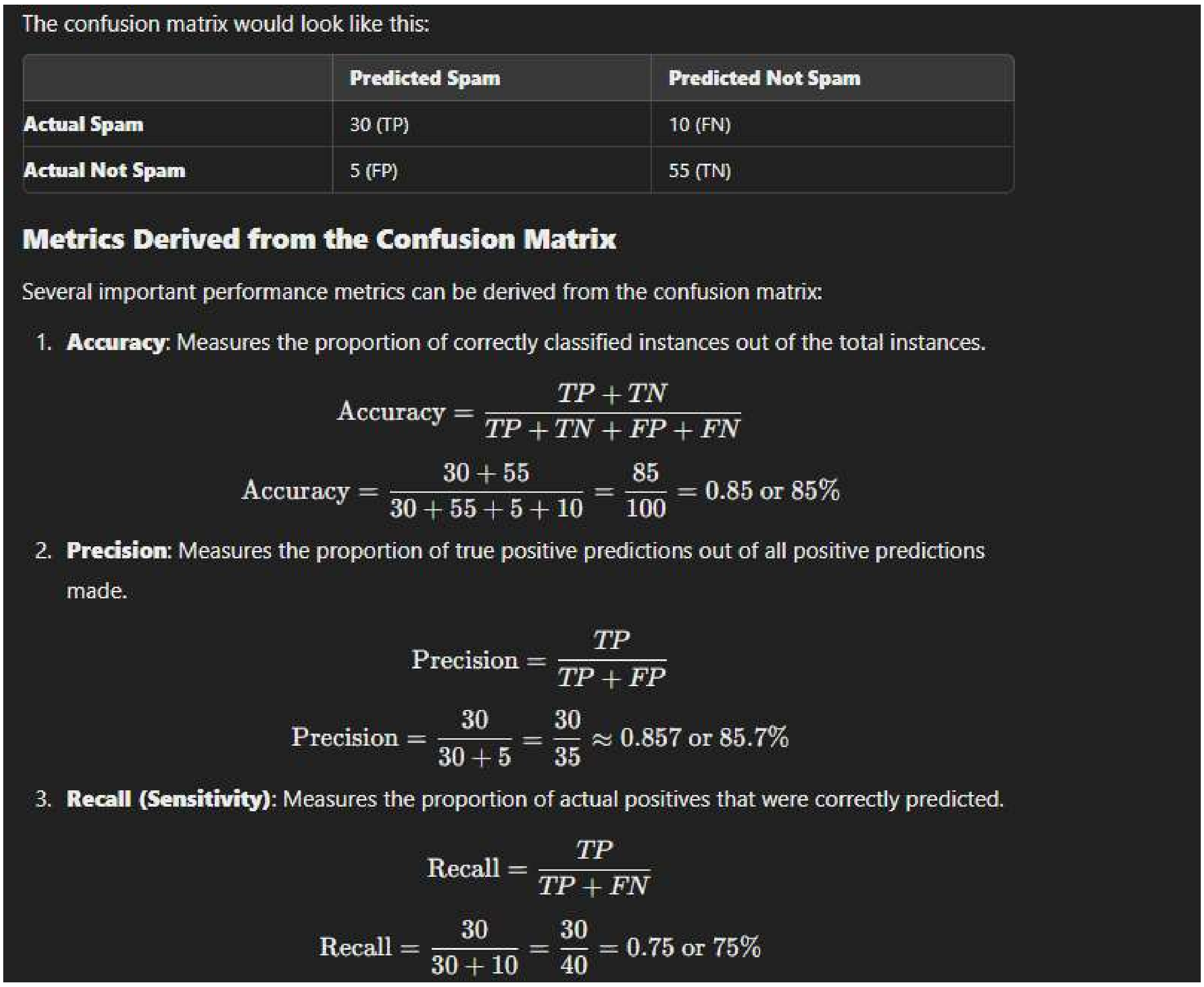
- **True Positive (TP)**: The number of positive cases correctly predicted as positive.
- **False Negative (FN)**: The number of positive cases incorrectly predicted as negative.
- **False Positive (FP)**: The number of negative cases incorrectly predicted as positive.
- **True Negative (TN)**: The number of negative cases correctly predicted as negative.

Example

Scenario: Suppose we have a model to predict whether an email is spam or not. We test it on 100 emails, and the results are as follows:

- **True Positives (TP)**: 30 (spam emails correctly classified as spam)
- **False Negatives (FN)**: 10 (spam emails incorrectly classified as not spam)
- **False Positives (FP)**: 5 (non-spam emails incorrectly classified as spam)
- **True Negatives (TN)**: 55 (non-spam emails correctly classified as not spam)

The confusion matrix would look like this:



The confusion matrix would look like this:

	Predicted Spam	Predicted Not Spam
Actual Spam	30 (TP)	10 (FN)
Actual Not Spam	5 (FP)	55 (TN)

Metrics Derived from the Confusion Matrix

Several important performance metrics can be derived from the confusion matrix:

1. **Accuracy**: Measures the proportion of correctly classified instances out of the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Accuracy} = \frac{30 + 55}{30 + 55 + 5 + 10} = \frac{85}{100} = 0.85 \text{ or } 85\%$$

2. **Precision**: Measures the proportion of true positive predictions out of all positive predictions made.

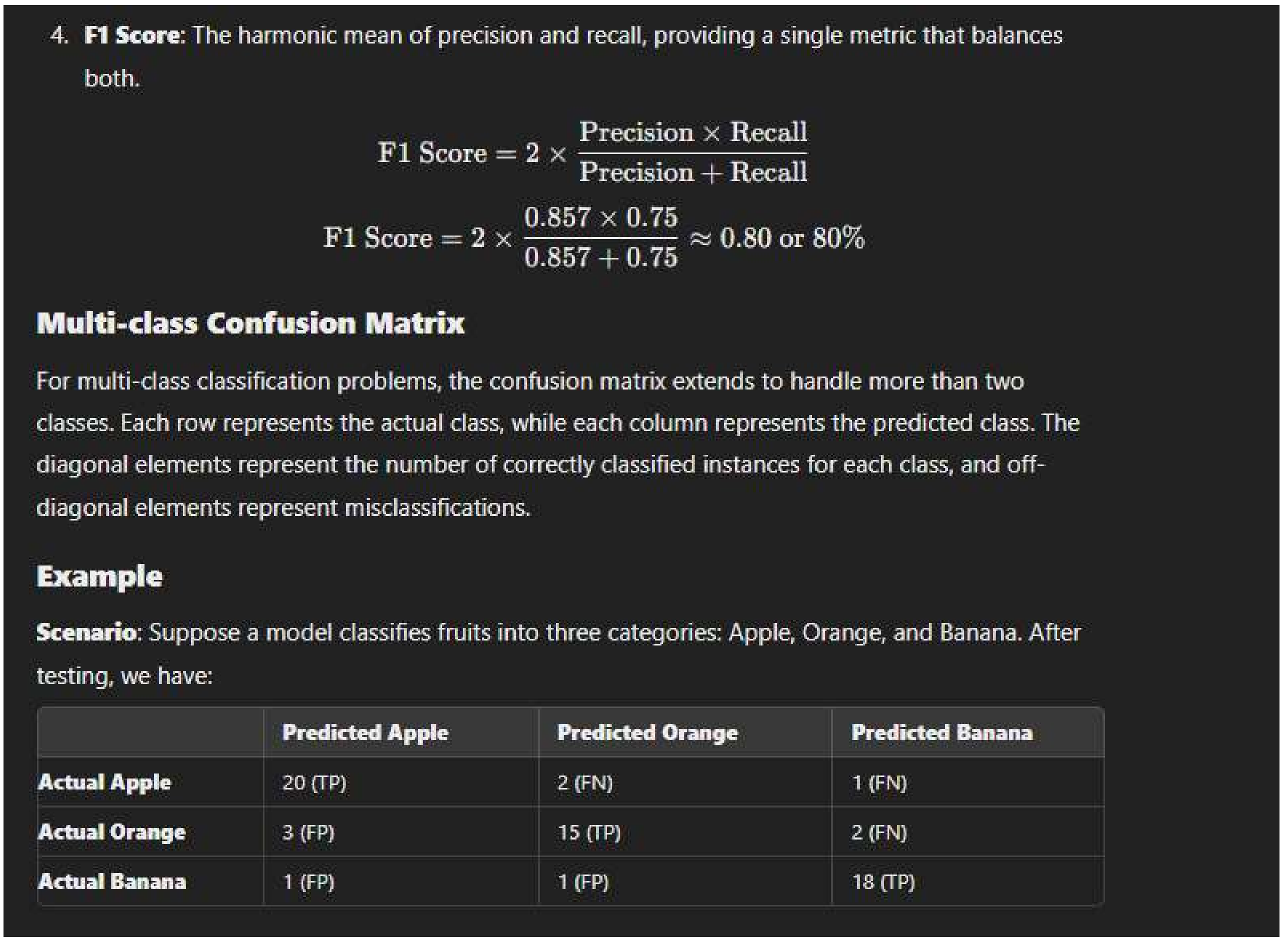
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{30}{30 + 5} = \frac{30}{35} \approx 0.857 \text{ or } 85.7\%$$

3. **Recall (Sensitivity)**: Measures the proportion of actual positives that were correctly predicted.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{30}{30 + 10} = \frac{30}{40} = 0.75 \text{ or } 75\%$$



4. **F1 Score**: The harmonic mean of precision and recall, providing a single metric that balances both.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{F1 Score} = 2 \times \frac{0.857 \times 0.75}{0.857 + 0.75} \approx 0.80 \text{ or } 80\%$$

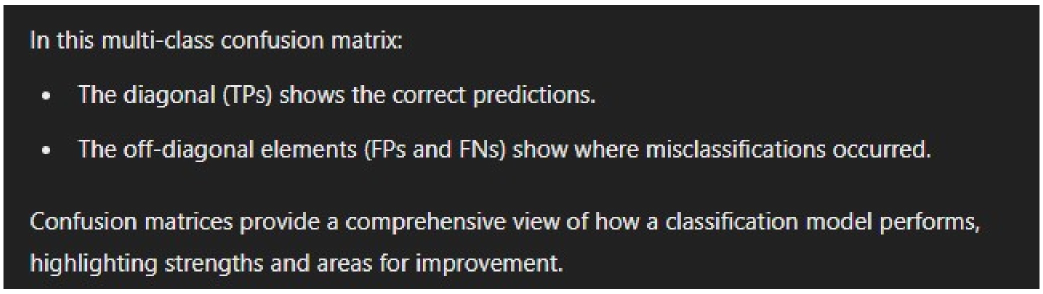
Multi-class Confusion Matrix

For multi-class classification problems, the confusion matrix extends to handle more than two classes. Each row represents the actual class, while each column represents the predicted class. The diagonal elements represent the number of correctly classified instances for each class, and off-diagonal elements represent misclassifications.

Example

Scenario: Suppose a model classifies fruits into three categories: Apple, Orange, and Banana. After testing, we have:

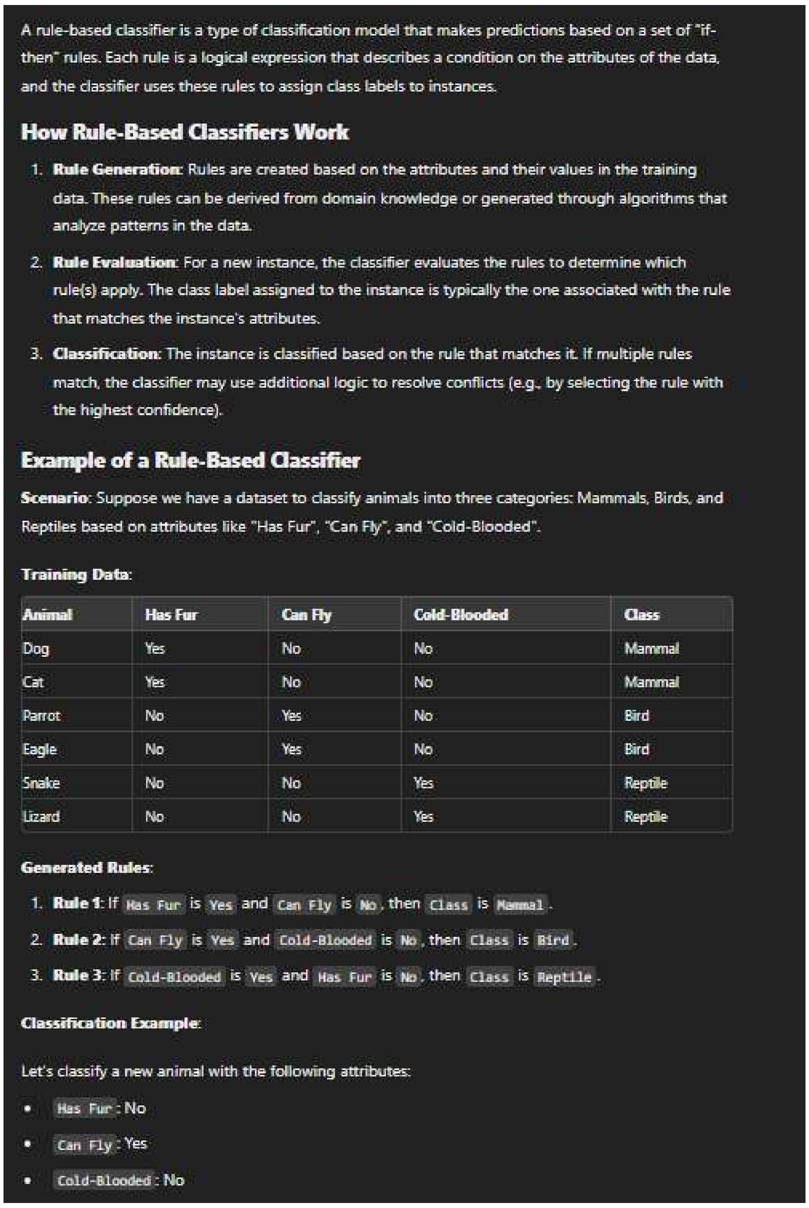
	Predicted Apple	Predicted Orange	Predicted Banana
Actual Apple	20 (TP)	2 (FN)	1 (FN)
Actual Orange	3 (FP)	15 (TP)	2 (FN)
Actual Banana	1 (FP)	1 (FP)	18 (TP)


In this multi-class confusion matrix:

- The diagonal (TPs) shows the correct predictions.
- The off-diagonal elements (FPs and FNs) show where misclassifications occurred.

Confusion matrices provide a comprehensive view of how a classification model performs, highlighting strengths and areas for improvement.

Explain Rule based classifier with an example


A rule-based classifier is a type of classification model that makes predictions based on a set of "if-then" rules. Each rule is a logical expression that describes a condition on the attributes of the data, and the classifier uses these rules to assign class labels to instances.

How Rule-Based Classifiers Work

1. **Rule Generation**: Rules are created based on the attributes and their values in the training data. These rules can be derived from domain knowledge or generated through algorithms that analyze patterns in the data.
2. **Rule Evaluation**: For a new instance, the classifier evaluates the rules to determine which rule(s) apply. The class label assigned to the instance is typically the one associated with the rule that matches the instance's attributes.
3. **Classification**: The instance is classified based on the rule that matches it. If multiple rules match, the classifier may use additional logic to resolve conflicts (e.g., by selecting the rule with the highest confidence).

Example of a Rule-Based Classifier

Scenario: Suppose we have a dataset to classify animals into three categories: Mammals, Birds, and Reptiles based on attributes like "Has Fur", "Can Fly", and "Cold-Blooded".

Training Data:

Animal	Has Fur	Can Fly	Cold-Blooded	Class
Dog	Yes	No	No	Mammal
Cat	Yes	No	No	Mammal
Parrot	No	Yes	No	Bird
Eagle	No	Yes	No	Bird
Snake	No	No	Yes	Reptile
Lizard	No	No	Yes	Reptile

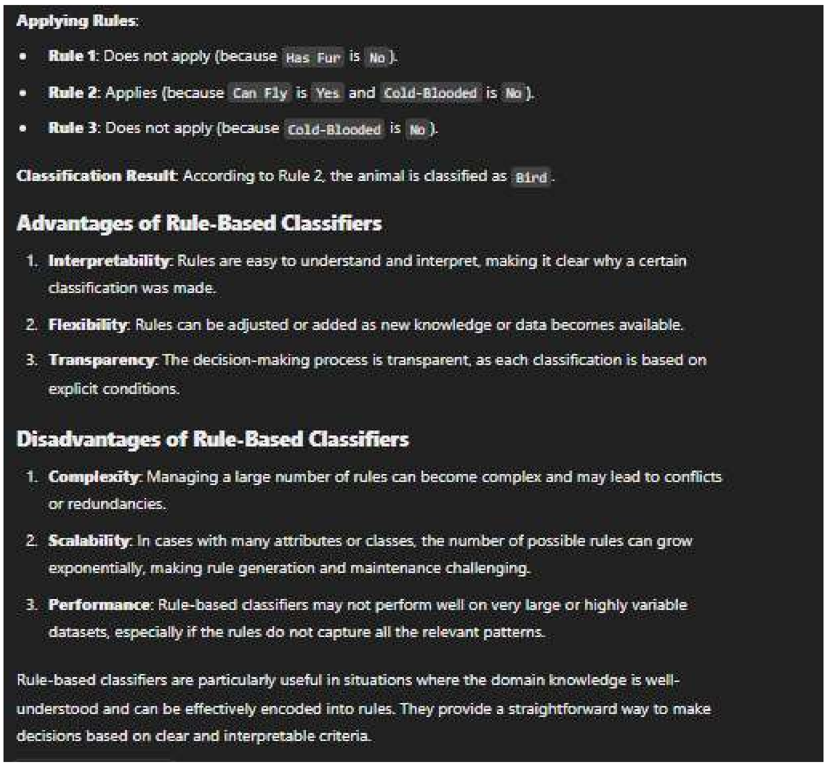
Generated Rules:

1. **Rule 1**: If `Has Fur` is `Yes` and `Can Fly` is `No`, then `Class` is `Mammal`.
2. **Rule 2**: If `Can Fly` is `Yes` and `Cold-Blooded` is `No`, then `Class` is `Bird`.
3. **Rule 3**: If `Cold-Blooded` is `Yes` and `Has Fur` is `No`, then `Class` is `Reptile`.

Classification Example:

Let's classify a new animal with the following attributes:

- `Has Fur`: No
- `Can Fly`: Yes
- `Cold-Blooded`: No


Applying Rules:

- **Rule 1**: Does not apply (because `Has Fur` is `No`).
- **Rule 2**: Applies (because `Can Fly` is `Yes` and `Cold-Blooded` is `No`).
- **Rule 3**: Does not apply (because `Cold-Blooded` is `No`).

Classification Result: According to Rule 2, the animal is classified as `Bird`.

Advantages of Rule-Based Classifiers

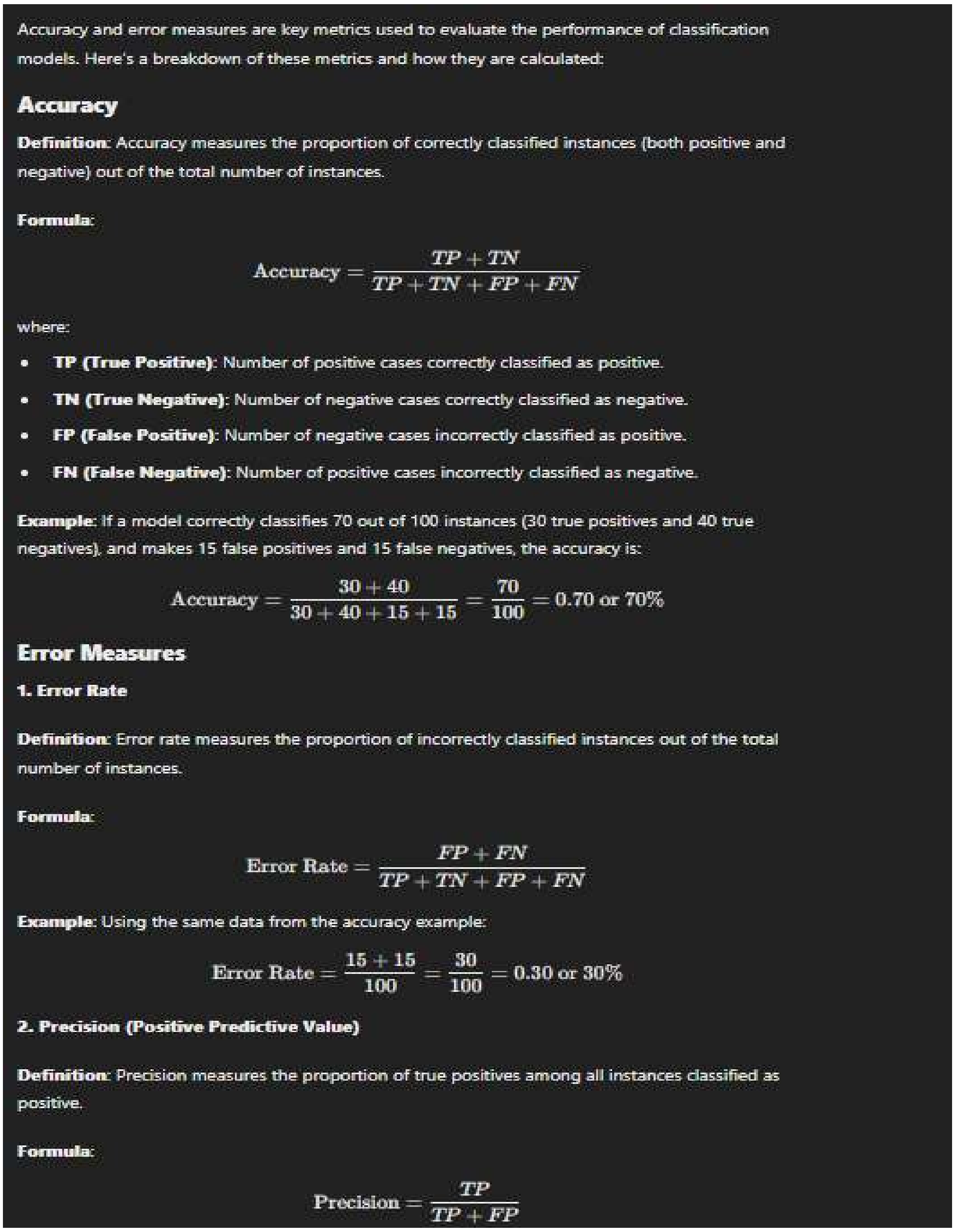
1. **Interpretability**: Rules are easy to understand and interpret, making it clear why a certain classification was made.
2. **Flexibility**: Rules can be adjusted or added as new knowledge or data becomes available.
3. **Transparency**: The decision-making process is transparent, as each classification is based on explicit conditions.

Disadvantages of Rule-Based Classifiers

1. **Complexity**: Managing a large number of rules can become complex and may lead to conflicts or redundancies.
2. **Scalability**: In cases with many attributes or classes, the number of possible rules can grow exponentially, making rule generation and maintenance challenging.
3. **Performance**: Rule-based classifiers may not perform well on very large or highly variable datasets, especially if the rules do not capture all the relevant patterns.

Rule-based classifiers are particularly useful in situations where the domain knowledge is well-understood and can be effectively encoded into rules. They provide a straightforward way to make decisions based on clear and interpretable criteria.

Accuracy and error measures



Accuracy and error measures are key metrics used to evaluate the performance of classification models. Here's a breakdown of these metrics and how they are calculated:

Accuracy

Definition: Accuracy measures the proportion of correctly classified instances (both positive and negative) out of the total number of instances.

Formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- **TP (True Positive):** Number of positive cases correctly classified as positive.
- **TN (True Negative):** Number of negative cases correctly classified as negative.
- **FP (False Positive):** Number of negative cases incorrectly classified as positive.
- **FN (False Negative):** Number of positive cases incorrectly classified as negative.

Example: If a model correctly classifies 70 out of 100 instances (30 true positives and 40 true negatives), and makes 15 false positives and 15 false negatives, the accuracy is:

$$\text{Accuracy} = \frac{30 + 40}{30 + 40 + 15 + 15} = \frac{70}{100} = 0.70 \text{ or } 70\%$$

Error Measures

1. Error Rate

Definition: Error rate measures the proportion of incorrectly classified instances out of the total number of instances.

Formula:

$$\text{Error Rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

Example: Using the same data from the accuracy example:

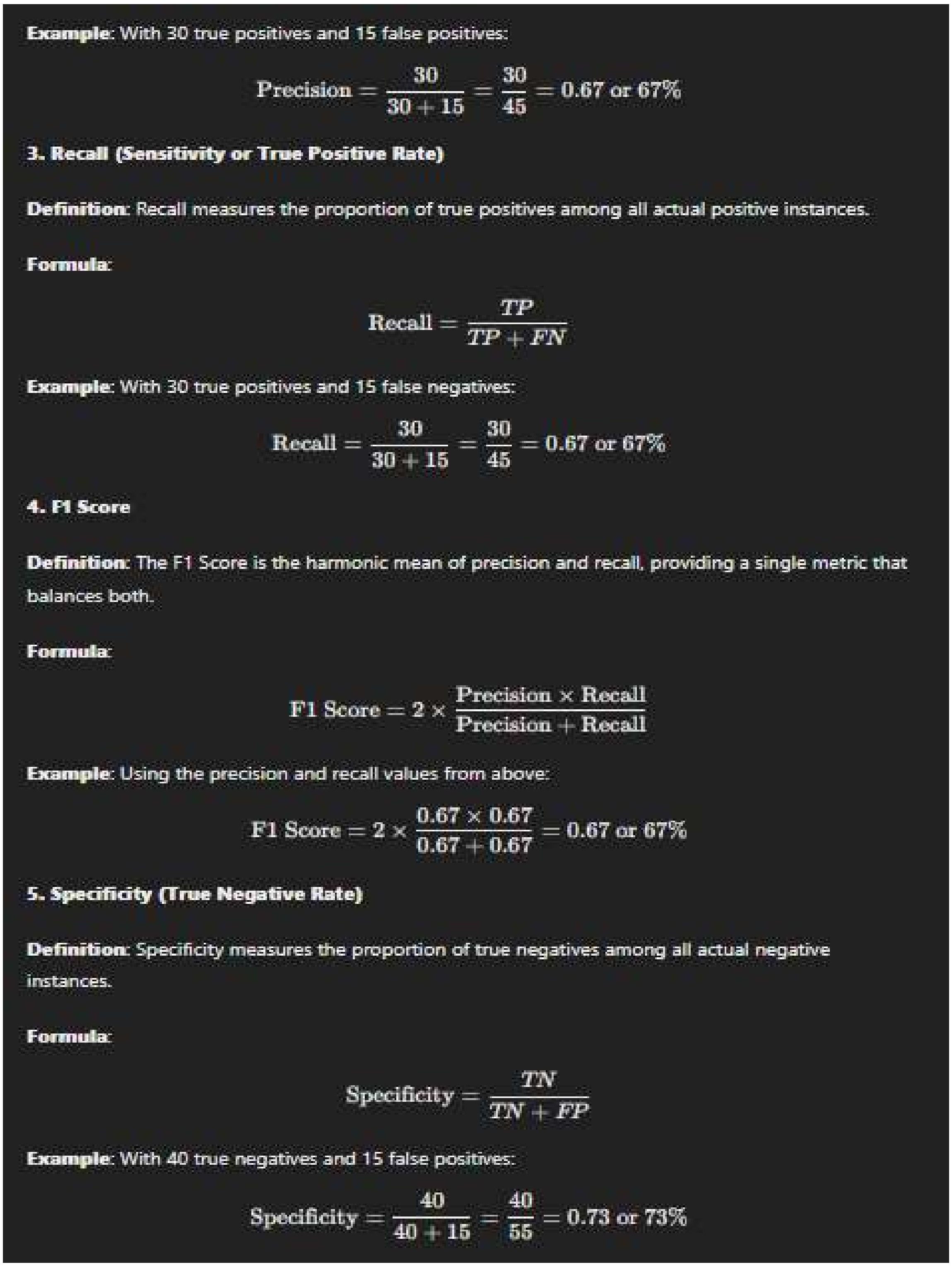
$$\text{Error Rate} = \frac{15 + 15}{100} = \frac{30}{100} = 0.30 \text{ or } 30\%$$

2. Precision (Positive Predictive Value)

Definition: Precision measures the proportion of true positives among all instances classified as positive.

Formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$



Example: With 30 true positives and 15 false positives:

$$\text{Precision} = \frac{30}{30+15} = \frac{30}{45} = 0.67 \text{ or } 67\%$$

3. Recall (Sensitivity or True Positive Rate)

Definition: Recall measures the proportion of true positives among all actual positive instances.

Formula:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Example: With 30 true positives and 15 false negatives:

$$\text{Recall} = \frac{30}{30+15} = \frac{30}{45} = 0.67 \text{ or } 67\%$$

4. F1 Score

Definition: The F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both.

Formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Example: Using the precision and recall values from above:

$$\text{F1 Score} = 2 \times \frac{0.67 \times 0.67}{0.67 + 0.67} = 0.67 \text{ or } 67\%$$

5. Specificity (True Negative Rate)

Definition: Specificity measures the proportion of true negatives among all actual negative instances.

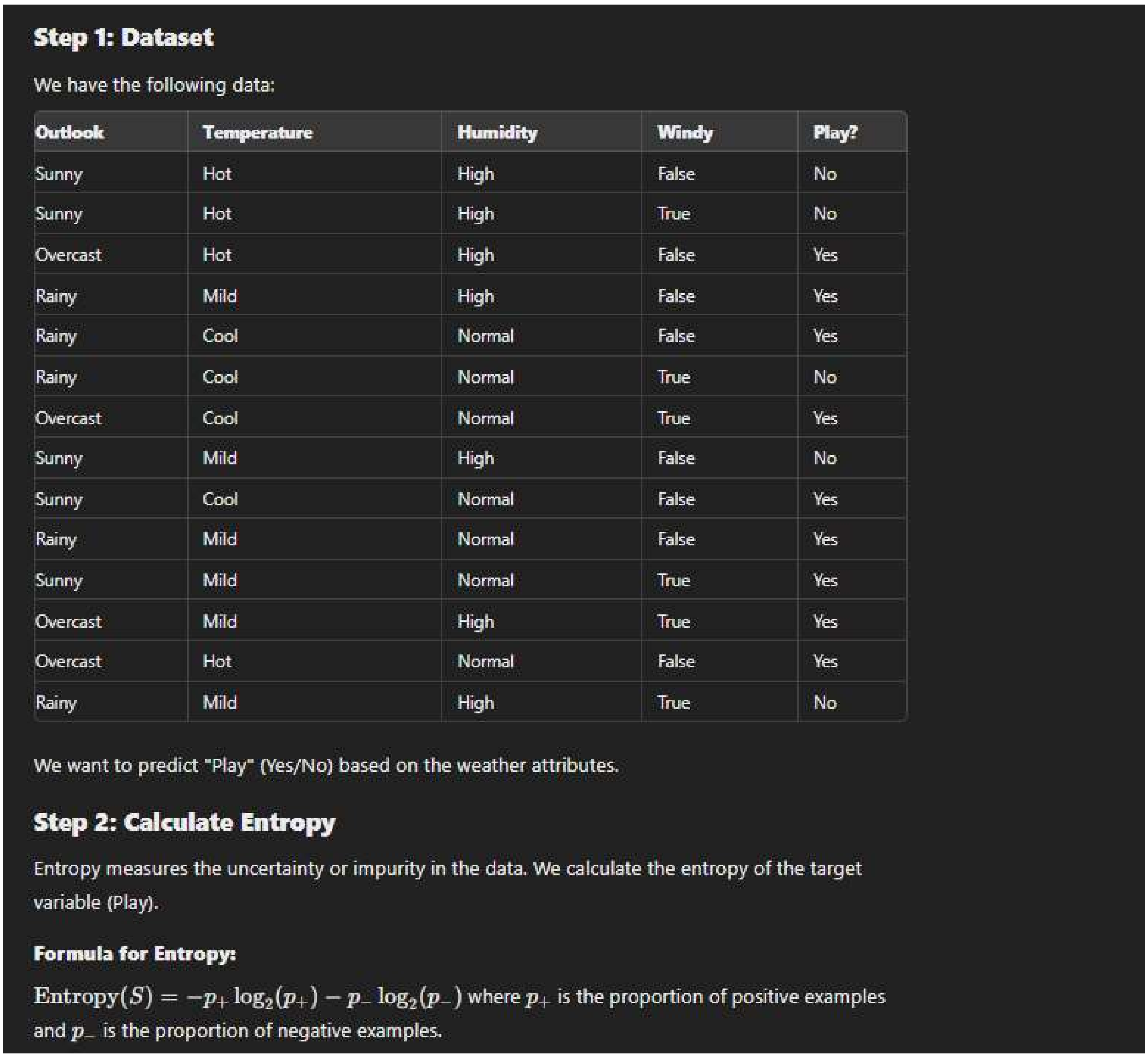
Formula:

$$\text{Specificity} = \frac{TN}{TN+FP}$$

Example: With 40 true negatives and 15 false positives:

$$\text{Specificity} = \frac{40}{40+15} = \frac{40}{55} = 0.73 \text{ or } 73\%$$

Qn:



Step 1: Dataset

We have the following data:

Outlook	Temperature	Humidity	Windy	Play?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

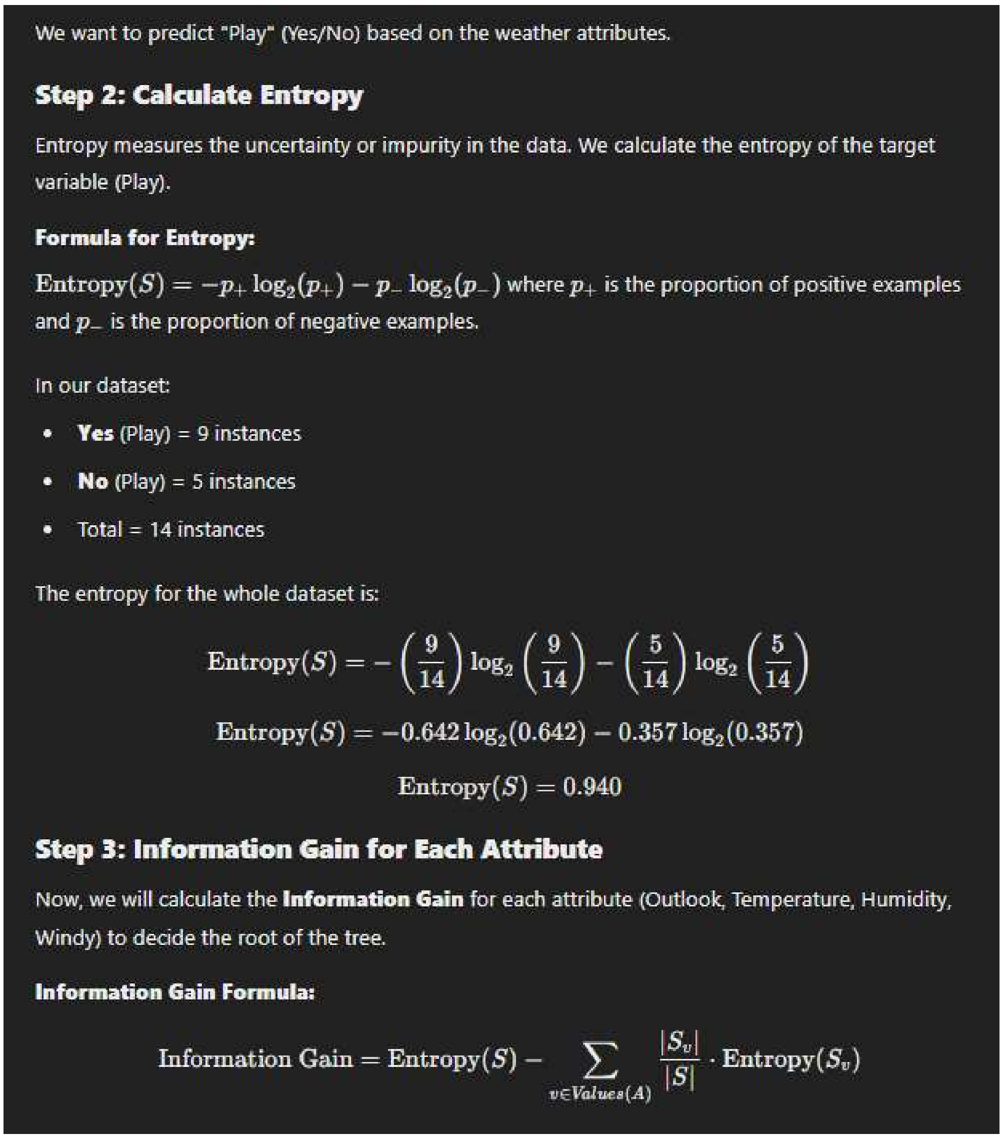
We want to predict "Play" (Yes/No) based on the weather attributes.

Step 2: Calculate Entropy

Entropy measures the uncertainty or impurity in the data. We calculate the entropy of the target variable (Play).

Formula for Entropy:

$\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$ where p_+ is the proportion of positive examples and p_- is the proportion of negative examples.



We want to predict "Play" (Yes/No) based on the weather attributes.

Step 2: Calculate Entropy

Entropy measures the uncertainty or impurity in the data. We calculate the entropy of the target variable (Play).

Formula for Entropy:

$\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$ where p_+ is the proportion of positive examples and p_- is the proportion of negative examples.

In our dataset:

- **Yes** (Play) = 9 instances
- **No** (Play) = 5 instances
- Total = 14 instances

The entropy for the whole dataset is:

$$\text{Entropy}(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$\text{Entropy}(S) = -0.642\log_2(0.642) - 0.357\log_2(0.357)$$

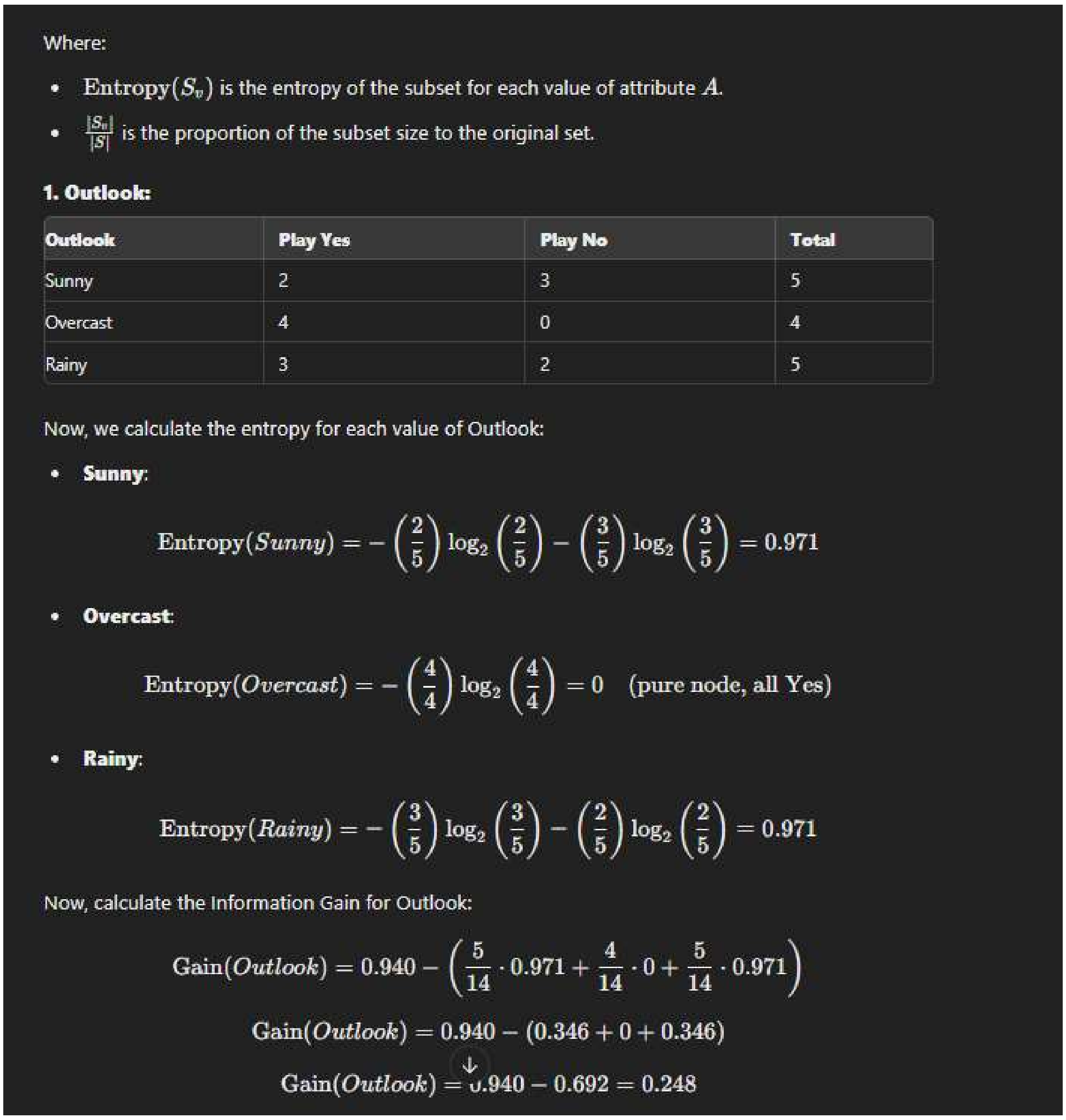
$$\text{Entropy}(S) = 0.940$$

Step 3: Information Gain for Each Attribute

Now, we will calculate the **Information Gain** for each attribute (Outlook, Temperature, Humidity, Windy) to decide the root of the tree.

Information Gain Formula:

$$\text{Information Gain} = \text{Entropy}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$


Where:

- $\text{Entropy}(S_v)$ is the entropy of the subset for each value of attribute A.
- $\frac{|S_v|}{|S|}$ is the proportion of the subset size to the original set.

1. Outlook:

Outlook	Play Yes	Play No	Total
Sunny	2	3	5
Overcast	4	0	4
Rainy	3	2	5

Now, we calculate the entropy for each value of Outlook:

- **Sunny**:

$$\text{Entropy}(Sunny) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

- **Overcast**:

$$\text{Entropy}(Overcast) = -\left(\frac{4}{4}\right)\log_2\left(\frac{4}{4}\right) = 0 \quad \text{(pure node, all Yes)}$$

- **Rainy**:

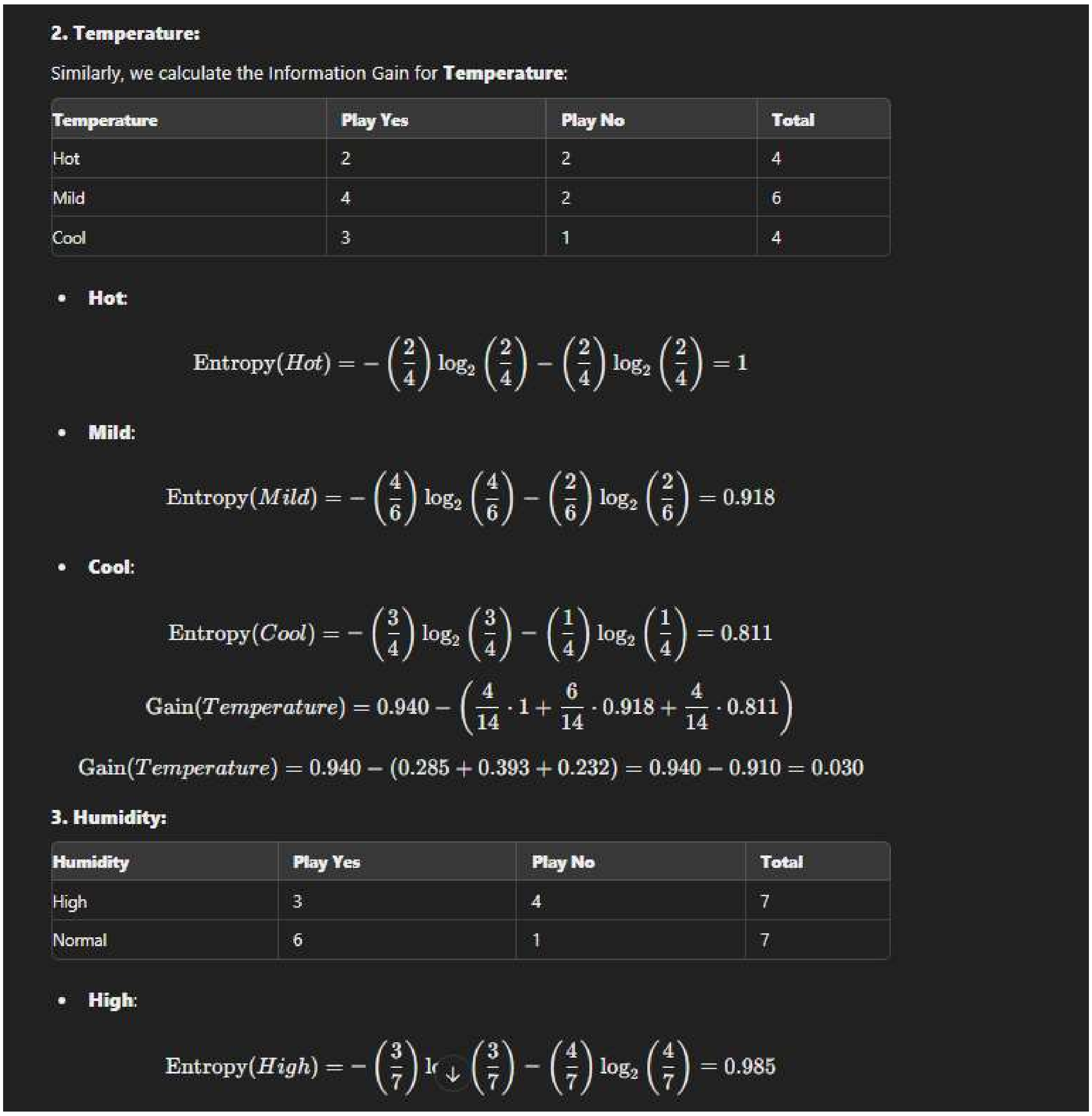
$$\text{Entropy}(Rainy) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.971$$

Now, calculate the Information Gain for Outlook:

$$\text{Gain}(Outlook) = 0.940 - \left(\frac{5}{14}\cdot 0.971 + \frac{4}{14}\cdot 0 + \frac{5}{14}\cdot 0.971\right)$$

$$\text{Gain}(Outlook) = 0.940 - (0.346 + 0 + 0.346)$$

$$\text{Gain}(Outlook) = 0.940 - 0.692 = 0.248$$



2. Temperature:

Similarly, we calculate the Information Gain for **Temperature**:

Temperature	Play Yes	Play No	Total
Hot	2	2	4
Mild	4	2	6
Cool	3	1	4

- **Hot**:

$$\text{Entropy}(Hot) = -\left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) = 1$$

- **Mild**:

$$\text{Entropy}(Mild) = -\left(\frac{4}{6}\right)\log_2\left(\frac{4}{6}\right) - \left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right) = 0.918$$

- **Cool**:

$$\text{Entropy}(Cool) = -\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) = 0.811$$

$$\text{Gain}(Temperature) = 0.940 - \left(\frac{4}{14}\cdot 1 + \frac{6}{14}\cdot 0.918 + \frac{4}{14}\cdot 0.811\right)$$

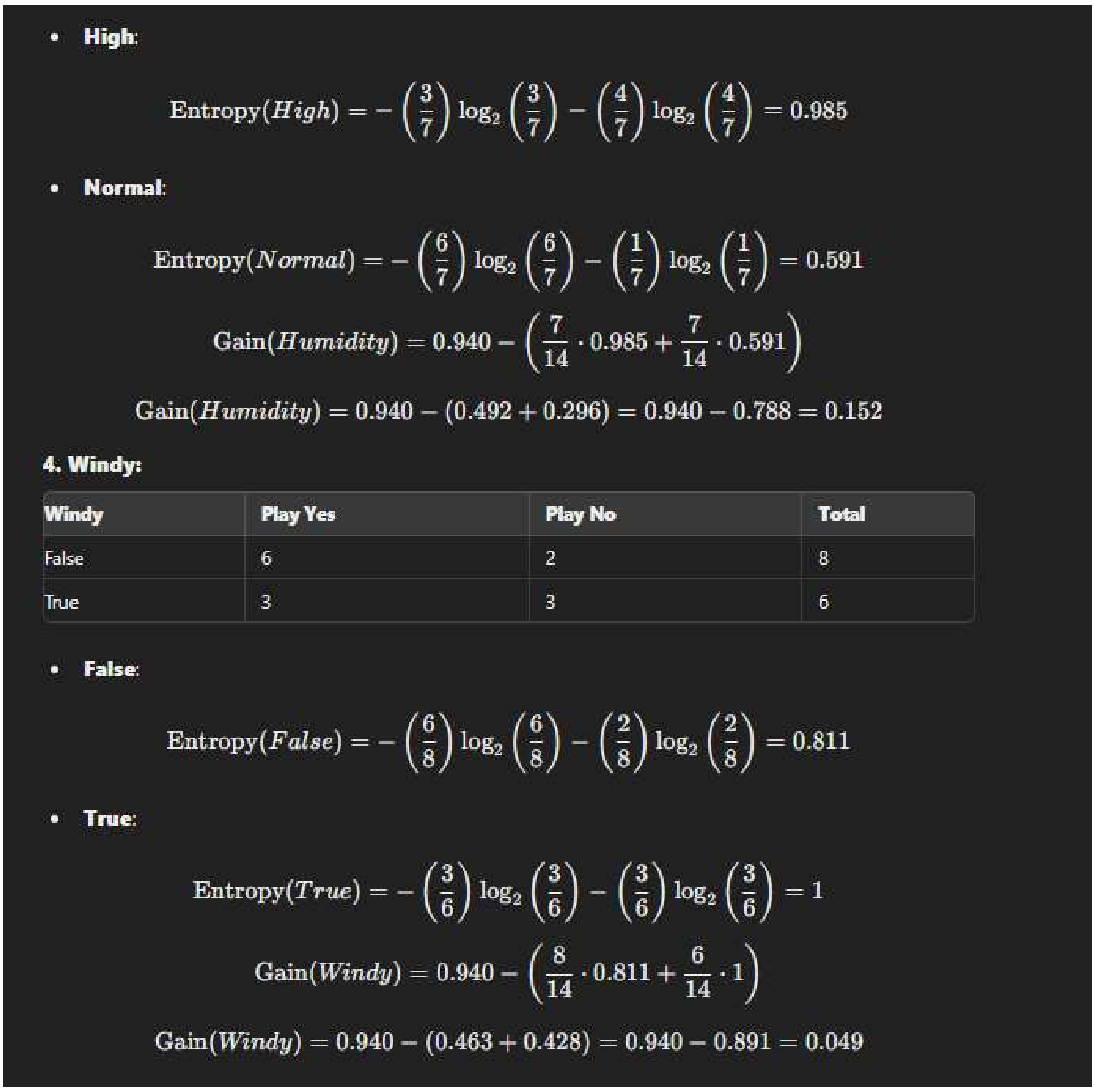
$$\text{Gain}(Temperature) = 0.940 - (0.285 + 0.393 + 0.232) = 0.940 - 0.910 = 0.030$$

3. Humidity:

Humidity	Play Yes	Play No	Total
High	3	4	7
Normal	6	1	7

- **High**:

$$\text{Entropy}(High) = -\left(\frac{3}{7}\right)\text{l[illegible]}\left(\frac{3}{7}\right) - \left(\frac{4}{7}\right)\log_2\left(\frac{4}{7}\right) = 0.985$$



- **High**:

$$\text{Entropy}(High) = -\left(\frac{3}{7}\right)\log_2\left(\frac{3}{7}\right) - \left(\frac{4}{7}\right)\log_2\left(\frac{4}{7}\right) = 0.985$$

- **Normal**:

$$\text{Entropy}(Normal) = -\left(\frac{6}{7}\right)\log_2\left(\frac{6}{7}\right) - \left(\frac{1}{7}\right)\log_2\left(\frac{1}{7}\right) = 0.591$$

$$\text{Gain}(Humidity) = 0.940 - \left(\frac{7}{14}\cdot 0.985 + \frac{7}{14}\cdot 0.591\right)$$

$$\text{Gain}(Humidity) = 0.940 - (0.492 + 0.296) = 0.940 - 0.788 = 0.152$$

4. Windy:

Windy	Play Yes	Play No	Total
False	6	2	8
True	3	3	6

- **False**:

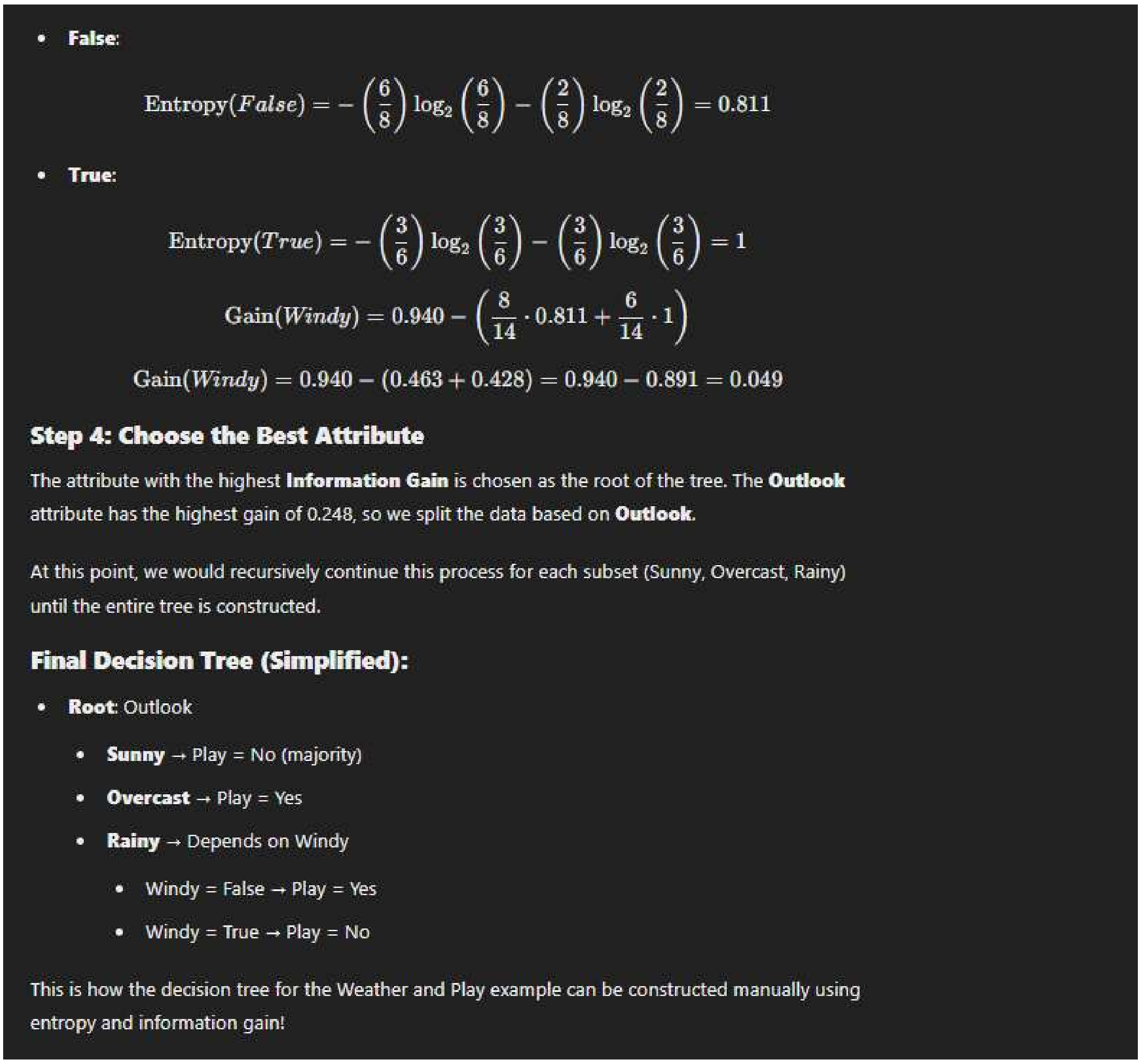
$$\text{Entropy}(False) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right) = 0.811$$

- **True**:

$$\text{Entropy}(True) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) = 1$$

$$\text{Gain}(Windy) = 0.940 - \left(\frac{8}{14}\cdot 0.811 + \frac{6}{14}\cdot 1\right)$$

$$\text{Gain}(Windy) = 0.940 - (0.463 + 0.428) = 0.940 - 0.891 = 0.049$$



- **False**:

$$\text{Entropy}(False) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right) = 0.811$$

- **True**:

$$\text{Entropy}(True) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) = 1$$

$$\text{Gain}(Windy) = 0.940 - \left(\frac{8}{14}\cdot 0.811 + \frac{6}{14}\cdot 1\right)$$

$$\text{Gain}(Windy) = 0.940 - (0.463 + 0.428) = 0.940 - 0.891 = 0.049$$

Step 4: Choose the Best Attribute

The attribute with the highest **Information Gain** is chosen as the root of the tree. The **Outlook** attribute has the highest gain of 0.248, so we split the data based on **Outlook**.

At this point, we would recursively continue this process for each subset (Sunny, Overcast, Rainy) until the entire tree is constructed.

Final Decision Tree (Simplified):

- **Root**: Outlook
  - **Sunny** → Play = No (majority)
  - **Overcast** → Play = Yes
  - **Rainy** → Depends on Windy
    - Windy = False → Play = Yes
    - Windy = True → Play = No

This is how the decision tree for the Weather and Play example can be constructed manually using entropy and information gain!