Chapter 4
Classification

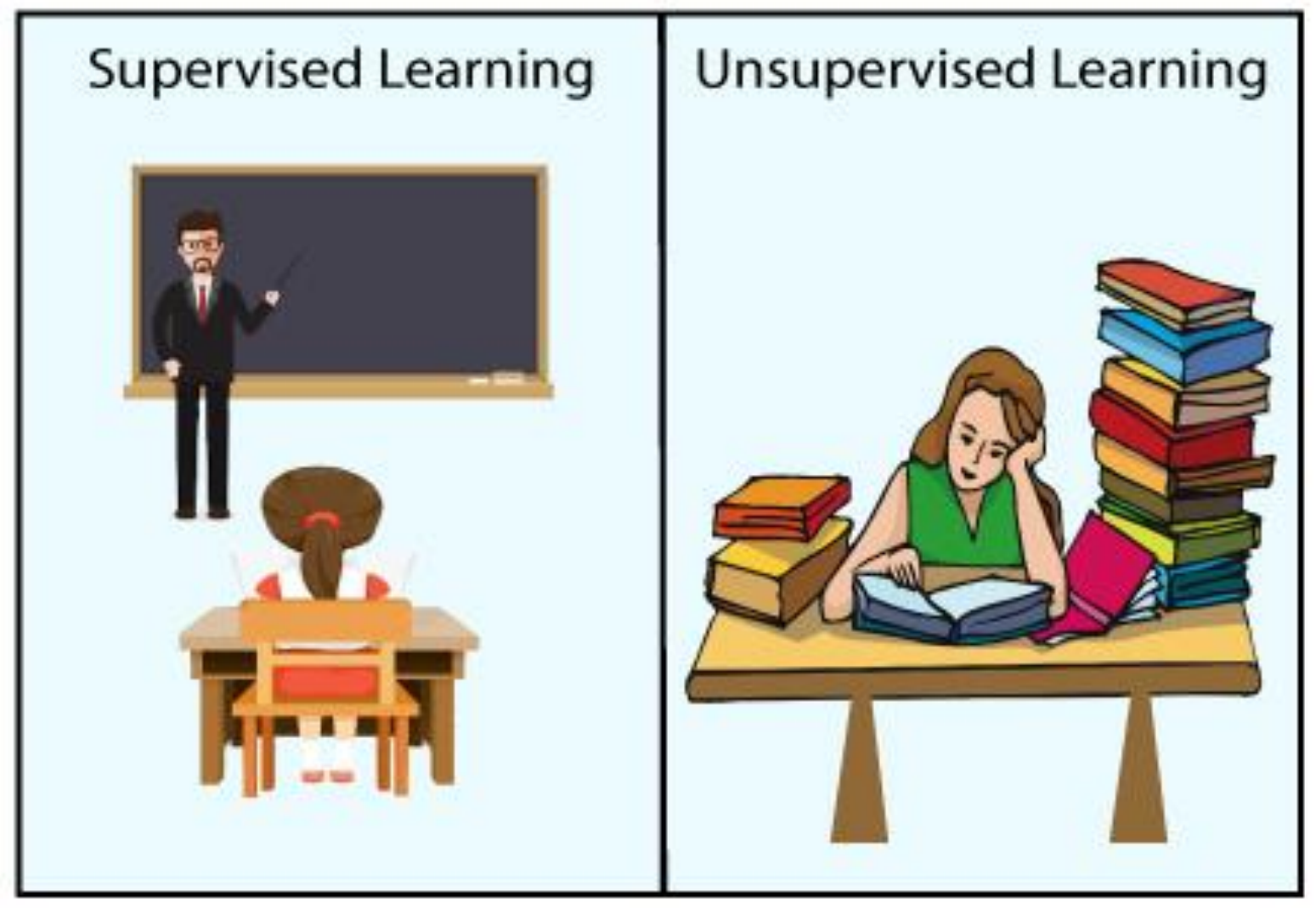
Er. Shiva Ram Dam
Assistant Professor
Gandaki University

Content:

1. Basic concepts: General Approach to classification
2. Decision Tree Classifiers
3. Rule Based Classifiers
4. Nearest Neighbor Classifiers
5. Bayesian Classifier
6. Random Forest Classifier
7. Model Evaluation and Metrics

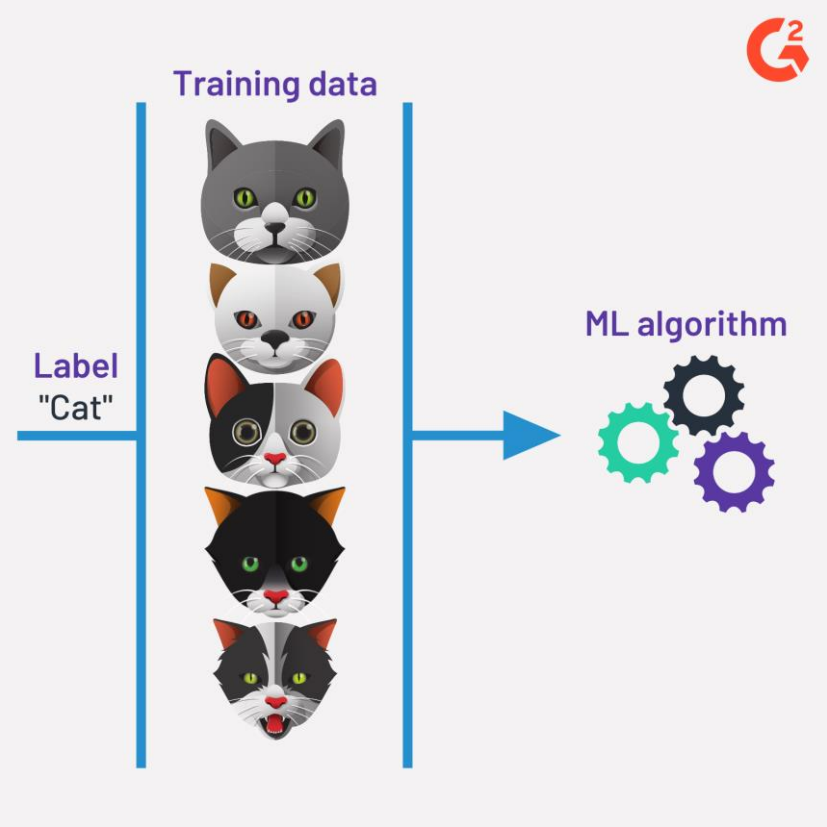
4.1 Basics and Algorithms

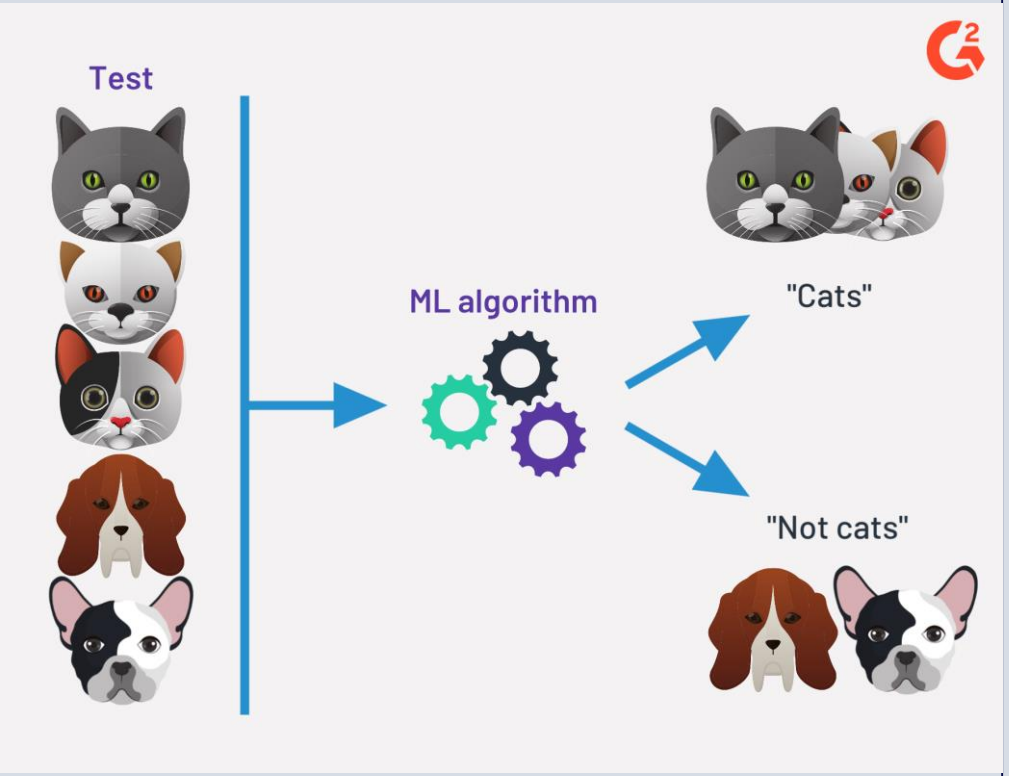
Supervised and Unsupervised Learning



a) Supervised Learning

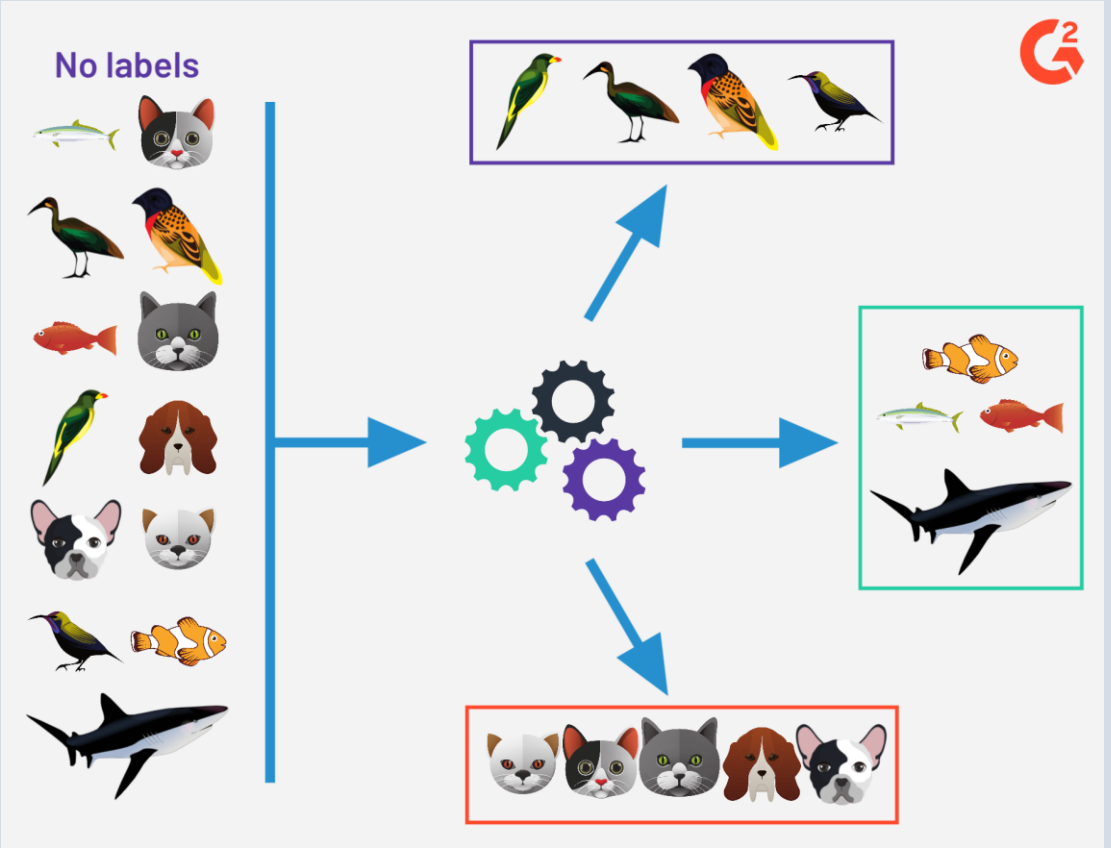
- Supervised learning is a type of machine learning that **uses labeled data** to train machine learning models.
- In labeled data, the output is already known. In Supervised Learning, the **machine learns under supervision.**
- It contains a model that is able to predict with the help of a labeled dataset.
- Supervised learning can be further divided into two types:
  1. Classification
  2. Regression


Training data
Label
"Cat"
ML algorithm


Test
ML algorithm
"Cats"
"Not cats"

b) Unsupervised Learning

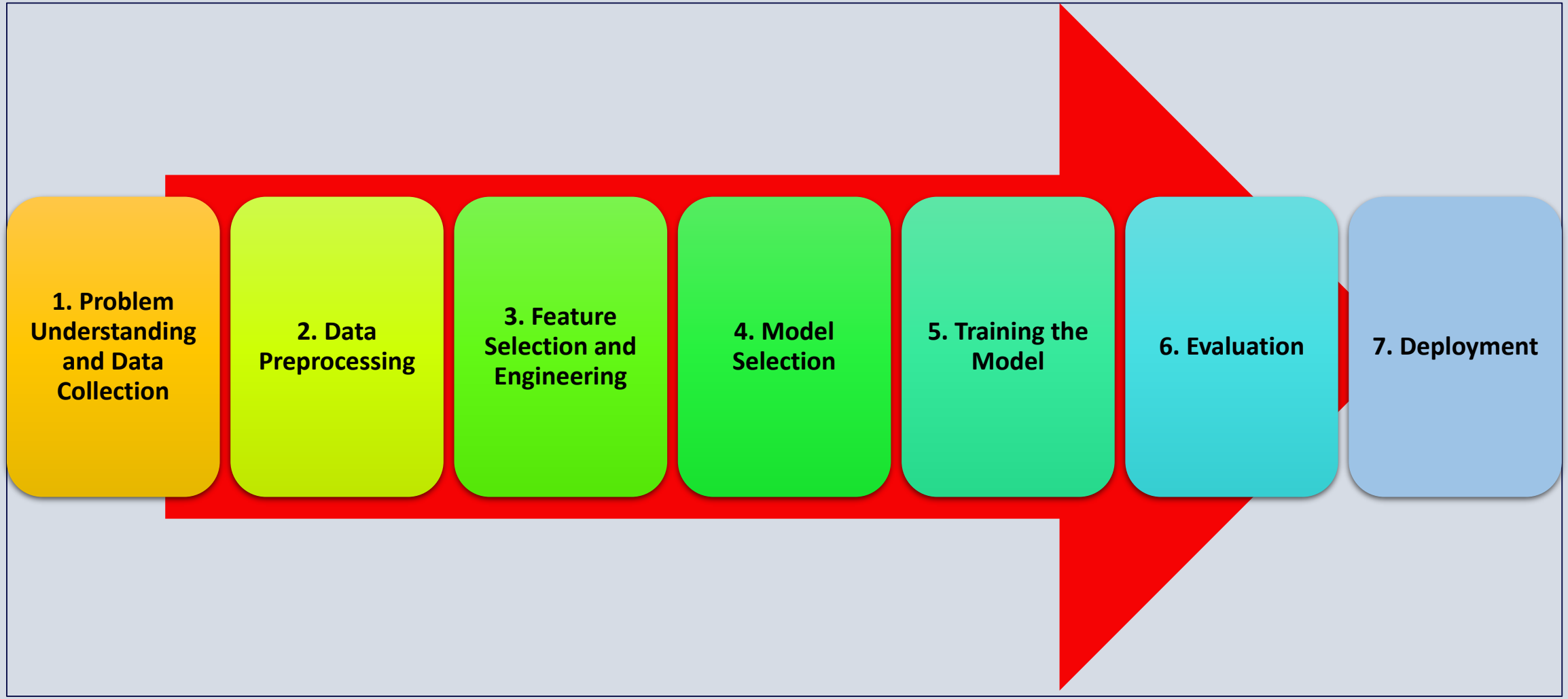
- Unsupervised learning is a type of machine learning that **uses unlabeled data** to train machines.
- Unlabeled data doesn't have a fixed output variable.
- The **model learns from the data**, discovers the patterns and features in the data, and returns the output.
- Unsupervised learning **finds patterns and understands the trends in the data** to discover the output. So, the model tries to label the data based on the features of the input data.
- Unsupervised learning can be further divided into two types:
  1. Clustering
  2. Association



Introduction to Classification

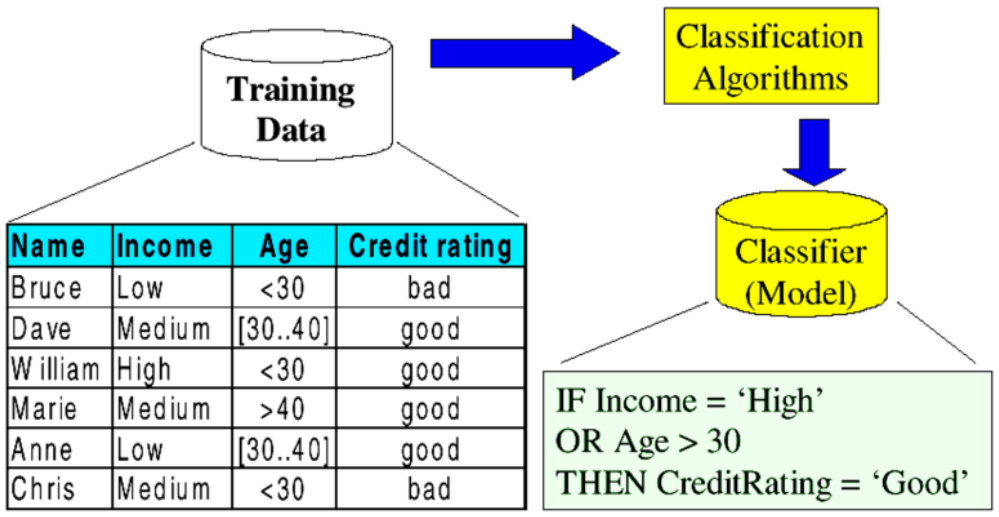
- Classification is a supervised learning technique where the goal is to assign a label or category to a given input based on its features.
- This classification process involves learning a mapping from input features to output labels based on a training dataset.
- It involves mapping input data to predefined classes or categories.
- Mainly two classification problem:
  - **Binary classification:** Binary classification is when a model can apply **only two class labels.**
  - **Multiple class classification :** Multiple class classification is when models reference **more than the two class** labels.
- Examples include:
  - **Spam Detection**: Classify emails as spam or non-spam.
  - **Medical Diagnosis**: Classify diseases based on patient symptoms.
  - **Sentiment Analysis**: Classify text as positive, negative, or neutral.

General Approach to Classification



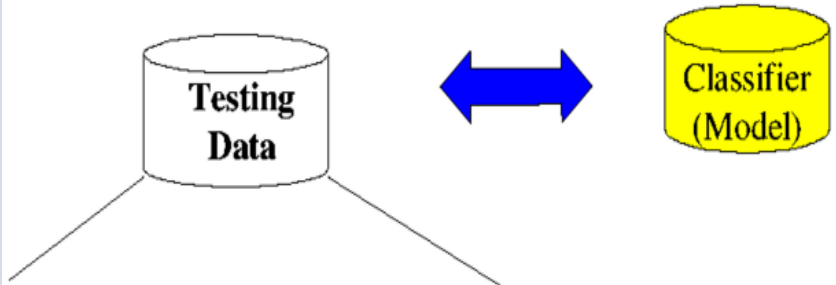
1. **Problem Understanding and Data Collection**: Define the problem and gather labeled data.
2. **Data Preprocessing**: Handle missing values, normalize data, and encode categorical variables. Split data into training and testing sets.
3. **Feature Selection and Engineering:** Identify relevant features and create new ones if needed.
4. **Model Selection**: Choose an algorithm (e.g., Logistic Regression, Decision Trees, SVM, Neural Networks).
5. **Training**: Train the model using the training set and optimize parameters.
6. **Evaluation**: Assess performance using metrics like accuracy, precision, recall, and F1-score.
7. **Deployment**: Deploy the model and monitor its performance over time.

Model Construction



Name	Income	Age	Credit rating
Bruce	Low	<30	bad
Dave	Medium	[30..40]	good
William	High	<30	good
Marie	Medium	>40	good
Anne	Low	[30..40]	good
Chris	Medium	<30	bad

Model Evaluation

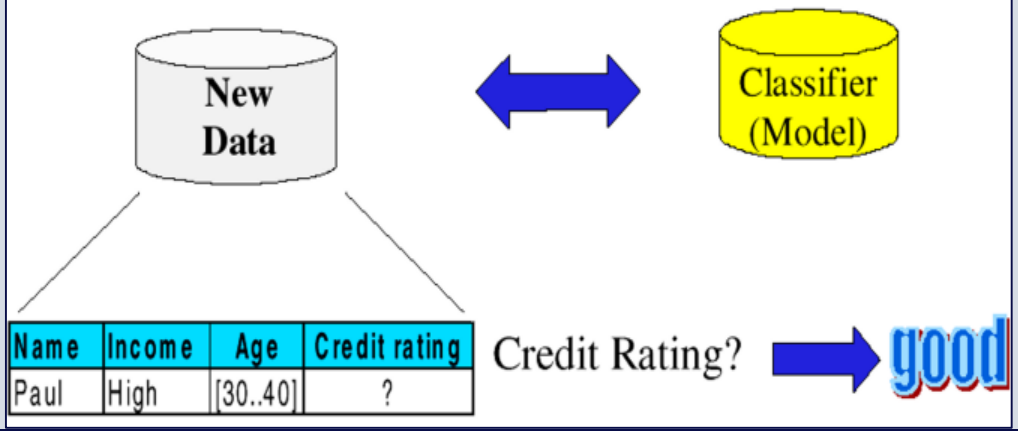


Name	Income	Age	Credit rating
Tom	Medium	<30	bad
Jane	High	<30	bad
Wei	High	>40	good
Hua	Medium	[30..40]	good

How accurate is the model?

IF Income = 'High'
OR Age > 30
THEN CreditRating = 'Good'

Model Use: Classification

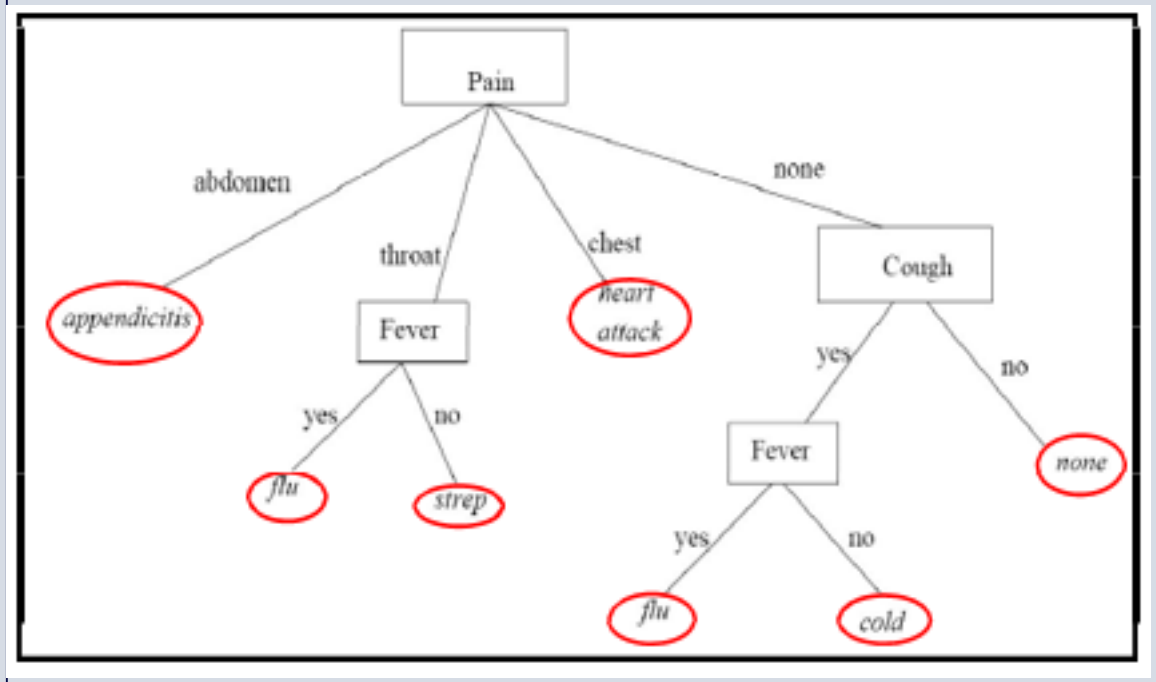


Name	Income	Age	Credit rating
Paul	High	[30..40]	?

4.2 Decision Tree Classifier

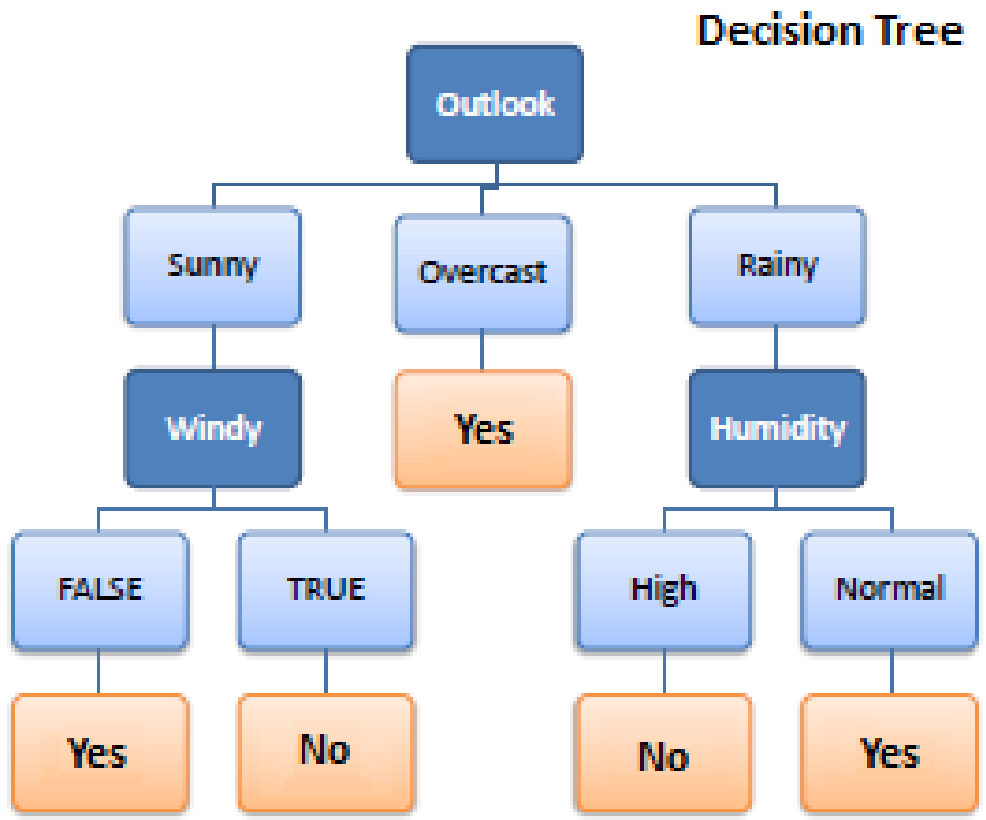
Decision Tree

- A **decision tree** is a tree-structured model used for classification and regression tasks.
- It splits the dataset into subsets based on feature values, using decision rules at each node.
- It consists of:
  - **Root Node**: Represents the entire dataset.
  - **Internal Nodes**: Represent tests on features.
  - **Leaf Nodes**: Represent class labels or output values.



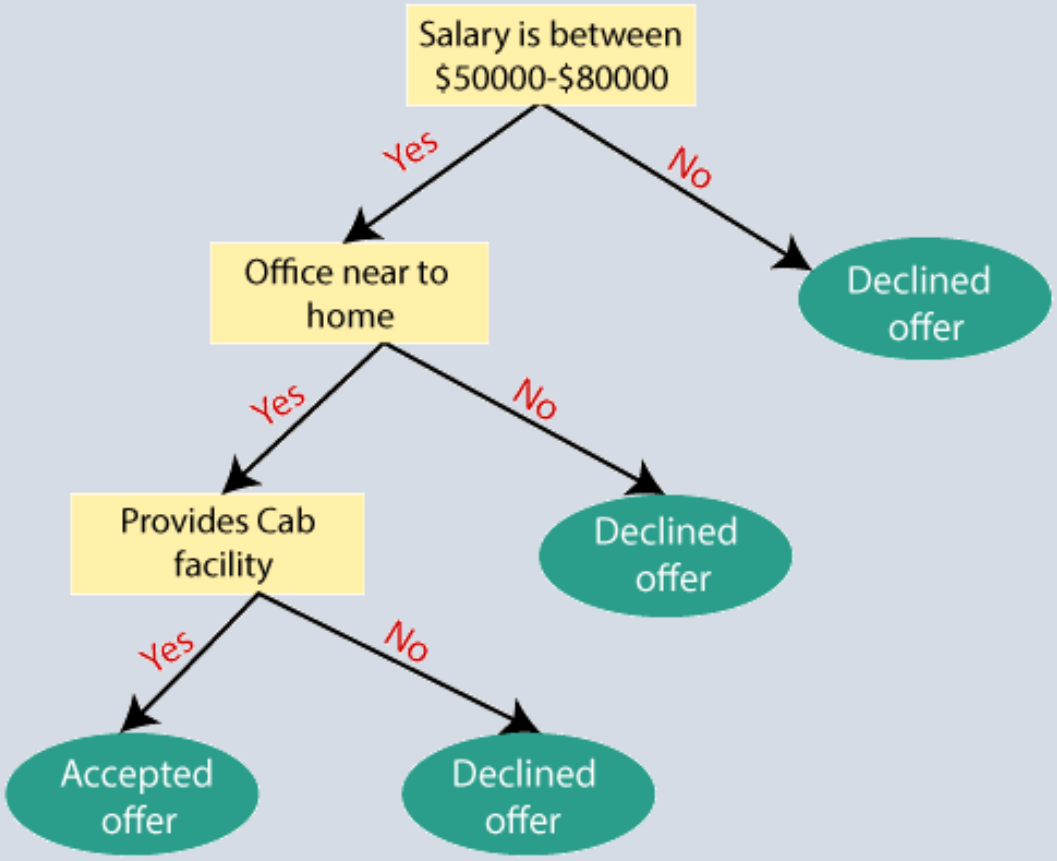
Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



How does the Decision Tree algorithm Work?

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.
- This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.



Decision Tree Construction

1. **Tree Construction**
   - Select the best feature to split the data at each node.
   - Use a metric to measure the quality of splits (e.g., Gini Index, Information Gain).
2. **Recursive Splitting**
   - Subdivide data at each node until stopping criteria are met (e.g., maximum depth, minimum samples per node).
3. **Tree Pruning**
   - Reduce overfitting by removing unnecessary branches (post-pruning or pre-pruning).
   - *Pruning is a **process of deleting the unnecessary nodes** from a tree in order **to get the optimal decision tree.***

Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes.
- So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- By this measurement, we can easily select the best attribute for the nodes of the tree.
- There are three popular techniques for ASM, which are:
  1. **Information Gain**
  2. **Gain ratio**
  3. **Gini Index**

1. Information Gain:

- Information gain is the **measurement of changes in entropy** after the segmentation of a dataset based on an attribute.
- It **calculates how much information a feature provides** us about a class.
- According to the value of information gain, **we split the node and build the decision tree.**
- A decision tree algorithm **always tries to maximize the value of information gain,** and a node/attribute having **the highest information gain is split first.**
- Information gain is **used by ID3 algorithm.**
- It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)]

Entropy:

- The entropy **is a measure of the uncertainty associated with a random variable.**
- *For given data partition D:* $Entropy\,(D) \equiv \sum_{i=1} - p_i \log_2(p_i)$

Predictors / Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Reference: https://www.saedsayad.com/decision_tree.htm

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

Information Gain:

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

- The dataset is then split on the different attributes.
- The entropy for each branch is calculated.
- Then it is added proportionally, to get total entropy for the split.
- The resulting entropy is subtracted from the entropy before the split.
- The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Decision Tree: Construction

Solved Example 1:

1. What is the entropy of this collection of training examples with respect to the target function ***Play***?
2. What is the information gain of ***Temp***, ***Humidity*** and ***Wind*** relative to these training examples?
3. Draw decision tree for the given dataset using ID3 algorithm.

Day	Temp	Humidity	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D3	Cool	High	Weak	Yes
D4	Cool	High	Strong	Yes
D5	Hot	Normal	Weak	Yes
D6	Cool	Normal	Weak	Yes
D7	Cool	Normal	Strong	No

Reference video:
https://www.youtube.com/watch?v=coOTEc-0OGw

Step1:

- Computation of entropy of each attributes:

Reference

Day	Temp	Humidity	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D3	Cool	High	Weak	Yes
D4	Cool	High	Strong	Yes
D5	Hot	Normal	Weak	Yes
D6	Cool	Normal	Weak	Yes
D7	Cool	Normal	Strong	No

$$\text{Entropy}(\text{Play}) = \text{Entropy}(4,3)$$
$$= -\left(\frac{4}{7}\log_2\frac{4}{7}\right) - \left(\frac{3}{7}\log_2\frac{3}{7}\right)$$
$$= -(-0.461) - (-0.523)$$
$$= 0.985$$

Now,

$$\text{Entropy}(\text{Play, Temp}) = P(\text{Hot}) * E(1,2) + P(\text{Cool}) * E(3,1)$$
$$= \frac{3}{7} * \left[-\left(\frac{1}{3}\log_2\frac{1}{3}\right) - \left(\frac{2}{3}\log_2\frac{2}{3}\right)\right] + \frac{4}{7}\left[-\left(\frac{3}{4}\log_2\frac{3}{4}\right) - \left(\frac{1}{4}\log_2\frac{1}{4}\right)\right]$$
$$= 0.3935 + 0.4636$$
$$= 0.8570$$

$$\text{Entropy}(\text{Play, Humidity}) = P(\text{High}) * E(2,2) + P(\text{Normal}) * E(2,1)$$
$$= \frac{4}{7} * \left[-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right] + \frac{3}{7}\left[-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right]$$
$$= 0.5714 + 0.3935$$
$$= 0.9649$$

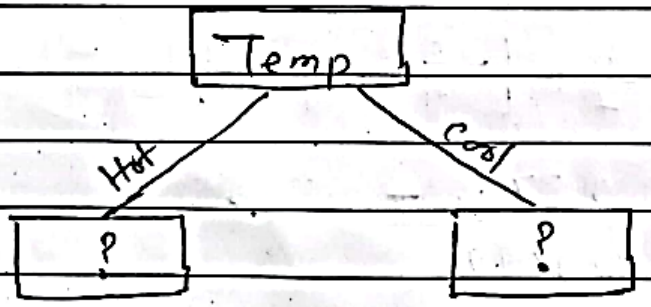
$$\text{Entropy}(\text{Play, Wind}) = P(\text{Strong}) * E(1,2) + P(\text{Weak}) * E(3,1)$$
$$= \frac{3}{7} * \left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right] + \frac{4}{7} * \left[-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right]$$
$$= 0.3935 + 0.4636$$
$$= 0.8571$$

- Computation of information gain:

Gain (Play, Temp) = Entropy (Play) - Entropy (Play, Temp)
= 0.985 − 0.8570
= 0.128
Gain (Play, Humidity) = Entropy (Play) − Entropy (Play, Humidity)
= 0.985 − 0.9649
= 0.0201
Gain (Play, Wind) = Entropy (Play) - Entropy (Play, Wind)
= 0.985 − 0.8571
= 0.1279

- Construction of partial decision tree with root node.

Since, the Gain (Play, Temp) has the highest information gain, we place this attribute (Temp) at the root node.

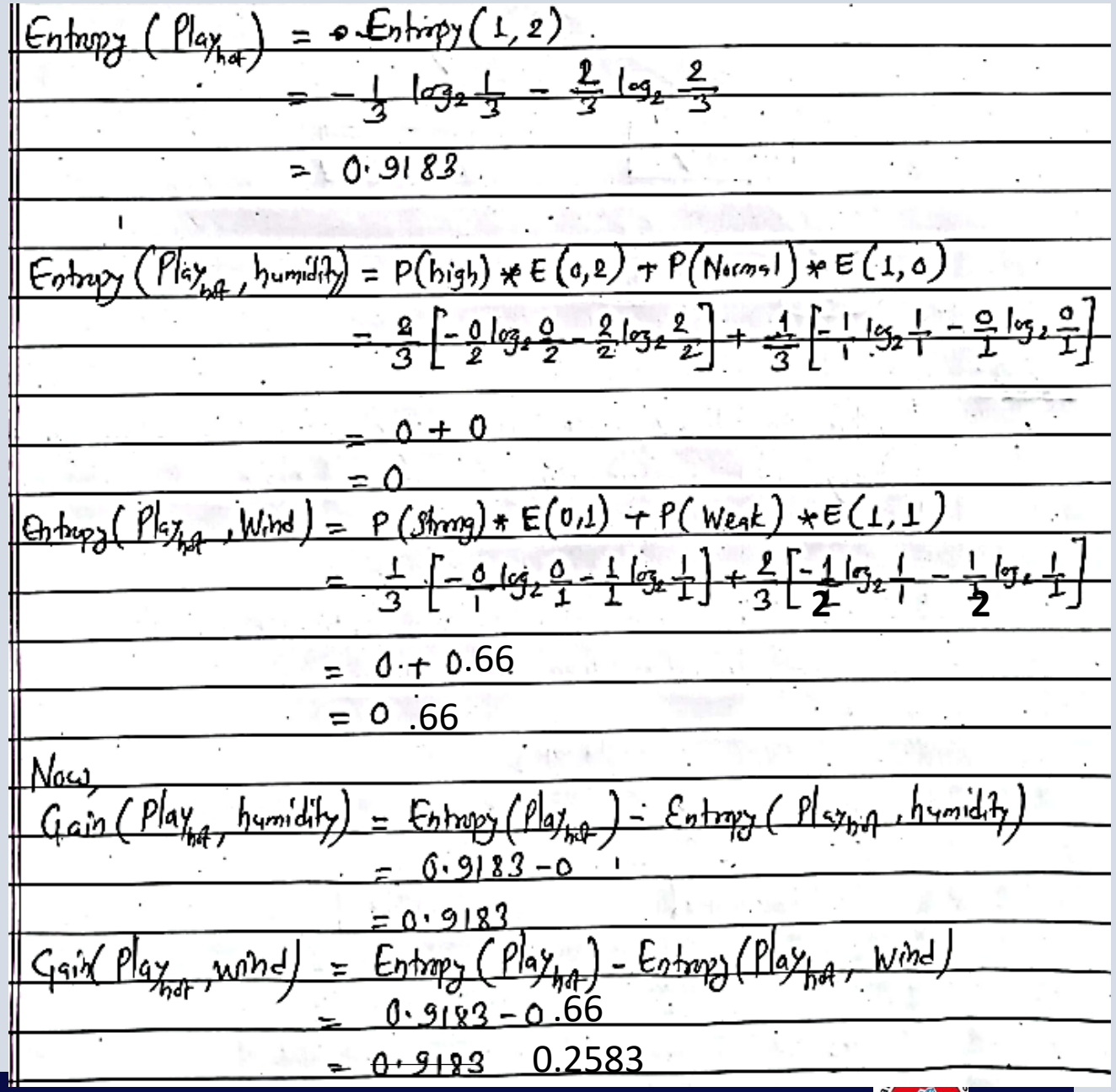


Step 2:

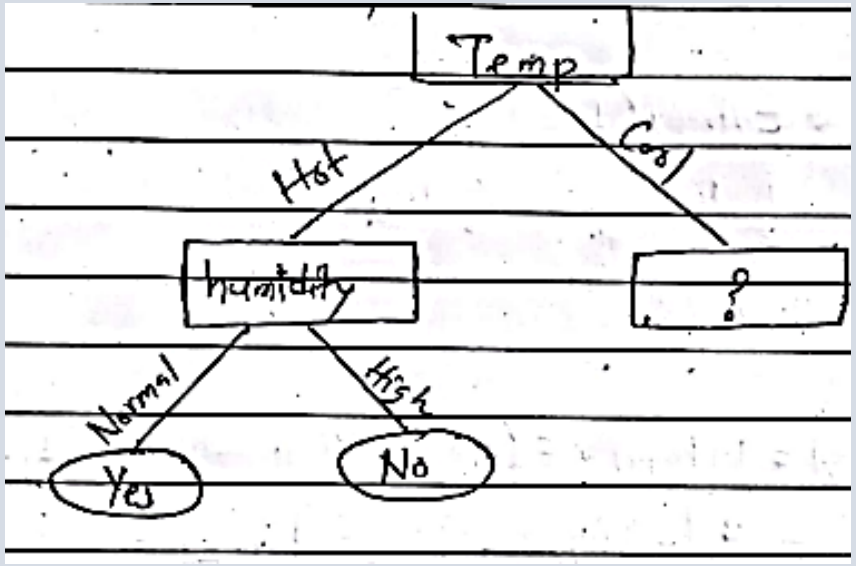
- Now again we continue to compute the entropy of each attribute (***Humidity, wind***) with reference to the data value of ***Temp*** attribute. The data values are ***Hot*** and ***Cool***.
- The new table with reference to ***Hot*** is:

Day	Humidity	Wind	Play
D1	High	Weak	No
D2	High	Strong	No
D5	Normal	Weak	Yes

- Now we compute Entropy and Information gain as in Step 1.



- Since the information gain of humidity is high we take it as the parent node.
- Our partial decision tree is:

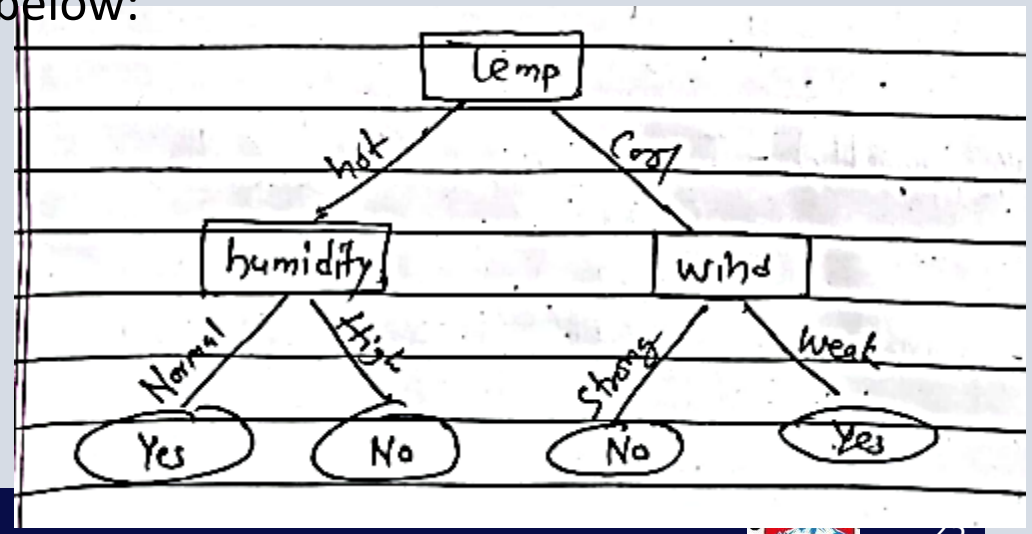


Step 3:

- Similar to step 2, we need to compute the entropy and information gain for ($Play_{Cool}$) with reference to ***wind*** and ***humidity*** attribute with below table.

Day	Humidity	Wind	Play
D3	High	Weak	Yes
D4	High	Strong	Yes
D6	Normal	Weak	Yes
D7	Normal	Strong	No

- But , since the remaining is ***wind*** attribute only. So, we can directly draw the final decision tree as below:



Python code Implementation:

- Given a dataset as:

- Classify the new data: **Outlook=Sunny, Temp=Cool, Humidity=High, Windy=False** using Decision Tree classifier and also evaluate your model.

Outlook	Temp	Humidity	Windy	Play
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

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Create the Play Golf dataset
data = {
    "Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain", "Overcast",
                "Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast", "Rain"],
    "Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",
             "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],
    "Humidity": ["High", "High", "High", "High", "Normal", "Normal", "Normal",
                 "High", "Normal", "Normal", "Normal", "High", "Normal", "High"],
    "Windy": [False, True, False, False, False, True, True,
              False, False, False, True, True, False, True],
    "Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",
             "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]
}
# Load data into a pandas DataFrame
df = pd.DataFrame(data)
df
```

```
# Encode categorical variables into numeric values
encoder = LabelEncoder()
df["Outlook"] = encoder.fit_transform(df["Outlook"]) # Convert(Overcast=0, Rain=1,
Sunny=2)
df["Temp"] = encoder.fit_transform(df["Temp"]) # Convert(Cool=0, Mild=1, Hot=2)
df["Humidity"] = encoder.fit_transform(df["Humidity"]) # Convert(High=0, Normal=1)
df["Windy"] = df["Windy"].astype(int)  # Convert boolean to integer (False=0, True=1)
df["Play"] = encoder.fit_transform(df["Play"]) # Convert(No=0, Yes=1)

# Features and target variable
X = df[["Outlook", "Temp", "Humidity", "Windy"]]
y = df["Play"]

# Train-test split with 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a Decision Tree Classifier
clf = DecisionTreeClassifier(criterion="entropy", random_state=42)  # Using
Information Gain
clf.fit(X_train, y_train)
```

```
# Evaluate the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Data: {accuracy:.2f}")


# Display the Decision Tree Rules
tree_rules = export_text(clf, feature_names=list(X.columns))
print("\nDecision Tree Rules:")
print(tree_rules)

# Visualize the Decision Tree
plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=list(X.columns),
class_names=encoder.classes_, filled=True, rounded=True)
plt.title("Decision Tree for Play Golf Dataset")
plt.show()
```

```
# Predict a new instance
# Example: Outlook=Sunny, Temp=Cool, Humidity=High, Windy=False
# Encoded values: Outlook=2, Temp=0, Humidity=0, Windy=0
new_instance = [[2, 0, 0, 0]]  # Sunny, Cool, High, Not Windy
prediction = clf.predict(new_instance)

# Decode the prediction back to the original label
predicted_label = encoder.inverse_transform(prediction)
print("\nPrediction for the new instance:")
print("Play =", predicted_label[0])
```

Output:

```
Prediction for the new instance: Play = No
```

Assignment:

1. Perform as below for below table :
   a) What is the entropy of this collection of training examples with respect to the target label “classification”?
   b) What is the information gain of a1, a2 and a3 with respective to these training examples?
   c) Draw decision tree for the given dataset.
   d) What will be the classification of a new data {“True”, “Cool”, “Low”} ?

Instance	a1	a2	a3	Classification
1	True	Hot	High	No
2	True	Hot	High	No
3	False	Hot	High	Yes
4	False	Cool	Normal	Yes
5	False	Cool	Normal	Yes
6	True	Cool	High	No
7	True	Hot	High	No
8	True	Hot	Normal	Yes
9	False	Cool	Normal	Yes
10	False	Cool	High	Yes

Reference video:
https://www.youtube.com/watch?v=JO2wiZif2OM&t=3s
https://www.youtube.com/watch?v=coOTEc-0OGw

2. Gain Ratio

- Gain Ratio is an alternative to Information Gain that is used to select the attribute for splitting in a decision tree.
- Used by C4.5 (the successor of ID3)
- It is used to **overcome the problem of bias** towards the attribute with many outcomes.
- For example,
  - Suppose we have two features, “Color” and “Size” and we want to build a decision tree to predict the type of fruit based on these two features.
  - The “Color” feature has three outcomes (red, green, yellow) and the “Size” feature has two outcomes (small, large).
  - Using the information gain method, the “Color” feature would be chosen as the best feature to split on because it has the highest information gain.
  - However, this could be a problem because the “Size” feature could be a better feature to split on because it is less ambiguous and has fewer outcomes.
- Gain Ratio **normalizes the Information Gain with respect to the total entropy** of all splits based on values of an attribute
- It introduces a normalizing term called **Intrinsic Information.**

Reference video for solved example: https://www.youtube.com/watch?v=rueA-YSLpAI

- Gain ratio is given by:

$$GainRatio = \frac{\text{Information Gain}}{\text{Intrinsic Information}}$$

- Where:

$$\text{Intrinsic Information} = -\sum_{v} \frac{|S_v|}{|S|} \cdot \log_2 \left(\frac{|S_v|}{|S|} \right)$$

- Where:
  - |S| = total number of samples before the split.
  - $|S_v|$= number of samples in subset v after the split.
  - v iterates over all possible values of the splitting attribute.

Example:

Day	Temp	Humidity	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D3	Cool	High	Weak	Yes
D4	Cool	High	Strong	Yes
D5	Hot	Normal	Weak	Yes
D6	Cool	Normal	Weak	Yes
D7	Cool	Normal	Strong	No

Here, Intrinsic Information of Temp is :

$$= -\left(\frac{4}{7} \cdot \log_2 \frac{4}{7} + \frac{3}{7} \cdot \log_2 \frac{3}{7} \right)$$

$$= 0.985$$

Gain ratio $_{Temp}$= 0.128 / 0.9851 = 0.01299

Assignment 1:

- Draw decision tree for the given dataset using Gain ratio.

Day	Temp	Humidity	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D3	Cool	High	Weak	Yes
D4	Cool	High	Strong	Yes
D5	Hot	Normal	Weak	Yes
D6	Cool	Normal	Weak	Yes
D7	Cool	Normal	Strong	No

3. Gini Index:

- Gini index is a **measure of impurity or purity used** while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- **An attribute with the low Gini index should be preferred** as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

Solved Example:

1. Compute the Gini Index for overall collection of the training examples, and construct a decision tree for below training dataset.

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Reference video:
https://www.youtube.com/watch?v=zNYdkpAcP-g

Solution:

- Step 1: Computation of Gini Index for target class attribute

Gini Index for Decision,

$$Gini(S) = 1 - \left[\left(\frac{6}{10}\right)^2 + \left(\frac{2}{10}\right)^2 + \left(\frac{1}{10}\right)^2 + \left(\frac{1}{10}\right)^2\right]$$

$$= 0.58$$

Reference

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Step 2: Computation of Gini Index for other attributes as well.

Computation of Gini Index for "Money"

Poor

$$Gini(Money_{poor}) = 1 - \left[\left(\frac{3}{3}\right)^2\right]$$

$$= 0$$

Tennis, Cinema, Stay In, Shopping

$$Gini(Money_{rich}) = 1 - \left[\left(\frac{2}{7}\right)^2 + \left(\frac{3}{7}\right)^2 + \left(\frac{1}{7}\right)^2 + \left(\frac{1}{7}\right)^2\right]$$

$$= 0.694$$

Weighted average (Money)

Poor, Rich

$$Gini(Money) = 0 * \frac{3}{10} + 0.694 * \frac{7}{10}$$

$$= 0.486$$

Computation of Gini Index for "Parents" attribute.

$$Gini(Parent_{yes}) = 1 - \left[\left(\frac{5}{5}\right)^2\right]$$
$$= 0$$

$$Gini(Parent_{No}) = 1 - \left[\left(\frac{2}{5}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{5}\right)^2\right]$$
$$= 0.72$$

Now, Weighted average (Parents)

$$Gini(Parent) = 0 * \left(\frac{5}{10}\right) + 0.72 * \left(\frac{5}{10}\right)$$
$$= 0.36$$

Computation of Gini Index for "Weather" attribute.

$$Gini(Weather_{sunny}) = 1 - \left[\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right]$$
$$= 0.444$$

$$Gini(Weather_{rainy}) = 1 - \left[\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right]$$
$$= 0.444$$

$$Gini(Weather_{windy}) = 1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right]$$
$$= 0.375$$

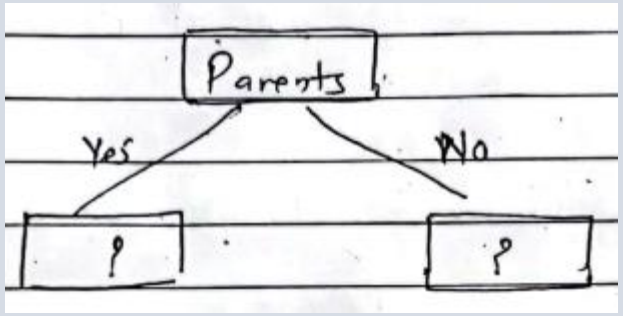
Weighted average (Weather)

$$Gini(Weather) = 0.444 * \frac{3}{10} + 0.444 * \frac{3}{10} + 0.375 * \frac{4}{10}$$
$$= 0.416$$

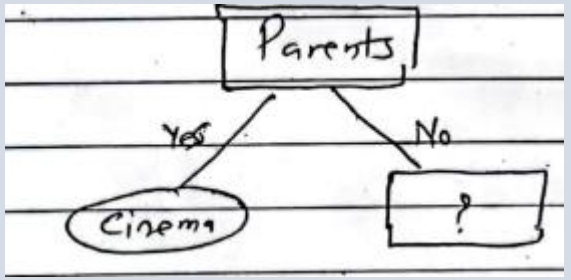
Reference

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

- Since the Gini Index of ***Parent*** is the smallest, we put this attribute at the root node.
- Our partial decision tree is:



- Here, from the table if the value for ***Parents*** is ***Yes***, then the decision is always ***Cinema***. So our decision tree is:



- Now our remaining table with ***Parents*** = ***No*** is:

Weekend	Weather	Parents	Money	Decision
W2	Sunny	No	Rich	Tennis
W5	Rainy	No	Rich	Stay In
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W10	Sunny	No	Rich	Tennis

Reference

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

- **Step 3:** Computation of Gini Index for ***Weather*** and ***Money*** with ***Parents*=*No***

Computation of Gini Index for attribute Weather:

$$Gini(Weather_{sunny}) = 1 - \left[\left(\frac{2}{2}\right)^2\right]$$
$$= 0$$

$$Gini(Weather_{rainy}) = 1 - \left[\left(\frac{1}{1}\right)^2\right]$$
$$= 0$$

Cinema Cinema

$$Gini(Weather_{windy}) = 1 - \left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right]$$
$$= 0.5$$

Weighted average Gini

$$Gini(Weather) = 0 * \frac{2}{5} + 0 * \frac{1}{5} + 0.5 * \frac{2}{5}$$
$$= 0.2$$

Computation of Gini Index for attribute Money:

Tennis StayIn

$$Gini(Money_{rich}) = 1 - \left[\left(\frac{2}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right]$$
$$= 0.625$$

$$Gini(Money_{poor}) = 1 - \left[\left(\frac{1}{1}\right)^2\right]$$
$$= 0$$

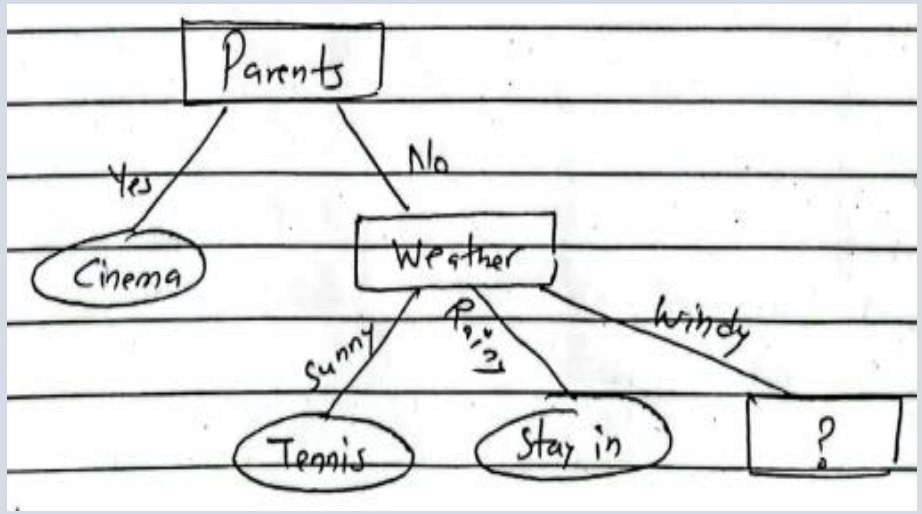
Weighted average Gini

$$Gini(Money) = 0.625 * \frac{4}{5} + 0 * \frac{1}{5}$$
$$= 0.5$$

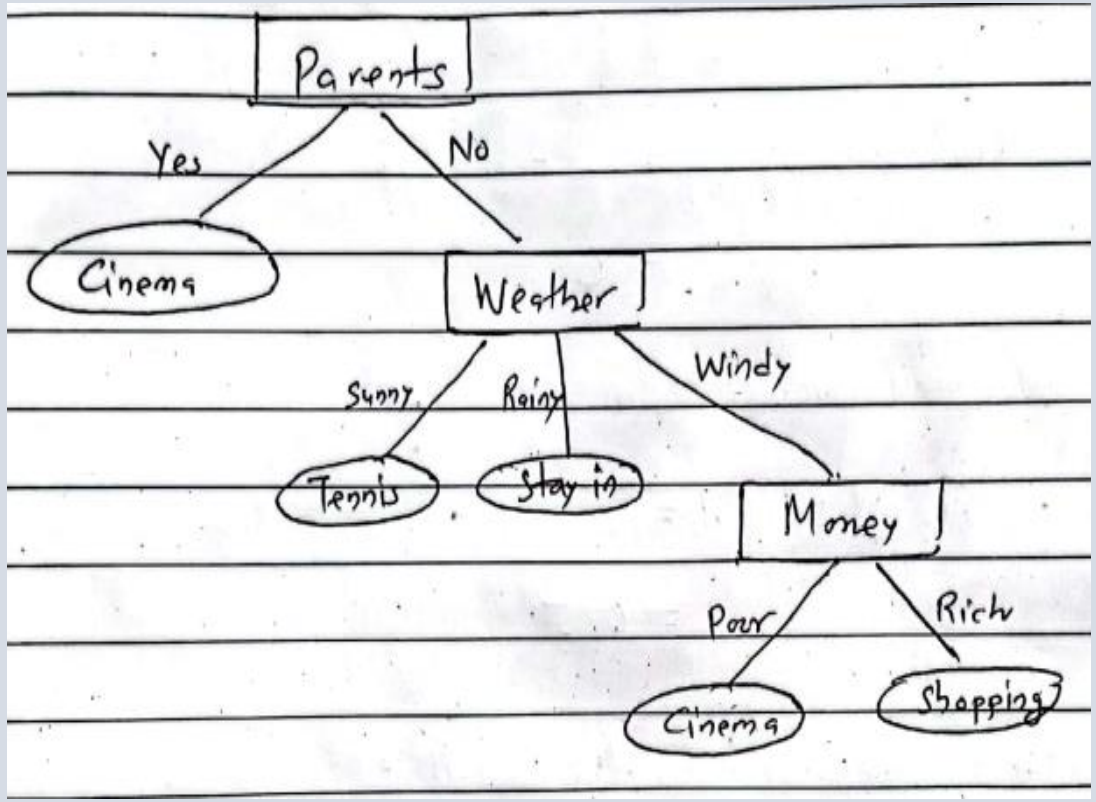
Reference

Weekend	Weather	Parents	Money	Decision
W2	Sunny	No	Rich	Tennis
W5	Rainy	No	Rich	Stay In
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W10	Sunny	No	Rich	Tennis

- Here, ***weather*** has the smallest Gini Index. So, it is selected as the next node in the decision tree as shown below:



- Definitely, the next remaining attribute is ***Money***. So, it is the next level node. Hence, out final decision tree is as:



Reference

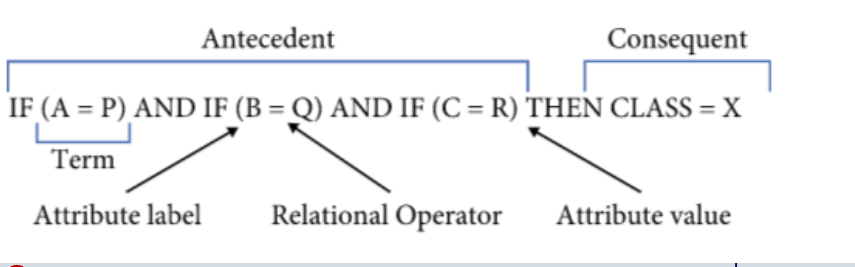
Weekend	Weather	Parents	Money	Decision
W2	Sunny	No	Rich	Tennis
W5	Rainy	No	Rich	Stay In
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W10	Sunny	No	Rich	Tennis

Assignment 2:

- Download Iris Dataset.
- Split the dataset with Training and Testing dataset with 80:20 ratio.
- Create a model with Decision Tree classifier using Gini Index and maximum depth= 3.
- Perform Train-test with your model and evaluate your model.
- Also visualize the decision tree.

4.3 Rule Based Classifier

Rule Based Classifier



- Rule-based classifiers are classifiers which **uses various "if..else" rules.**
- To make a decision tree model more readable, a path to each leaf can be transformed into an IF-THEN production rule.
- The IF part consists of all tests on a path, and the THEN part is a final classification.
- Rules in this form are called **decision rules.**
- An IF-THEN rule is an expression of the form:
  - IF *condition* THEN *conclusion*
- Let us consider a rule R1,

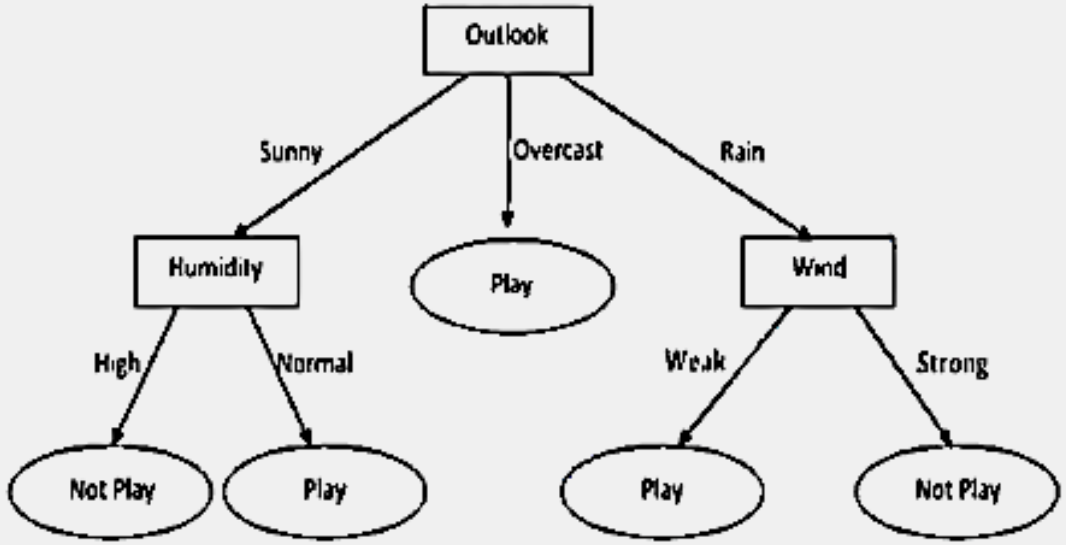
```
R1: IF age = youth AND student = yes
    THEN buy_computer = yes
```

- The condition used with "if" is called the **antecedent** and the predicted class of each rule is called the **consequent**.
- We can also write rule R1 as follows –

```
R1: (age = youth) ^ (student = yes))(buys computer = yes)
```

Rule extraction from Decision Tree

- To extract rules form a Decision tree, **one rule is created for each path from the root** to a leaf node.
- Each **splitting criterion is logically ANDed** to form the rule antecedent (IF part).
- **Leaf node holds the class prediction** for rule consequent (THEN part).



For the decision tree above, there are five possible rules which can be extracted (because there are five leaf nodes). They are as follows:

R1: IF *Outlook* = *sunny*	AND *Humidity* = *High*	THEN *Play_Tennis* = *no*
R2: IF *Outlook* = *sunny*	AND *Humidity* = *Normal*	THEN *Play_Tennis* = *yes*
R3: IF *Outlook* = *Overcast*		THEN *Play_Tennis* = *yes*
R4: IF *Outlook* = *Rain*	AND *Wind* = *Weak*	THEN *Play_Tennis* = *yes*
R5: IF *Outlook* = *Rain*	AND *Wind* = *Strong*	THEN *Play_Tennis* = *no*

Characteristics of Rule Based Data Mining Classifiers

Rule Based Data Mining classifiers possess two significant characteristics:

1. **Rules may not be mutually exclusive.**

   Different rules are generated for data, so it is possible that many rules can cover the same record. That is why rules are called non-mutually exclusive.

2. **Rules may not be exhaustive.**

   It is possible that some of the data entries may not be covered by any of the rules; thus, rules are called not to be exhaustive.

Advantages Rule Based Classifier

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Assignment 3

- Suppose we have a dataset of weather conditions and whether to play tennis.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Overcast	Cool	Normal	Strong	Yes
Rainy	Mild	High	Weak	Yes

- **and the Generated Rules** are:
  - IF outlook = 'Sunny' AND humidity = 'High' THEN Play Tennis = 'No'
  - IF outlook = 'Overcast' THEN Play Tennis = 'Yes'
  - IF outlook = 'Rainy' AND wind = 'Weak' THEN Play Tennis = 'Yes‘
- Write python program to create a rule-based classifier model for above and then predict “Play Tennis” for a new instance:
- {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Strong'}

```python
# Rule-based classifier function
def predict_play_tennis(instance):
    outlook = instance['Outlook']
    humidity = instance['Humidity']
    wind = instance['Wind']

    # Applying rules
    if outlook == 'Sunny' and humidity == 'High':
        return 'No'
    elif outlook == 'Overcast':
        return 'Yes'
    elif outlook == 'Rainy' and wind == 'Weak':
        return 'Yes'
    else:
        return 'Unknown'  # Default case if no rule matches
# New instance
new_instance = {
    'Outlook': 'Sunny',
    'Temperature': 'Cool',
    'Humidity': 'High',
    'Wind': 'Strong'
}
# Prediction
prediction = predict_play_tennis(new_instance)
print(f"Prediction for Play Tennis: {prediction}")
```

Output:

```
Prediction for Play Tennis: No
```

4.4 Nearest Neighbor Classifier

Nearest Neighbor Classifier

- KNN is the simplest, **supervised** , non-parametric learning algorithm and **lazy learning algorithm** used for classification and for regression in data mining.
- K- NN algorithm is based on the principle that, **"the similar things or objects exist closer to each other."**
- KNN classifies the data points based on the different kind of similarity measures (e.g. Euclidean distance, Manhattan distance, Hamming distance, Cosine Similarity, Minkowski distance, etc).
- In KNN algorithm 'K' refers to the number of neighbors to consider for classification. The value of 'K' in KNN algorithm must be selected carefully otherwise it may cause defects in our model.
- Advantages of KNN Algorithm:
  - It is simple to implement.
  - It is robust to the noisy training data
  - It can be more effective if the training data is large.
- Disadvantages of KNN Algorithm:
  - Always needs to determine the value of K which may be complex some time.
  - The computation cost is high because of calculating the distance between the data points for all the training samples

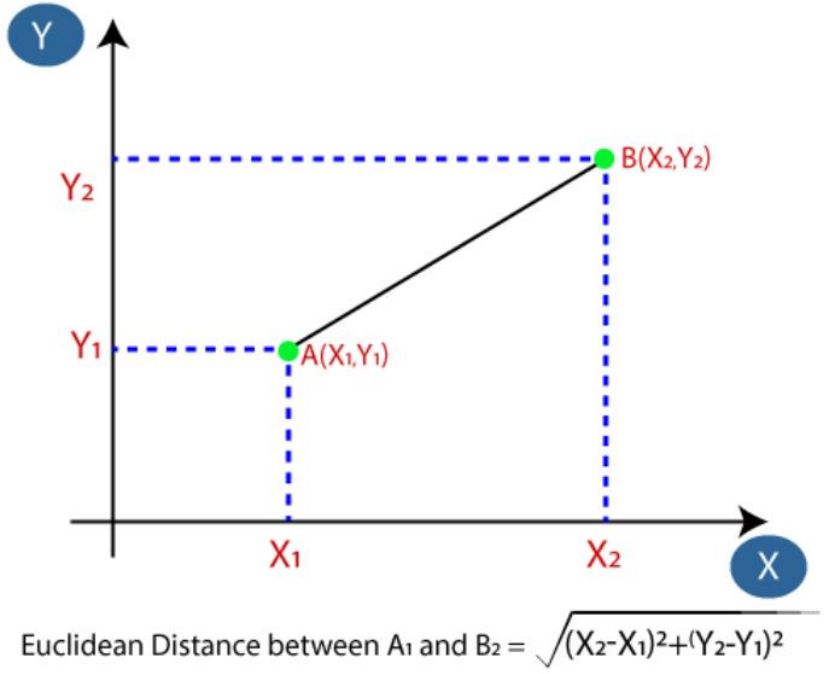
Hamming Distance

- Given two integers, the task is to find the hamming distance between two integers.
- Hamming Distance between two integers is the number of bits that are different at the same position in both numbers.
- Hamming distance= 3

Examples:

```
Input: n1 = 9, n2 = 14
Output: 3
9 = 1001, 14 = 1110
No. of Different bits = 3
```

Euclidian Distance



- Let x1 and x2 are the attribute values of two instances.
- Then, in hamming distance, if the categorical values are same or matching, i.e. x1 is same as x2, then the distance is 0, otherwise 1.
- For example:
  - If value of **x1 is blue** and **x2 is also blue**, then the hamming distance between x1 and x2 is **0.**
  - If value of **x1 is blue** and **x2 is red**, then the hamming distance between x1 and x2 is **1**.

KNN algorithm

- **Step-1:** Select the parameter K (Eg., 1, 3, 5)
- **Step-2:** Calculate the distance of the new data (test instance) with all the training data.
- **Step 3:** Sort distances in ascending order. And Select K (1 or 3 or 5) nearest neighbors.
- **Step-4:** Among these K nearest neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that *category (i.e.. Target class)* for which the number of the neighbor is maximum.

Example 1

1. **Apply K Nearest Neighbor classifier to predict the diabetic patient with the given features (BMI, Age) of Training dataset. The target label is Sugar. The test example is: BMI= 43.6 and Age= 40, Sugar= ? . Assume K=3**

BMI	Age	Sugar
33.6	50	1
26.6	30	0
23.4	40	0
43.1	67	0
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	0
23.3	29	0
31	56	1

BMI	Age	Sugar
33.6	50	1
26.6	30	0
23.4	40	0
43.1	67	0
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	0
23.3	29	0
31	56	1

- First Calculate the distance between the test instance and training instances.

Test Example

BMI=43.6, Age=40, Sugar=?

- $Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Step 2: Take the K nearest neighbors as per the calculated Euclidean distance.

BMI	Age	Sugar	Distance	Rank
33.6	50	1	14.14	2
26.6	30	0	19.72	
23.4	40	0	20.20	
43.1	67	0	27.00	
35.3	23	1	18.92	
35.9	67	1	28.08	
36.7	45	1	8.52	1
25.7	46	0	18.88	3
23.3	29	0	23.09	
31	56	1	20.37	

Step-1: Calculate the distance of the new data with all the training data.

BMI	Age	Sugar	Distance	
33.6	50	1	$\sqrt{(43.6-33.6)^2+(40-50)^2}$	14.14
26.6	30	0	$\sqrt{(43.6-26.6)^2+(40-30)^2}$	19.72
23.4	40	0	$\sqrt{(43.6-23.4)^2+(40-40)^2}$	20.20
43.1	67	0	$\sqrt{(43.6-43.1)^2+(40-67)^2}$	27.00
35.3	23	1	$\sqrt{(43.6-35.3)^2+(40-23)^2}$	18.92
35.9	67	1	$\sqrt{(43.6-35.9)^2+(40-67)^2}$	28.08
36.7	45	1	$\sqrt{(43.6-36.7)^2+(40-45)^2}$	8.52
25.7	46	0	$\sqrt{(43.6-25.7)^2+(40-46)^2}$	18.88
23.3	29	0	$\sqrt{(43.6-23.3)^2+(40-29)^2}$	23.09
31	56	1	$\sqrt{(43.6-31)^2+(40-56)^2}$	20.37

Test Example

BMI=43.6, Age=40, Sugar=?

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that *category (i.e.. Target class)* for which the number of the neighbor is maximum

BMI	Age	Sugar	Distance	Rank
33.6	50	1	14.14	2
26.6	30	0	19.72	
23.4	40	0	20.20	
43.1	67	0	27.00	
35.3	23	1	18.92	
35.9	67	1	28.08	
36.7	45	1	8.52	1
25.7	46	0	18.88	3
23.3	29	0	23.09	
31	56	1	20.37	

Test Example

BMI=43.6, Age=40, Sugar=?

Sugar = 1

Assignment 4

- Write python code to implement KNN classifier (with K =3) for predicting the new point with reference to below dataset.

BMI	Age	Sugar
33.6	50	1
26.6	30	0
23.4	40	0
43.1	67	0
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	0
23.3	29	0
31	56	1

Test Example

BMI=43.6, Age=40, Sugar=?

Practical Session:

Perform in https://colab.research.google.com

```
[2]  import matplotlib.pyplot as plt
     from sklearn.neighbors import KNeighborsClassifier
```

```
[3]  x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
     y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
     classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
```

```
[4]  data = list(zip(x, y))
     print(data)
```

```
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (8, 22), (10, 21), (12, 21)]
```

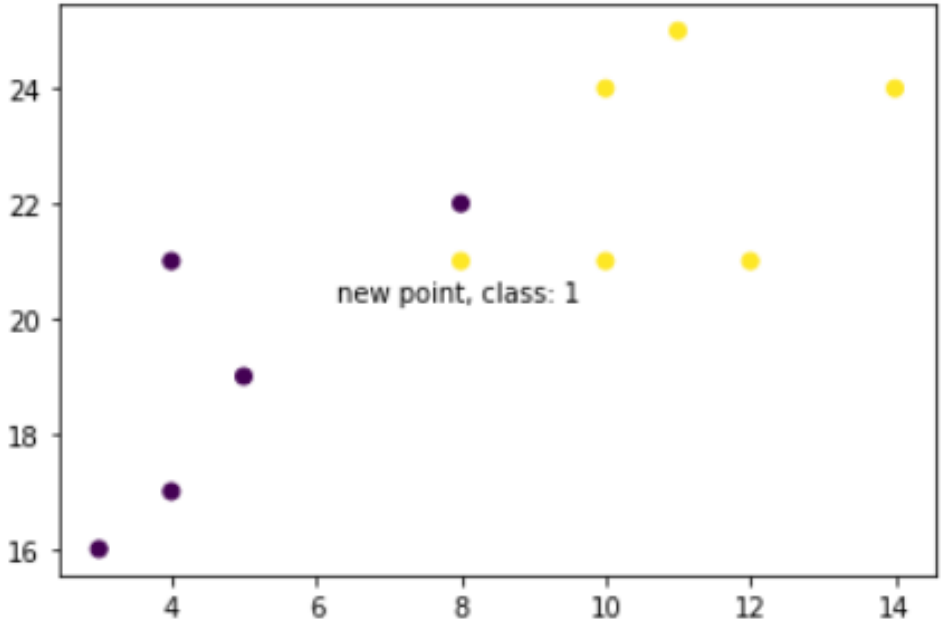
```
[11] knn = KNeighborsClassifier(n_neighbors=5)
     knn.fit(data, classes)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
[9]  new_x = 8
     new_y = 21
     new_point = [(new_x, new_y)]
     prediction = knn.predict(new_point)
     print(prediction)
```

```
     [1]
```

```
[10] plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
     plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
     plt.show()
```



Example 2

2. **"Restaurant A" sells burger with optional flavors: Pepper, Ginger and Chilly. Everyday this week you have tried a burger (A to E) and kept a record of which you liked.**

	Pepper	Ginger	Chilly	Liked
A	True	True	True	False
B	True	False	False	True
C	False	True	True	False
D	False	True	False	True
E	True	False	False	True

- **Using Hamming distance, show how the KNN classifier (k=3) would classify (liked or Not liked) with majority voting for burger with below flavor:**
  - **Pepper: false,**
  - **Ginger: true,**
  - **Chilly: true**

Reference Video: https://www.youtube.com/watch?v=T0YkfWssHjk

Step 1: Finding the Hamming distance from query example (Q) to Training examples (A-E),

	Pepper	Ginger	Chilly	Liked	Distance
A	True	True	True	False	1 + 0 + 0 = 1
B	True	False	False	True	1 + 1 + 1 = 3
C	False	True	True	False	0 + 0 + 0 = 0
D	False	True	False	True	0 + 0 + 1 = 1
E	True	False	False	True	1 + 1 + 1 = 3

New Example - Q:

1 0 0

pepper: false, ginger: true, chilly : true

Step 2: Take the K nearest neighbors as per the calculated Hamming distance.

	Pepper	Ginger	Chilly	Liked	Distance	3NN
A	True	True	True	False	1 + 0 + 0 = 1	2
B	True	False	False	True	1 + 1 + 1 = 3	
C	False	True	True	False	0 + 0 + 0 = 0	1
D	False	True	False	True	0 + 0 + 1 = 1	2
E	True	False	False	True	1 + 1 + 1 = 3	

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that *category (i.e.. Target clas* for which the number of the neighbor is maximum

	Pepper	Ginger	Chilly	Liked	Distance	3NN
A	True	True	True	False	1 + 0 + 0 = 1	2
B	True	False	False	True	1 + 1 + 1 = 3	
C	False	True	True	False	0 + 0 + 0 = 0	1
D	False	True	False	True	0 + 0 + 1 = 1	2
E	True	False	False	True	1 + 1 + 1 = 3	

Hence, the classification by majority voting is : **Unliked (i.e False)**

Example 3:

Assume the following training set with two classes, Food and Beverage. Apply KNN with K=3 to classify the new document "turkey soda".

Food : "turkey stuffing"
Food : "buffalo wings"
Beverage : "cream soda"
Beverage : "orange soda"

Here, the vocabulary is: **Buffalo, Cream, Orange, Soda, Stuffing, Turkey, Wings**

Applying one-hot encoding and calculating the Euclidean distance of each point with the test data, we get:

	Buffalo	Cream	Orange	Soda	Stuffing	Turkey	Wings	Category	Euclidian Distance	Rank with 3NN
D1: "Turkey Stuffing"	0	0	0	0	1	1	0	Food	$\sqrt{(0-1)^2+(1-0)^2} = \sqrt{2} = 1.414$	1
D2: "Buffalo Wings"	1	0	0	0	0	0	1	Food	$\sqrt{(1-0)^2+(0-1)^2+(0-1)^2+(1-0)^2} = \sqrt{4} = 2$	
D3: "Cream Soda"	0	1	0	1	0	0	0	Beverage	$\sqrt{(1-0)^2+(0-1)^2} = \sqrt{2} = 1.414$	2
D4: " Orange Soda"	0	0	1	1	0	0	0	Beverage	$\sqrt{(1-0)^2+(0-1)^2} = \sqrt{2} = 1.414$	3

Q : "Turkey Soda"	0	0	0	1	0	1	0

Hence, the classification for "Turkey Soda" by majority voting is : **Beverage**

For more examples:

- Reference Videos:
- https://www.youtube.com/watch?v=Vk9lGGODaJA
- https://www.youtube.com/watch?v=HZT0lxD5h6k
- https://www.youtube.com/watch?v=kCNpLbCUo7g

Assignment:

1. Write the algorithm for K nearest neighbor algorithm.
2. Assume the following training set with two classes, Food and Beverage. Apply KNN with K=3 to classify the new document “turkey soda”.

Food : "turkey stuffing"

Food : "buffalo wings"

Beverage : "cream soda"

Beverage : "orange soda"

4.5 Bayesian Classifier

3.5 Bayesian Classifier

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- **It is a probabilistic classifier, based on Bayes' theorem which means it predicts on the basis of the probability of an object**.
- **Bayes' Theorem:**
  - Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
  - The formula for Bayes' theorem is given as:

$$P(Y|X) = \frac{P(X|Y)\ P(Y)}{P(X)}$$

  - **Where,**
    - **P(Y|X) is Posterior probability**: Probability of hypothesis Y on the observed event X.
    - **P(X|Y) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.

Algorithm:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Example 1:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

1. Problem: Given below dataset, If the *weather* is *sunny*, then the Player should play or not?

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Solution: To solve this, first consider the below dataset:

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	4

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= **5/14 = 0.35**

P(Yes): **10/14 = 0.71**

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= **4/14 = 0.29**

P(Sunny)= **5/14 = 0.35**

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Example 2:

2. Estimate conditional probabilities of each attributes (color, height, smelly) for the species classes: (M, H) using the data given in the table.

Using these probabilities, estimate the probability values for the new instance: *color=green, legs=2, height=tall and smelly= No*. Classify the species for this entity.

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

Reference Video:
https://www.youtube.com/watch?v=z8K-598fqSo

2. Estimate conditional probabilities of each attributes (color, height, smelly) for the species classes: (M, H) using the data given in the table. Using these probabilities, estimate the probability values for the new instance: *color=green, legs=2, height=tall and smelly= no*. Classify the species for this entity.

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

We have : Bayes Theorem as:

$$P(\text{Class} \mid \text{New Instance}) = \frac{P(\text{New Instance} \mid Y) \quad P(\text{Class})}{P(\text{New Instance})}$$

Since P(New Instance) is constant for all classes, we ignore it in the calculations:

$$P(\text{Class} \mid \text{New Instance}) \propto P(\text{Class}) \cdot P(\text{Color} \mid \text{Class}) \cdot P(\text{Legs} \mid \text{Class}) \cdot P(\text{Height} \mid C) \cdot P(\text{Smelly} \mid \text{Class})$$

For example,
for M:

$$P(M \mid \text{New Instance}) = P(M) \cdot P(\text{Color} = \text{Green} \mid M) \cdot P(\text{Legs} = 2 \mid M) \cdot P(\text{Height} = \text{Tall} \mid M) \cdot P(\text{Smelly} = \text{No} \mid M)$$

for H:

$$P(H \mid \text{New Instance}) = P(H) \cdot P(\text{Color} = \text{Green} \mid H) \cdot P(\text{Legs} = 2 \mid H) \cdot P(\text{Height} = \text{Tall} \mid H) \cdot P(\text{Smelly} = \text{No} \mid H)$$

Reference:

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

Reference Video: https://www.youtube.com/watch?v=z8K-598fqSo

Now:

Calculation of Prior probabilities:

$$P(M) = \frac{4}{8} = 0.5 \quad P(H) = \frac{4}{8} = 0.5$$

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

Calculation of Conditional probabilities:

Color	M	H
White	2/4	3/4
Green	2/4	1/4

Legs	M	H
2	1/4	4/4
3	3/4	0/4

Height	M	H
Tall	3/4	2/4
Short	1/4	2/4

Smelly	M	H
Yes	3/4	1/4
No	1/4	3/4

$$p(M|New\ Instance) = p(M) * p(Color = Green|M) * p(Legs = 2|M) * p(Height = tall|M) * p(Smelly = no\ |M)$$

$$p(M|New\ Instance) = 0.5 * \frac{2}{4} * \frac{1}{4} * \frac{3}{4} * \frac{1}{4} = 0.0117$$

$$p(H|New\ Instance) = p(H) * p(Color = Green|H) * p(Legs = 2|H) * p(Height = tall|H) * p(Smelly = no\ |H)$$

$$p(H|New\ Instance) = 0.5 * \frac{1}{4} * \frac{4}{4} * \frac{2}{4} * \frac{3}{4} = 0.047$$

$$p(H|New\ Instance) > p(M|New\ Instance)$$

Hence the new instance belongs to Speices H

Subscribe

Reference Video: https://www.youtube.com/watch?v=z8K-598fqSo

Classwork:

- **Estimate conditional probabilities of each attributes (color, height, smelly) for the species classes: (M, H) using the data given in the table. Using these probabilities, estimate the probability values for the new instance: *color=green, legs=2, height=tall and smelly= Yes*. Classify the species for this entity.**

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

Assignment:

1. **From the given dataset 1, using Naïve Bayes Classifier, find if a Car (with Red color, SUV type and Domestic origin) will fall in Stolen class or not.**
2. **Consider given training dataset 2: Check whether given person does cheat or no using Bayesian classifier. Refund (x, 'yes') ^ Marital status (x, Divorced) ^ Income (x, <80K). [BIM 2018, Group C]**
3. **Consider given training dataset 3: Check whether given student buys computer or not using Bayesian classifier.**

Test data: X:(Age=Youth, Income=Medium, Student=Yes, Credit_Rating=Fair)

Dataset 1

Example 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

Dataset 2

ID	Refund	Martial Status	Income	Cheat
1	Yes	Single	>80K	No
2	No	Married	>80K	No
3	No	Single	<80K	No
4	Yes	Married	>80K	No
5	No	Divorced	>80K	Yes
6	No	Married	<80K	No
7	Yes	Divorced	>80K	No
8	No	Single	>80K	Yes
9	No	Married	<80K	No
10	No	Single	>80K	Yes

Dataset 3

ID	Age	Income	Student	Credit_Rating	Buy_Computer?
1	Youth	High	No	Fair	No
2	Youth	High	No	Excellent	No
3	Middle_aged	High	No	Fair	Yes
4	Senior	Medium	No	Fair	Yes
5	Senior	Low	Yes	Fair	Yes
6	Senior	Low	Yes	Excellent	No
7	Middle_aged	Low	Yes	Excellent	Yes
8	Youth	Medium	No	Fair	No
9	Youth	Low	Yes	Fair	Yes
10	Senior	Medium	Yes	Fair	Yes
11	Youth	Medium	Yes	Excellent	Yes
12	Middle_aged	Medium	No	Excellent	Yes
13	Middle_aged	High	Yes	Fair	Yes
14	Senior	Medium	No	Excellent	No

Refererence:

1. https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html
2. https://www.youtube.com/watch?v=fOK9DiKUGYs&t=285s

Other Solved Examples:

- Refer
  - https://www.youtube.com/watch?v=z8K-598fqSo&t=201s
  - https://www.youtube.com/watch?v=fOK9DiKUGYs&t=285s
  - https://docplayer.net/41403217-Text-categorization-1.html
  - https://www.site.uottawa.ca/~diana/csi4107/example_classif_clustering_sol.pdf

4.6 Ensemble Method: Random Forest

- An **ensemble method** is a way to combine multiple models to make better predictions. Instead of relying on one model, it uses several models and combines their results to make a stronger, more accurate prediction.
- ***Analogy:*** Think of it like asking multiple people for their opinion and making a decision based on the majority or average of their answers.

  Example
  - Imagine you're deciding whether to bring an umbrella:
    1. One friend says, "Yes, it will rain."
    2. Another says, "No, it won't rain."
    3. A weather app says, "80% chance of rain."
  - If you combine their opinions (e.g., by voting or averaging), you'll make a better decision than relying on just one.

Types of Ensemble Methods

1. Bagging (Bootstrap Aggregating)

- Train many models on different random samples of the data.
- Combine their predictions (e.g., take the average for numbers or the majority vote for categories).
- **Example**: Random Forest.
- **Goal**: Reduce mistakes caused by random changes in the training data.

2. Boosting

- Train models one after another, where each model tries to fix the errors of the previous one.
- Focus more on difficult cases that earlier models got wrong.
- **Examples**: AdaBoost, Gradient Boosting, XGBoost.
- **Goal**: Improve accuracy by focusing on tough examples.

3. Stacking

- Use different types of models (e.g., decision trees, SVM, neural networks).
- Combine their predictions using another model, called a "meta-model," to make the final decision.
- **Goal**: Use the strengths of different models together.

Advantages and Disadvantages

Advantages of Ensemble Methods

- **More Accurate**: They often perform better than a single model.
- **Less Risk**: Combining models reduces the chance of big mistakes.
- **Flexible**: They can mix different types of models.

Disadvantages

- **Slower**: Training and predicting with multiple models takes more time.
- **Hard to Understand**: It’s not easy to explain how the combined model works.

Random Forest

- **Random Forest** is an **ensemble method** that combines many decision trees to make better predictions. It is used for both **classification** (predicting categories) and **regression** (predicting numbers).

How Random Forest Works?

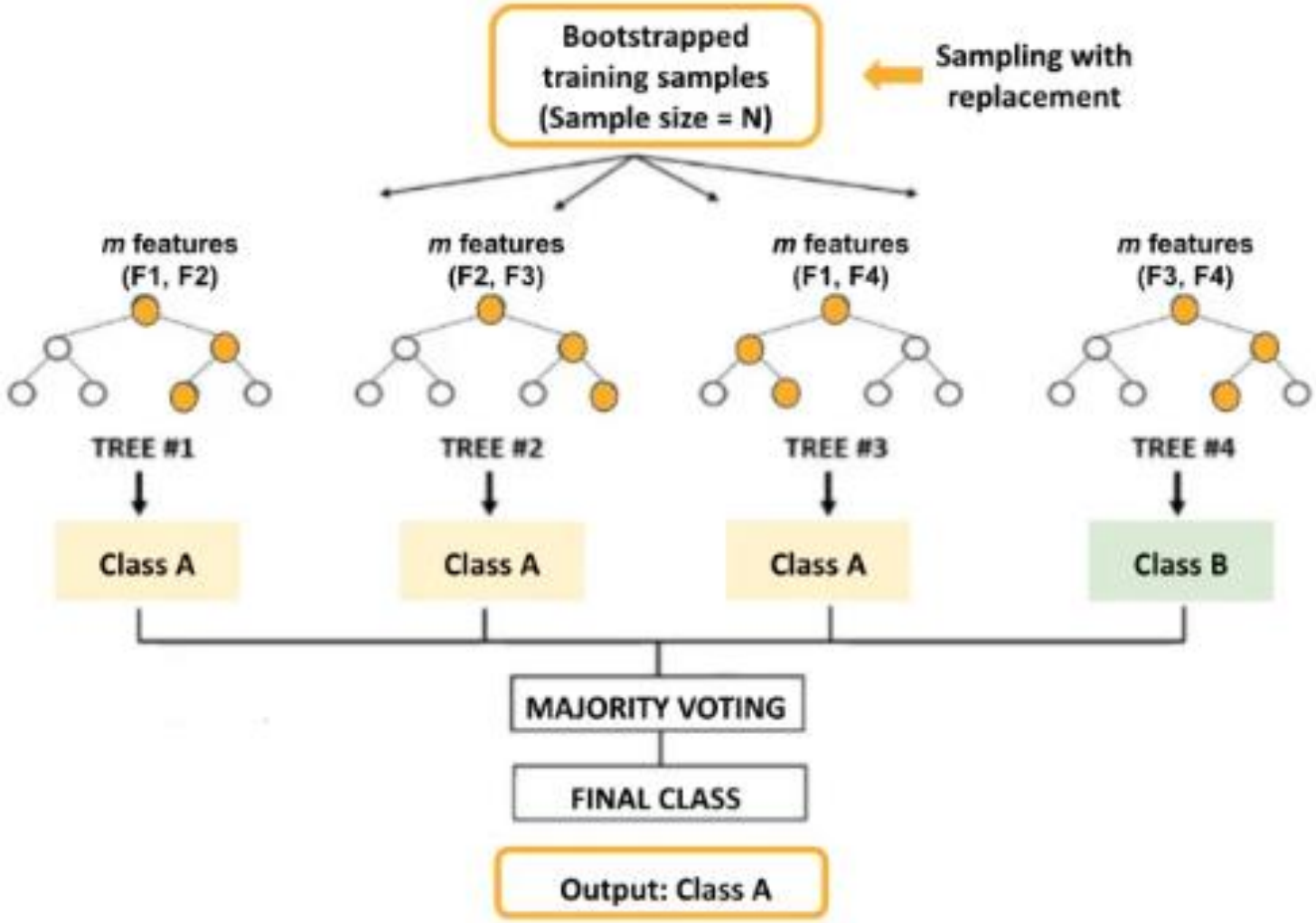
1. **Build Multiple Trees**: Random Forest creates many decision trees using random subsets of data and features.
2. **Combine Predictions**:
   - For **classification**, it takes a majority vote from all trees.
   - For **regression**, it averages the predictions.
3. **Adds Randomness**: Each tree is different because of random data sampling and random feature selection.

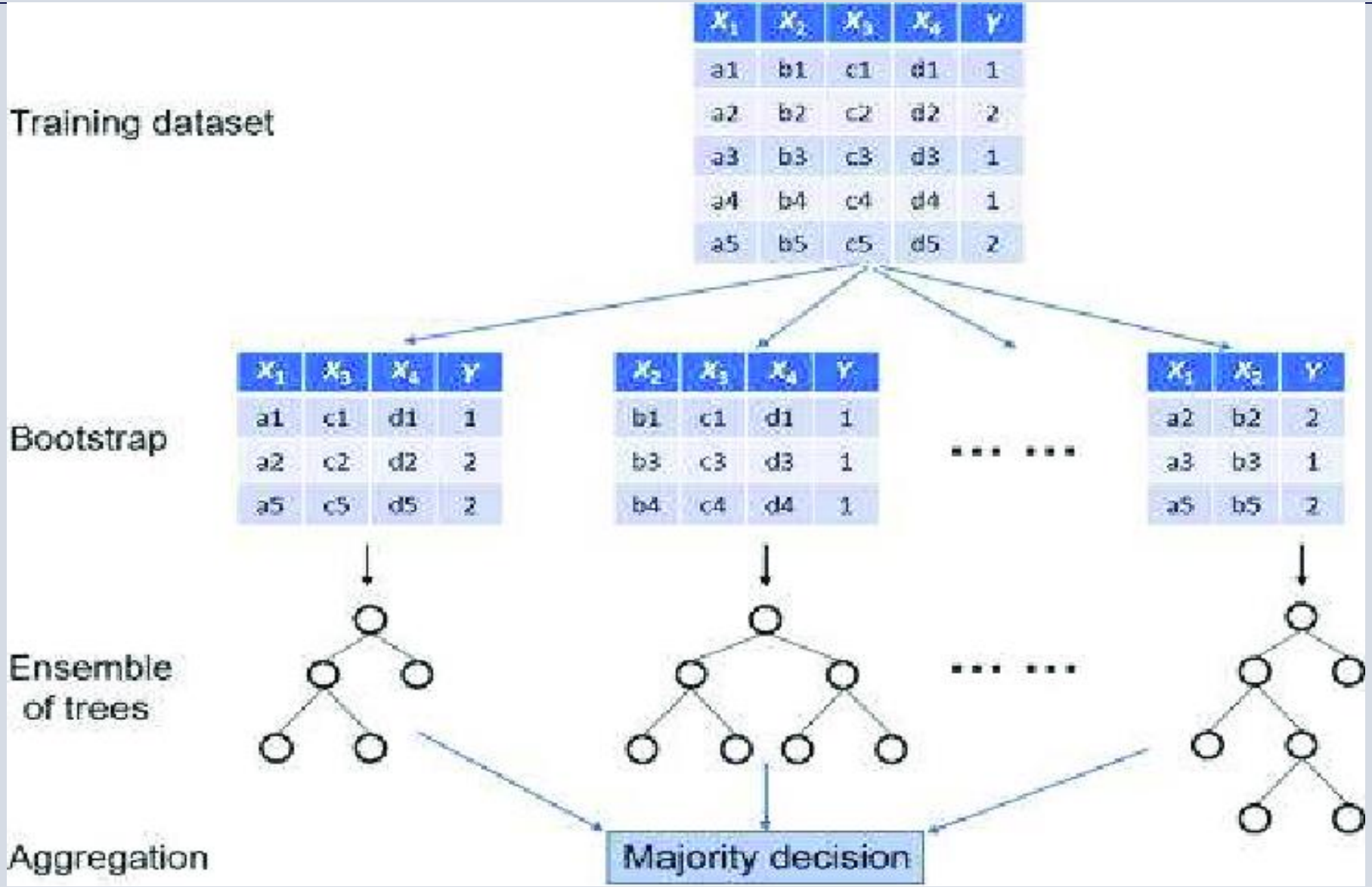
- This makes the model more accurate and less prone to overfitting.

Random Forest Classifier

Training Data
(Sample size, N= 6, No. of features, F = 4)

F1	F2	F3	F4	Y
2.1	0	400	-9	A
3.0	1	890	-42	B
2.2	1	929	0	B
4.0	0	324	-23	A
3.5	1	333	-15	A
6.0	0	215	-9	A





X_1	X_2	X_3	X_4	Y
a1	b1	c1	d1	1
a2	b2	c2	d2	2
a3	b3	c3	d3	1
a4	b4	c4	d4	1
a5	b5	c5	d5	2

X_1	X_3	X_4	Y
a1	c1	d1	1
a2	c2	d2	2
a5	c5	d5	2

X_2	X_3	X_4	Y
b1	c1	d1	1
b3	c3	d3	1
b4	c4	d4	1

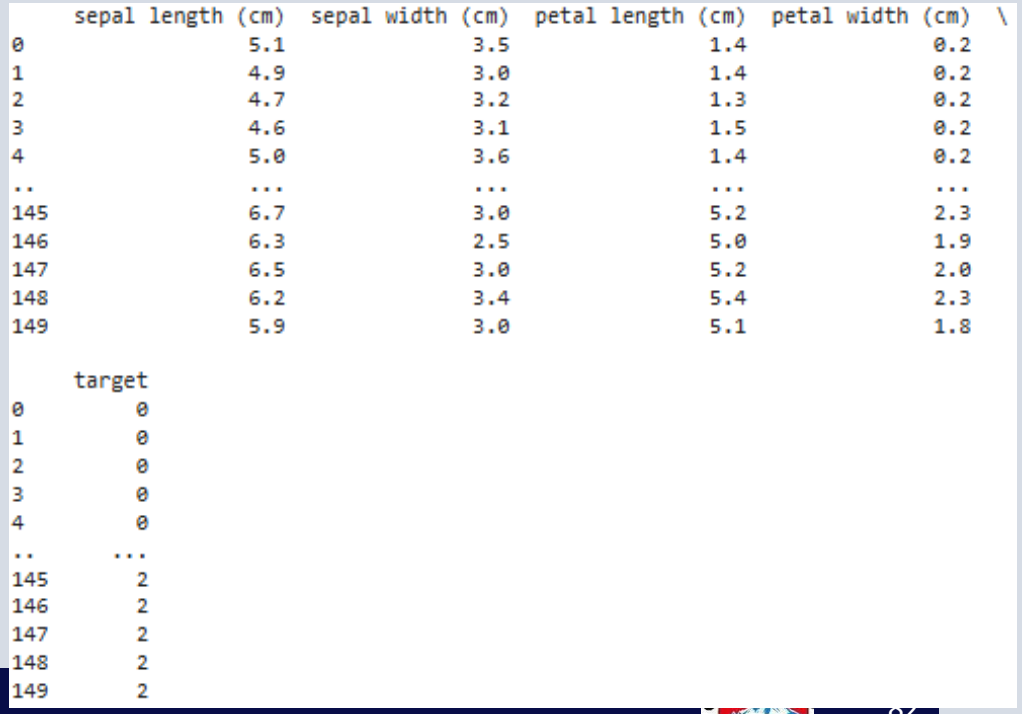
X_1	X_2	Y
a2	b2	2
a3	b3	1
a5	b5	2

Example: Classification with Random Forest Classifier

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

# Step 1: Load the Iris dataset directly as a DataFrame
iris = load_iris(as_frame=True)

# Access the complete DataFrame
df = iris.frame
print(df)
```



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

```
# Features Engineering

X = df[iris.feature_names]  # Features: sepal length, sepal width, petal
length, petal width
y = df['target']  # Target (Flower species)(0: Setosa, 1, 2: Virginica)


# Step 2: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Step 3: Create and train the Random Forest model
model = RandomForestClassifier(n_estimators=10, random_state=42)  # Use 10 trees
model.fit(X_train, y_train)

# Step 4: Make predictions on the test set
y_pred = model.predict(X_test)
```

```python
# Step 5: Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Step 6: Test prediction on new data
new_sample = [[5.1, 3.5, 5.4, 5.2]]  # Example data
prediction = model.predict(new_sample)
print("Predicted class for new sample:", prediction)
print(f"Predicted class for new sample: {data.target_names[prediction][0]}")
```

```
Accuracy: 100.00%
Predicted class for new sample: [2]
Predicted class for new sample: virginica
```

Iris dataset

- The **Iris dataset** is one of the most famous and widely used datasets in machine learning and statistics.
- It is a small dataset commonly used for classification and clustering tasks, especially when learning about machine learning algorithms.
- It contains **150 samples** of iris flowers. The target class is ***Species***: Setosa, Versocolor, Virginica labeled as 0,1 and 2 respectively.

Dataset Example:

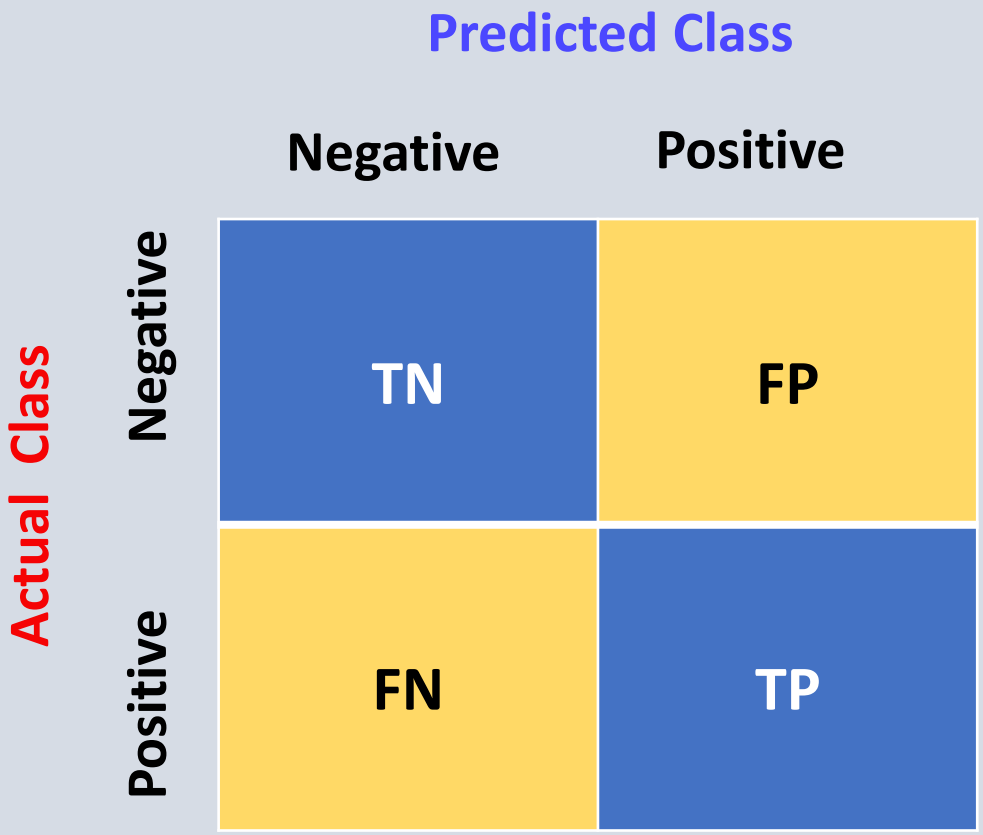
Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
6.3	3.3	6.0	2.5	Virginica

4.7 Model Evaluation

- When evaluating the performance of classification models, several metrics are derived from the **Confusion Matrix**, a table that summarizes the performance of a classifier on a set of test data for which the true labels are known.
- From the confusion matrix, we can derive the following key metrics:
  - Accuracy
  - Recall
  - Precision
  - F1 Score

Confusion Matrix

- The Confusion Matrix is a 2x2 table (for binary classification) that show the following:
  1. **True Positive (TP)**: The count of positive examples that were correctly predicted.
  2. **True Negative (TN)**: The count of negative examples that were correctly predicted.
  3. **False Positive(FP):** The count of negative examples that were incorrectly predicted as Positive. *(i.e actually negative but predicted positive)*. (Type I error).
  4. **False Negative (FN)**: The count of positive examples that were incorrectly predicted as Negative. *(i.e actually positive but predicted negative).(*Type II error).



Evaluation Matrix

1. Accuracy

- **Definition**: The ratio of correctly predicted instances to the total instances.
- **Use Case**: Useful when the classes are balanced. Can be misleading if classes are imbalanced.
- **Formula**:

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

2. Recall (Sensitivity or True Positive Rate)

- **Definition**: The proportion of actual positives that were correctly identified.
- **Use Case**: Important in cases where identifying positives is crucial (e.g., disease detection).
- **Formula**:

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

3. Precision

- **Definition**: The proportion of positive predictions that were actually correct.
- **Use Case**: Important when false positives are costly (e.g., spam email detection).
- **Formula:**

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

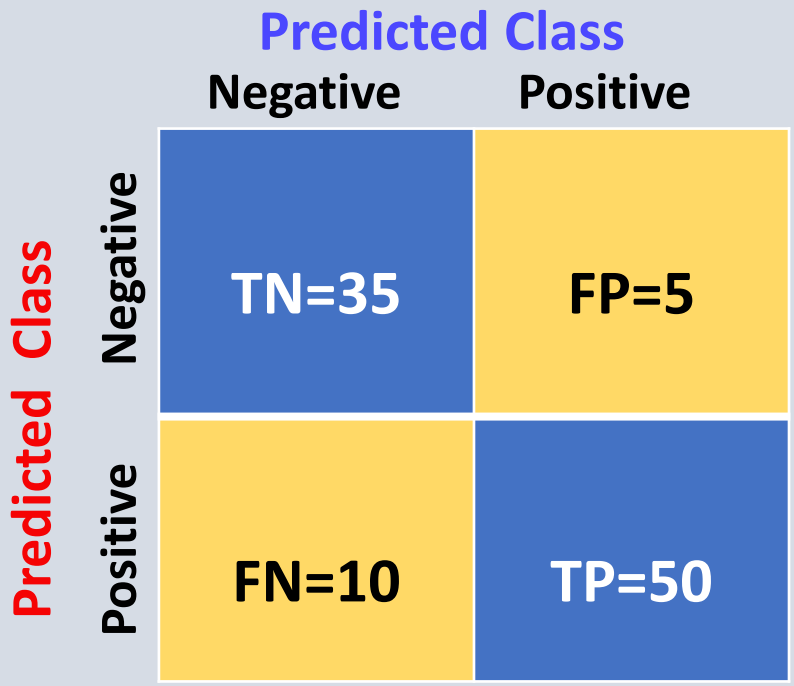
4. F1-Score

- **Definition**: The harmonic mean of precision and recall, balancing both metrics.
- **Use Case**: Useful when there is an imbalance between classes and when both precision and recall are important. Use when you need a balance between precision and recall.
- **Formula:**

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

Example:

- Let's consider an example of a binary classification model with the following confusion matrix:



Calculation:

- Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{50 + 35}{50 + 35 + 5 + 10} = 0.85 = 85\%$$

- Recall:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{50}{50 + 10} = 0.833 = 83.3\%$$

- Precision:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{50}{50 + 5} = 0.909 = 90.9\%$$

- F1-Score:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.909 \cdot 0.833}{0.909 + 0.833} = 0.87 = 87\%$$

Python Implementation

```
from sklearn.metrics import confusion_matrix

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Display results
print("Confusion Matrix:")
print(pd.DataFrame(conf_matrix,
                   index=iris.target_names,
                   columns=iris.target_names))
```

```
Confusion Matrix:
            setosa  versicolor  virginica
setosa          19           0          0
versicolor       0          13          0
virginica        0           0         13
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("\nEvaluation Metrics:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
```

```
Evaluation Metrics:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1-Score: 1.00
```

```
from sklearn.metrics import classification_report

# Display classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred,
target_names=iris.target_names))
```

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

End of Chapter 4