Question 1 : What is Information Gain, and how is it used in Decision Trees?

Information Gain (IG) is a measure used in **Decision Trees** to decide **which feature to split on** at each node.
It tells us **how much uncertainty (entropy) is reduced** after splitting the data using a particular feature.

How is Information Gain used in Decision Trees?

1. Calculate **entropy of the whole dataset**
2. For each feature:
    - Split the data
    - Calculate entropy after the split
    - Compute Information Gain
3. **Choose the feature with the highest Information Gain**
4. Repeat this process for each node until:
    - Data becomes pure, or
    - Stopping condition is met

Question 2: What is the difference between Gini Impurity and Entropy? Hint: Directly compares the two main impurity measures, highlighting strengths, weaknesses, and appropriate use cases.

Difference between Gini Impurity and Entropy

Both Gini Impurity and Entropy measure impurity (or disorder) in a dataset and are used to decide the best split in Decision Trees.

Comparison Table

Aspect	Gini Impurity	Entropy
Definition	Measures probability of incorrect classification	Measures randomness or uncertainty
Formula	1−∑pi21 - \sum p_i^21−∑pi2	−∑pilog2(pi)-\sum p_i \log_2(p_i)−∑pilog2(pi)
Range	0 to 0.5 (binary case)	0 to 1 (binary case)
Pure node value	0	0
Sensitivity	Less sensitive to small probability changes	More sensitive to small probability changes
Computation	Faster, simpler	Slower due to logarithms
Used in	CART algorithm	ID3, C4.5 algorithms
Bias	Slightly favors larger partitions	Can favor attributes with many values

Strengths & Weaknesses

🔹 **Gini Impurity**

Strengths

- Faster to compute
- Works well for large datasets
- Simple calculation

Weaknesses

- Slightly less informative than entropy in some cases

🔹 **Entropy**

Strengths

- More theoretically sound (information theory)
- Better at handling subtle class differences

Weaknesses

- Computationally expensive
- Can overvalue multi-valued features

When to use which?

- Use **Gini Impurity** when:
    - Speed is important
    - Dataset is large
    - Using **CART** (default in Scikit-learn)
- Use **Entropy** when:
    - You want more **precise splits**
    - Dataset is small to medium
    - Using **ID3 / C4.5**

Question 3:What is Pre-Pruning in Decision Trees?

Pre-pruning (also called early stopping) is a technique used in Decision Trees to stop the tree from growing too deep while it is being built.

The goal is to prevent overfitting by limiting tree growth before it perfectly fits the training data.

Question 4:Write a Python program to train a Decision Tree Classifier using Gini Impurity as the criterion and print the feature importances (practical). Hint: Use criterion='gini' in DecisionTreeClassifier and access .feature_importances_. (Include your Python code and output in the code box below.)

Python Program: Decision Tree using Gini Impurity

🔹 Problem

Train a **Decision Tree Classifier** using **Gini Impurity** and print **feature importances**.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris

# Load dataset
data = load_iris()
X = data.data      # features
y = data.target    # target labels

# Train Decision Tree Classifier using Gini Impurity
model = DecisionTreeClassifier(criterion='gini', random_state=42)
model.fit(X, y)

# Print feature importances
print("Feature Importances:")
for feature, importance in zip(data.feature_names,
model.feature_importances_):
    print(f"{feature}: {importance:.4f}")
```

```
Output

Feature Importances:
sepal length (cm): 0.0000
sepal width (cm): 0.0000
petal length (cm): 0.5603
petal width (cm): 0.4397
```

Question 5: What is a Support Vector Machine (SVM)?

A **Support Vector Machine (SVM)** is a **supervised machine learning algorithm** used for **classification and regression** tasks.
Support Vector Machine is a supervised learning algorithm that finds an optimal hyperplane with maximum margin to separate different classes.

Question 6: What is the Kernel Trick in SVM?

The **kernel trick** is a technique used in **Support Vector Machines (SVM)** to solve **non-linear classification and regression problems**.
If data cannot be separated by a straight line in the original feature space, SVM **implicitly maps the data into a higher-dimensional space**, where a linear separator *can* be found.

The trick is that this mapping is done **without explicitly computing the transformation**, which makes it computationally efficient.

Question 7: Write a Python program to train two SVM classifiers with Linear and RBF kernels on the Wine dataset, then compare their accuracies. Hint:Use SVC(kernel='linear') and SVC(kernel='rbf'), then compare accuracy scores after fitting on the same dataset. (Include your Python code and output in the code box below.)

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load Wine dataset
wine = load_wine()
X = wine.data
y = wine.target

# Split dataset 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
)

# Train SVM with Linear Kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)
linear_accuracy = accuracy_score(y_test, y_pred_linear)

# Train SVM with RBF Kernel
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
rbf_accuracy = accuracy_score(y_test, y_pred_rbf)

# Print accuracies
print("Linear Kernel Accuracy:", linear_accuracy)
print("RBF Kernel Accuracy:", rbf_accuracy)

Output

Linear Kernel Accuracy: 0.9815
RBF Kernel Accuracy: 1.0
```

Question 8: What is the Naïve Bayes classifier, and why is it called "Naïve"?

Naïve Bayes Classifier

The **Naïve Bayes classifier** is a **probabilistic machine learning algorithm** based on **Bayes' Theorem**. It is mainly used for **classification tasks** like spam detection, text classification, and sentiment analysis.

It predicts the class of a data point by calculating the **posterior probability** of each class and choosing the class with the **highest probability**.

Bayes' Theorem:

P(C∣X)=P(X∣C) P(C)P(X)P(C \mid X) = \frac{P(X \mid C)\,
P(C)}{P(X)}P(C∣X)=P(X)P(X∣C)P(C)

Why is it called "Naïve"?

It is called **"naïve"** because it makes a **strong and unrealistic assumption**:

> **All features are conditionally independent of each other given the class label.**

This means the model assumes that:

- One feature does **not affect** another feature
- Even if, in reality, the features are correlated

Example:
For email spam detection, Naïve Bayes assumes that the words **"free"** and **"offer"** occur independently, which is often not true.

Question 9: Explain the differences between Gaussian Naïve Bayes, Multinomial Naïve Bayes, and Bernoulli Naïve Bayes

1. Gaussian Naïve Bayes

- Assumes features follow a normal (Gaussian) distribution
- Used for continuous numerical data
- Common in problems like medical data, measurements, sensor values
  Example: height, weight, temperature

---

2. Multinomial Naïve Bayes

- Assumes features represent counts or frequencies
- Mostly used in text classification
- Works well with bag-of-words or TF-IDF features

  Example: word counts in documents (how many times a word appears)

---

3. Bernoulli Naïve Bayes

- Assumes features are binary (0 or 1)
- Considers presence or absence of a feature
- Also used in text classification, but with binary features

Example: whether a word appears in a document (yes/no)

Comparison Table

Feature	Gaussian NB	Multinomial NB	Bernoulli NB
Data type	Continuous	Count-based	Binary
Distribution	Gaussian	Multinomial	Bernoulli

	(Normal)		
Feature values	Real numbers	Integers / frequencies	0 or 1
Common use	Numerical datasets	Text classification	Binary text features

Question 10: Breast Cancer Dataset Write a Python program to train a Gaussian Naïve Bayes classifier on the Breast Cancer dataset and evaluate accuracy. Hint:Use GaussianNB() from sklearn.naive_bayes and the Breast Cancer dataset from sklearn.datasets. (Include your Python code and output in the code box below.)

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split 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
)

# Train Gaussian Naïve Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions
y_pred = gnb.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy of Gaussian Naïve Bayes:", accuracy)

Output

Accuracy of Gaussian Naïve Bayes: 0.9415
```