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

Answer:
Information Gain (IG) is a metric used to measure the effectiveness of an attribute in classifying data.
It is based on the concept of **entropy**, which measures the impurity or disorder of a dataset.

The formula for Information Gain is:

\`\`\` Information Gain = Entropy(Parent) - [Weighted Average] * Entropy(Children) \`\`\`

- A higher Information Gain means the attribute provides more information about the target class.
- Decision Trees use IG to decide which feature to split on at each node — choosing the one with the **highest IG**.

Question 2: What is the difference between Gini Impurity and Entropy?

Answer:

Metric	Formula	Range	Interpretation
Entropy	$(-p_1 \log_2 p_1 - p_2 \log_2 p_2 - ... - p_n \log_2 p_n)$	0–1	Measures impurity based on information theory
Gini Impurity	$(1 - \sum p_i^2)$	0–0.5	Measures how often a randomly chosen element is incorrectly labeled

Key Differences:

- Entropy involves logarithmic computation (slower but information-theoretic).
- Gini is simpler and faster to compute.
- Both yield similar trees, but Gini often performs better in practice for large datasets.

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

Answer:
Pre-Pruning (also called **Early Stopping**) prevents a Decision Tree from becoming

too complex.
Instead of fully growing the tree and pruning later, it stops the growth early when further splits don't significantly improve performance.

Common Pre-Pruning Criteria:

- Maximum depth (`max_depth`)
- Minimum samples per leaf (`min_samples_leaf`)
- Minimum information gain threshold

This helps reduce **overfitting** and improves **generalization**.

Question 4: Write a Python program to train a Decision Tree Classifier using Gini Impurity as the criterion and print the feature importances.

In [ ]:
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import pandas as pd

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

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

# Feature importances
importances = pd.Series(clf.feature_importances_, index=iris.feature_names)
print("Feature Importances:")
print(importances.sort_values(ascending=False))
```

```
Feature Importances:
petal length (cm)    0.564056
petal width (cm)     0.422611
sepal length (cm)    0.013333
sepal width (cm)     0.000000
dtype: float64
```

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

Answer:
Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression.
It finds the **optimal hyperplane** that best separates data points of different classes with **maximum margin**.

Key concepts:

- **Support Vectors:** Data points closest to the decision boundary.
- **Margin:** Distance between the hyperplane and support vectors.
- **Goal:** Maximize the margin for better generalization.

Question 6: What is the Kernel Trick in SVM?

Answer:
The **Kernel Trick** allows SVM to perform nonlinear classification efficiently.
It maps data into a higher-dimensional space without explicitly computing the transformation.

Common kernels:

- **Linear:** Suitable for linearly separable data.
- **Polynomial:** Adds interaction features.
- **RBF (Radial Basis Function):** Handles complex, nonlinear boundaries.

Formula example for RBF:
(K(x, x') = \exp(-\gamma ||x - x'||^2))

Question 7: Write a Python program to train two SVM classifiers with Linear and RBF kernels on the Wine dataset, then compare their accuracies.

In [ ]:
```
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 dataset
wine = load_wine()
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, te

# Linear Kernel
svm_linear = SVC(kernel='linear', random_state=42)
svm_linear.fit(X_train, y_train)
linear_acc = accuracy_score(y_test, svm_linear.predict(X_test))

# RBF Kernel
svm_rbf = SVC(kernel='rbf', random_state=42)
svm_rbf.fit(X_train, y_train)
rbf_acc = accuracy_score(y_test, svm_rbf.predict(X_test))

print(f"Linear Kernel Accuracy: {linear_acc:.4f}")
print(f"RBF Kernel Accuracy: {rbf_acc:.4f}")
```

```
Linear Kernel Accuracy: 1.0000
RBF Kernel Accuracy: 0.8056
```

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

Answer:
Naïve Bayes is a **probabilistic classifier** based on **Bayes' Theorem**, assuming independence between features.

Formula:
(P(C|X) = \frac{P(X|C) * P(C)}{P(X)})

It’s called “Naïve” because it **assumes all features are independent**, which is rarely true in practice, but still performs surprisingly well for many problems.

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

Answer:

Type	Data Type	Use Case	Distribution
GaussianNB	Continuous	Real-valued features (e.g., Iris, Breast Cancer)	Normal distribution
MultinomialNB	Discrete	Text classification (word counts)	Multinomial distribution
BernoulliNB	Binary	Binary/boolean features (e.g., presence/absence of word)	Bernoulli distribution

Question 10: Write a Python program to train a Gaussian Naïve Bayes classifier on the Breast Cancer dataset and evaluate accuracy.

In [ ]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, te

# Train model
```

```python
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Evaluate
y_pred = gnb.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"GaussianNB Accuracy: {accuracy:.4f}")
```

```
GaussianNB Accuracy: 0.9737
```