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

Information Gain is a metric used to measure how much uncertainty (entropy) is reduced after splitting a dataset on a particular feature. In decision trees, it helps determine the best feature to split the data at each node. The feature that provides the highest information gain is selected because it creates the most homogeneous child nodes.

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

Gini Impurity measures the probability of incorrectly classifying a randomly chosen data point, while Entropy measures the level of disorder in the dataset. Gini is computationally faster and commonly used in CART trees, whereas Entropy is more theoretically grounded in information theory. Both aim to create purer nodes, but often give similar results in practice.

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

Pre-pruning is a technique used to prevent overfitting in decision trees by stopping the tree growth early. This is done by setting constraints such as maximum tree depth, minimum samples per split, or minimum information gain required for a split.

Question 4: Decision Tree using Gini Impurity (Python Program)

The following code trains a Decision Tree Classifier using Gini Impurity and prints feature importances:

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

data = load_iris()
X, y = data.data, data.target

model = DecisionTreeClassifier(criterion='gini')
model.fit(X, y)

print("Feature Importances:", model.feature_importances_)
```

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

Support Vector Machine is a supervised learning algorithm used for classification and regression. It works by finding an optimal hyperplane that best separates data points of different classes with maximum margin.

Question 6: What is the Kernel Trick in SVM?

The Kernel Trick allows SVMs to solve non-linear classification problems by transforming data into a higher-dimensional space without explicitly computing the transformation. Common kernels include linear, polynomial, and RBF.

Question 7: SVM with Linear and RBF Kernels (Python Program)

The following code trains two SVM models and compares their accuracies:

```
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

X, y = load_wine(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

linear_svm = SVC(kernel='linear')
rbf_svm = SVC(kernel='rbf')
```

```
linear_svm.fit(X_train, y_train)
rbf_svm.fit(X_train, y_train)

print("Linear SVM Accuracy:", accuracy_score(y_test, linear_svm.predict(X_test)))
print("RBF SVM Accuracy:", accuracy_score(y_test, rbf_svm.predict(X_test)))
```

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

Naïve Bayes is a probabilistic classifier based on Bayes' Theorem. It is called 'Naïve' because it assumes that all features are independent of each other, which is rarely true in real-world data.

Question 9: Differences between Gaussian, Multinomial, and Bernoulli Naïve Bayes

Gaussian Naïve Bayes is used for continuous data assuming a normal distribution. Multinomial Naïve Bayes is suitable for discrete count data like word frequencies. Bernoulli Naïve Bayes works with binary features indicating presence or absence.

Question 10: Gaussian Naïve Bayes on Breast Cancer Dataset (Python Program)

The following code trains a Gaussian Naïve Bayes classifier and evaluates accuracy:

```
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

X, y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = GaussianNB()
model.fit(X_train, y_train)

print("Accuracy:", accuracy_score(y_test, model.predict(X_test)))
```