Supervised Classification: Decision Trees, SVM, and Naive Bayes | Assignment Answers

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

Information Gain measures how much 'information' a feature gives us about the class label. It is based on the concept of entropy from information theory, which measures impurity or uncertainty in data. When building a Decision Tree, Information Gain helps decide which feature to split on. The feature with the highest Information Gain (i.e., reduces entropy the most) is chosen.

Formula: Information Gain = Entropy(Parent) - Σ (n■/n × Entropy(Child■))

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

Gini Impurity measures how often a randomly chosen element would be incorrectly classified, while Entropy measures the average amount of information in the dataset.

Gini = 1 - Σ p■²
Entropy = -Σ p■ log■(p■)

Gini is faster to compute (no log) and used in CART; Entropy is used in ID3 and C4.5. Lower values of both indicate purer nodes.

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

Pre-pruning (early stopping) prevents overfitting by stopping tree growth early. Splitting stops when Information Gain is too small, sample size is low, or depth exceeds a limit.

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

Code:

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

iris = load_iris()
X, y = iris.data, iris.target
clf = DecisionTreeClassifier(criterion='gini', random_state=42)
clf.fit(X, y)
```

```
print('Feature Importances:')
for name, imp in zip(iris.feature_names, clf.feature_importances_):
print(f'{name}: {imp:.4f}')
```

Sample Output:

```
sepal length (cm): 0.0203
sepal width (cm): 0.0000
petal length (cm): 0.5632
petal width (cm): 0.4165
```

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

SVM is a supervised learning algorithm that finds the best hyperplane separating classes with the maximum margin. Data points closest to this boundary are called support vectors. It performs well in high-dimensional spaces.

Question 6: What is the Kernel Trick in SVM?

The Kernel Trick allows SVM to classify non-linear data by mapping it into a higher-dimensional space. Common kernels: Linear, Polynomial, RBF, and Sigmoid. It helps create a linear boundary in transformed space.

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

Code:

```
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

wine = load_wine()
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3,
random_state=42)

svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
acc_linear = accuracy_score(y_test, svm_linear.predict(X_test))
```

```
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
acc_rbf = accuracy_score(y_test, svm_rbf.predict(X_test))

print(f'Linear Kernel Accuracy: {acc_linear:.4f}')
print(f'RBF Kernel Accuracy: {acc_rbf:.4f}')
```

Sample Output:

```
Linear Kernel Accuracy: 0.9815
RBF Kernel Accuracy: 0.9630
```

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

The Naïve Bayes classifier uses Bayes’ Theorem for classification. It assumes all features are independent given the class, which makes it 'naïve'. Despite this, it performs well for text classification, spam detection, etc.

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

GaussianNB: for continuous data (assumes normal distribution).
MultinomialNB: for discrete counts (used in text classification).
BernoulliNB: for binary features (presence/absence of words).

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

Code:

```
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

data = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3,
random_state=42)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
```

Sample Output:

```
Accuracy: 0.9591
```