Assignment of Decision Tree

Theory questions

1. What is a Decision Tree, and how does it work?

A **Decision Tree** is like a flowchart used in machine learning for both classification and regression tasks.

How it works:

- It asks questions about features (like: "Is age > 30?")
- Based on the answer (Yes/No), it splits the data
- This continues until it reaches a final answer (leaf node)

2. What are impurity measures in Decision Trees?

Impurity measures are used to decide:

Where to split the tree to get the most "pure" groups (mostly same class).

Two common impurity measures:

- **Gini Impurity**
- **Entropy (used for Information Gain)**

Lower impurity = purer node = better split.

3. What is the mathematical formula for Gini Impurity?

$$Gini = 1 - \sum pi^2$$

Where:

- pip_ipi is the probability of class iii

Example: If 3 out of 5 are "Yes" and 2 are "No":

- Pyes = 3/5 , Pno = 2/5
- Gini = 1−[(3/5)2+(2/5)2] = 1−(0.36+0.16) = 0.48

4. What is the mathematical formula for Entropy?

$$\text{Entropy} = -\sum p_i * \log(p_i)$$

Example: Same as above (3 Yes, 2 No):

$$\text{Entropy} = -[(3/5)\log(3/5) + (2/5)\log(2/5)] \approx 0.971$$

Higher entropy = more disorder.

5. What is Information Gain, and how is it used in Decision Trees?

Information Gain (IG) measures how much entropy is reduced after a split.

$$IG = \text{Entropy(parent)} - \text{Weighted avg of child entropies}$$

- Used to choose the best feature to split on
- Higher IG = better feature

6. What is the difference between Gini Impurity and Entropy?

It is the probability of misclassifying a randomly chosen element in a set.	While entropy measures the amount of uncertainty or randomness in a set.
The range of the Gini index is **[0, 0.5]**, where 0 indicates perfect purity and 0.5 indicates maximum impurity.	The range of entropy is **[0, log2(C)]**, where c is the number of classes. The range becomes **[0, 1]** for binary classification.
Gini index is a linear measure.	Entropy is a logarithmic measure.
It can be interpreted as the expected error rate in a classifier.	It can be interpreted as the average amount of information needed to specify the class of an instance.
It is sensitive to the distribution of classes in a set.	It is sensitive to the number of classes.

7. What is the mathematical explanation behind Decision Trees?

Decision Trees are built using:

- Recursive Binary Splitting
- At each node:
    - Calculate impurity (Gini/Entropy)
    - Split on feature that reduces impurity the most
    - Repeat until stopping condition

Formally:

- Optimize the impurity function
- Choose splits that minimize cost

8. What is Pre-Pruning in Decision Trees?

Pre-pruning = Stop tree before it becomes too complex

Done using:

- max_depth
- min_samples_split
- min_samples_leaf

Example:

Stop splitting if node has < 5 samples

9. What is Post-Pruning in Decision Trees?

Post-pruning = First grow the full tree, then cut back unnecessary branches

Usually done using:

- Cost Complexity Pruning (CCP)
- Cross-validation

Benefit:

- More accurate
- Reduces overfitting

10. What is the difference between Pre-Pruning and Post-Pruning?

Post-Pruning is used generally for small datasets whereas Pre-Pruning is used for larger ones. Pre-Pruning is considered more efficient and effective as it considered multiple parameters and choose best ones from them.

11. What is a Decision Tree Regressor?

A Decision Tree Regressor predicts continuous values instead of categories.

Example: Predict house price based on:

- Size
- Location
- Rooms

Splits are based on minimizing:

Mean Squared Error (MSE) or Variance

12. What are the advantages and disadvantages of Decision Trees?

✅ **Advantages:**

- Easy to understand and visualize
- No need for feature scaling
- Handles both numerical and categorical data

❌ **Disadvantages:**

- Prone to overfitting
- Sensitive to small data changes
- Biased toward features with many levels

13. How does a Decision Tree handle missing values?

- Some algorithms use surrogate splits (use another feature if one is missing)

- Others fill missing values with:
    - Mean (numerical)
    - Most frequent (categorical)

In sklearn, you usually fill missing values before training using SimpleImputer.

14. How does a Decision Tree handle categorical features?

In **scikit-learn**, you must convert categorical features using:

- One-hot encoding (pd.get_dummies())
- Label encoding (LabelEncoder)

Some libraries like XGBoost or CatBoost handle categories natively.

15. What are some real-world applications of Decision Trees?

- ✅ Medical diagnosis (Is this tumor cancerous?)
- ✅ Credit risk assessment (Should we give this person a loan?)
- ✅ Customer churn prediction (Will this customer cancel?)
- ✅ Fraud detection (Is this transaction suspicious?)
- ✅ Product recommendation (Which product to suggest?)

Practical Questions

0.1 16) Write a Python program to train a Decision Tree Classifier on the Iris dataset and print the model accuracy

[2]:
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,␣
 ↪test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 1.0
```

0.2 17) Write a Python program to train a Decision Tree Classifier using Gini Impurity as the criterion and print the feature importances

[4]:
```
clf = DecisionTreeClassifier(criterion='gini', random_state=0)
clf.fit(X_train, y_train)

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

```
Feature Importances:
sepal length (cm): 0.0000
sepal width (cm): 0.0167
petal length (cm): 0.4059
petal width (cm): 0.5774
```

0.3 18) Write a Python program to train a Decision Tree Classifier using Entropy as the splitting criterion and print the model accuracy

[6]:
```
clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy (Entropy):", accuracy_score(y_test, y_pred))
```

```
Accuracy (Entropy): 1.0
```

0.4 19) Write a Python program to train a Decision Tree Regressor on a housing dataset and evaluate using Mean Squared Error (MSE)

[8]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

housing = fetch_california_housing()
X_train, X_test, y_train, y_test = train_test_split(housing.data, housing.
 ↪target, test_size=0.2, random_state=0)

reg = DecisionTreeRegressor()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)

print("MSE:", mean_squared_error(y_test, y_pred))
```

```
MSE: 0.5418481433277373
```

0.5 20) Write a Python program to train a Decision Tree Classifier and visualize the tree using graphviz

[10]:
```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
import graphviz

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
class_names = iris.target_names

# Split data into training and test sets
```

```
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(max_depth=3, random_state=0)
clf.fit(X_train, y_train)

# Export the tree to DOT format
dot_data = export_graphviz(
    clf,
    out_file=None,
    feature_names=feature_names,
    class_names=class_names,
    filled=True,
    rounded=True,
    special_characters=True
)

# Create and render the tree using graphviz
graph = graphviz.Source(dot_data)
graph.render("iris_decision_tree", format="pdf")  # Saves as iris_decision_tree.
 ↪pdf
graph.view()  # Opens the file
```

[10]: 'iris_decision_tree.pdf'

0.6 21)Write a Python program to train a Decision Tree Classifier with a maximum depth of 3 and compare its accuracy with a fully grown tree

[12]:
```
clf1 = DecisionTreeClassifier(max_depth=3)
clf2 = DecisionTreeClassifier()

clf1.fit(X_train, y_train)
clf2.fit(X_train, y_train)

print("Accuracy (max_depth=3):", accuracy_score(y_test, clf1.predict(X_test)))
print("Accuracy (full tree):", accuracy_score(y_test, clf2.predict(X_test)))
```

```
Accuracy (max_depth=3): 1.0
Accuracy (full tree): 1.0
```

0.7 22) Write a Python program to train a Decision Tree Classifier using min_samples_split=5 and compare its accuracy with a default tree

[14]:
```
clf1 = DecisionTreeClassifier(min_samples_split=5)
clf2 = DecisionTreeClassifier()
```

```
clf1.fit(X_train, y_train)
clf2.fit(X_train, y_train)

print("Accuracy (min_samples_split=5):", accuracy_score(y_test, clf1.
 ↪predict(X_test)))
print("Accuracy (default):", accuracy_score(y_test, clf2.predict(X_test)))
```

```
Accuracy (min_samples_split=5): 1.0
Accuracy (default): 1.0
```

0.8 23) Write a Python program to apply feature scaling before training a Decision Tree Classifier and compare its accuracy with unscaled dataasplit=5)

[16]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

clf_unscaled = DecisionTreeClassifier()
clf_scaled = DecisionTreeClassifier()

clf_unscaled.fit(X_train, y_train)
clf_scaled.fit(X_train_scaled, y_train)

print("Unscaled Accuracy:", accuracy_score(y_test, clf_unscaled.
 ↪predict(X_test)))
print("Scaled Accuracy:", accuracy_score(y_test, clf_scaled.
 ↪predict(X_test_scaled)))
```

```
Unscaled Accuracy: 1.0
Scaled Accuracy: 1.0
```

0.9 24) Write a Python program to train a Decision Tree Classifier using One-vs-Rest (OvR) strategy for multiclass classification

[18]:
```
from sklearn.multiclass import OneVsRestClassifier

ovr = OneVsRestClassifier(DecisionTreeClassifier())
ovr.fit(X_train, y_train)

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

```
OvR Accuracy: 1.0
```

0.10 25) Write a Python program to train a Decision Tree Classifier and display the feature importance scores

[20]:
```
print("Feature Importance:")
for name, importance in zip(iris.feature_names, clf.feature_importances_):
    print(f"{name}: {importance:.2f}")
```

```
Feature Importance:
sepal length (cm): 0.00
sepal width (cm): 0.00
petal length (cm): 0.39
petal width (cm): 0.61
```

0.11 26) Write a Python program to train a Decision Tree Regressor with max_depth=5 and compare its performance with an unrestricted tree

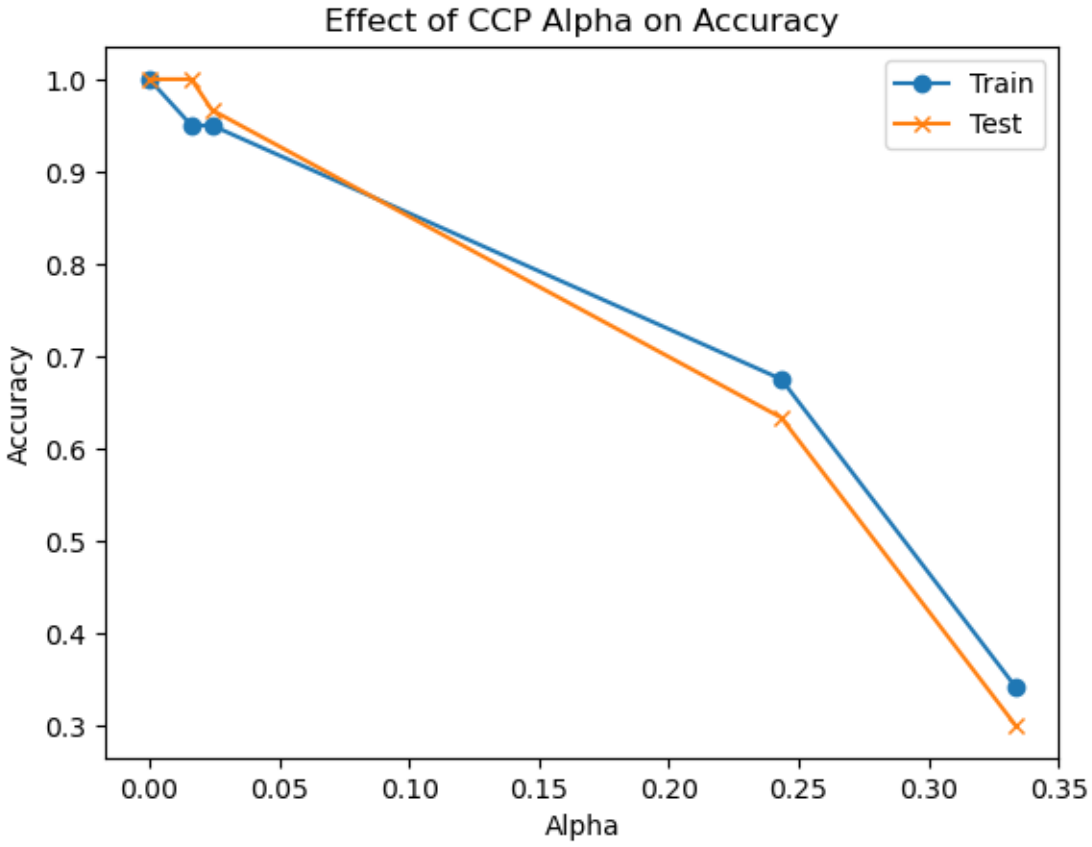
[22]:
```
reg1 = DecisionTreeRegressor(max_depth=5)
reg2 = DecisionTreeRegressor()

reg1.fit(X_train, y_train)
reg2.fit(X_train, y_train)

print("MSE (max_depth=5):", mean_squared_error(y_test, reg1.predict(X_test)))
print("MSE (full tree):", mean_squared_error(y_test, reg2.predict(X_test)))
```

```
MSE (max_depth=5): 0.0
MSE (full tree): 0.0
```

0.12 27) Write a Python program to train a Decision Tree Classifier, apply Cost Complexity Pruning (CCP), and visualize its effect on accuracy

[24]:
```
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

import matplotlib.pyplot as plt

train_scores = []
test_scores = []

for alpha in ccp_alphas:
    tree = DecisionTreeClassifier(ccp_alpha=alpha)
    tree.fit(X_train, y_train)
    train_scores.append(tree.score(X_train, y_train))
    test_scores.append(tree.score(X_test, y_test))

plt.figure()
plt.plot(ccp_alphas, train_scores, marker='o', label='Train')
plt.plot(ccp_alphas, test_scores, marker='x', label='Test')
```

```
plt.xlabel("Alpha")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Effect of CCP Alpha on Accuracy")
plt.show()
```



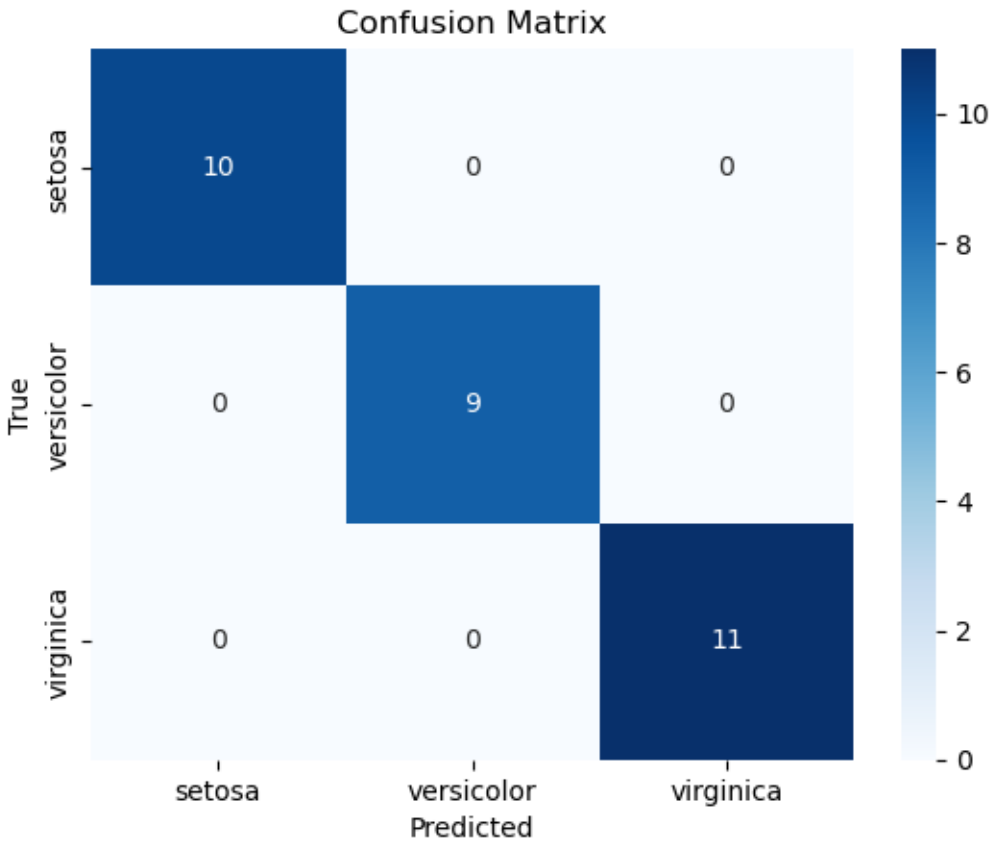
0.13 28) Write a Python program to train a Decision Tree Classifier and evaluate its performance using Precision, Recall, and F1-Score

[26]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

y_pred = clf.predict(X_test)
print("Precision:", precision_score(y_test, y_pred, average='macro'))
print("Recall:", recall_score(y_test, y_pred, average='macro'))
print("F1 Score:", f1_score(y_test, y_pred, average='macro'))
```

```
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

0.14 29) Write a Python program to train a Decision Tree Classifier and visualize the confusion matrix using seaborn

[28]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, cmap="Blues", xticklabels=iris.
 ↪target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```



0.15 30) Write a Python program to train a Decision Tree Classifier and use GridSearchCV to find the optimal values for max_depth and min_samples_split.

[30]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': [2, 3, 4, 5, None],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best Accuracy:", grid_search.best_score_)
```

```
Best Parameters: {'max_depth': 4, 'min_samples_split': 2}
Best Accuracy: 0.9416666666666668
```