INTERVIEW QUESTIONS

1.How does a decision tree work?

2.What is entropy and information gain?

3.How is random forest better than a single tree?

4.What is overfitting and how do you prevent it?

5.What is bagging?

6.How do you visualize a decision tree?

7.How do you interpret feature importance?

8.What are the pros/cons of random forests?

1.How does a decision tree work?

A **Decision Tree** is a flowchart-like structure used for classification or regression. It works by **recursively splitting** the dataset into smaller, purer groups based on feature values.

Step-by-Step Process

1. **Start at the Root Node**
   The algorithm looks at all features and finds the split (feature + threshold) that best separates the data.
   Example: "Is age ≤ 50?"
2. **Splitting Criterion**
   - For classification: use **Gini impurity** or **Entropy** to measure node purity.
   - For regression: use **Mean Squared Error (MSE)**.
3. **Recursive Partitioning**
   The process repeats for each child node, splitting until:
   - Nodes are pure (all samples belong to the same class)

- Or stopping conditions are met (max_depth, min_samples_split, etc.)

4. **Leaf Nodes**
    - Contain the final prediction.
    - In classification: the majority class in that node.
    - In regression: the mean of target values in that node.

Example Diagram

Here's a simplified decision tree for **Heart Disease Prediction**:

yaml Copy Edit

```
                [ChestPainType <= 2.5]
                  /                \
            Yes /                   \ No
              /                      \
   [MaxHeartRate > 150]          [Age <= 55]
      /          \                  /     \
    Yes           No              Yes      No
Disease=No   Disease=Yes     Disease=No  Disease=Yes
```

- At each branch, the algorithm asks a **yes/no** question.
- The leaves (bottom) show predictions.

2. **What is entropy and information gain?**

Entropy

- **Definition**: A measure of impurity or randomness in a dataset.
- If all samples in a node belong to one class → **Entropy = 0** (pure).
- Formula:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

where:

- c = number of classes
- p_i = proportion of samples belonging to class i

Example:

If a node has 8 "Yes" and 2 "No" samples:

$$p_{Yes} = 0.8, \quad p_{No} = 0.2$$

$$Entropy = -(0.8 \log_2 0.8 + 0.2 \log_2 0.2) \approx 0.72$$

Information Gain (IG)

- **Definition**: The reduction in entropy after splitting a dataset on a feature.
- Formula:

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where:

- S = parent node dataset
- S_v = subset for feature value v
- $|S_v|/|S|$ = proportion of samples in the subset

Idea:

- High IG means the split reduces uncertainty the most.
- Decision trees choose the **feature with the highest IG** at each step.

Visual Intuition

mathematica

```
[Parent Node]
Entropy = 0.96 (mixed classes)

Split on Feature X
    ↓
[Left Node] Entropy = 0.0 (pure)
[Right Node] Entropy = 0.65 (less pure)

IG = 0.96 - (weighted avg. of 0.0 and 0.65)
```

- Higher IG → better split.

3.How is random forest better than a single tree?

1. Single Decision Tree

- **Strengths**:
  - Easy to interpret.
  - Fast to train.
- **Weaknesses**:
  - **High variance** — small changes in the data can produce a very different tree.
  - **Overfitting risk** — may memorize noise instead of learning patterns.
  - Performance can be unstable.

---

2. Random Forest

- **How it works**:
  - Builds **many decision trees** on different bootstrapped samples of the training data.
  - At each split, considers **a random subset of features**.
  - Predictions are made by **majority vote** (classification) or **average** (regression).

- **Advantages over a single tree**:

1. **Reduces overfitting** — averaging many trees smooths out noise.
2. **More stable predictions** — less sensitive to small data changes.
3. **Better accuracy** — combines strengths of multiple weak learners.
4. **Handles high-dimensional data** well.

- **Trade-offs**:
    - Less interpretable.
    - More computationally expensive.

Analogy

- **Single Tree** = one person's opinion.
- **Random Forest** = opinions from a large, diverse group — less likely to be biased by one wrong perspective.

4.What is overfitting and how do you prevent it?

Definition

Overfitting happens when a model learns **not only the general patterns** in the training data but also **the noise and random fluctuations**.

- Result: Very high training accuracy ✅
- Poor performance on unseen/test data ❌

In Decision Trees

- If a tree grows too deep, it might perfectly classify training samples (even outliers), but it won't generalize well.
- Example:
- Training Accuracy: 100%
- Test Accuracy: 68%

→ Signs of overfitting.

Prevention Methods

For Decision Trees:

1. **Limit Depth** → max_depth
2. **Minimum Samples**:
    - min_samples_split — minimum samples to split a node
    - min_samples_leaf — minimum samples in a leaf
3. **Pruning**:
    - Remove branches that don't improve accuracy on validation data.
4. **Limit Features per Split** (max_features)
5. **Use Ensemble Methods**:
    - Random Forest, Gradient Boosting, etc.

General ML Practices:

- Cross-validation to detect overfitting early.
- More training data.
- Remove irrelevant/noisy features.

Analogy

- Think of overfitting like memorizing the answer key to one exam — you ace that test, but fail a different one with similar concepts.

5.What is bagging?

Bagging *(Bootstrap Aggregating)*

Definition

Bagging is an **ensemble learning technique** that reduces variance and helps prevent overfitting.
It works by:

1. **Bootstrapping**: Create multiple training datasets by sampling **with replacement** from the original dataset.
2. **Training**: Train a separate model (usually the same type, e.g., decision trees) on each bootstrap sample.
3. **Aggregating**:
    - **Classification** → Take the **majority vote** of all models.
    - **Regression** → Take the **average** of predictions.

Key Idea

- Each model sees a slightly different dataset.
- Combining many models smooths out random fluctuations (reduces variance).

Example: Random Forest

Random Forest = **Bagging + Random Feature Selection** at each split.

Analogy

- One doctor's diagnosis may be biased,
- but if **100 independent doctors** (trained on slightly different patient records) give their opinions, the majority vote will likely be more accurate.

Visual

Original Data

↓ (bootstrap sampling)

Dataset 1 → Train Model 1

Dataset 2 → Train Model 2

Dataset 3 → Train Model 3

↓

Aggregate predictions → Final Output

6.How do you visualize a decision tree?

Visualizing a Decision Tree

1. Why Visualize?

- Makes the model **interpretable**.
- Shows **decision rules** at each node.
- Helps detect **overfitting** if the tree is too deep or complex.

2. Methods in Scikit-learn

A. Using `plot_tree()`

python

Copy Edit

```python
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

clf = DecisionTreeClassifier(max_depth=3)
clf.fit(X_train, y_train)

plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=X.columns, class_names=['No Disease', 'Disease'], filled=True)
plt.show()
```

- **Pros**: Quick, no extra libraries.
- **Cons**: Not as pretty for large trees.

B. Using export_graphviz() + Graphviz

B. Using `export_graphviz()` + Graphviz

python

Copy Edit

```python
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(
    clf,
    out_file=None,
    feature_names=X.columns,
    class_names=['No Disease', 'Disease'],
    filled=True, rounded=True,
    special_characters=True
)
graph = graphviz.Source(dot_data)
graph.render("heart_tree")  # Saves as PDF or PNG
```

- **Pros**: Professional look, more control over formatting.
- **Cons**: Requires Graphviz installed.

3. Output Example

A decision tree diagram will have:

- **Feature & threshold** at each split
- **Gini/Entropy value**
- **Number of samples in the node**
- **Class distribution**
- **Majority class**

7.How do you interpret feature importance?

1. Definition

Feature importance tells you **how much each feature contributes** to the predictions made by the model.

- In tree-based models (Decision Tree, Random Forest, Gradient Boosting), it's usually based on how much **each feature reduces impurity** (e.g., Gini or Entropy) across all splits in all trees.

---

2. How it's Calculated (Scikit-learn)

For each feature:

1. Look at **every split** in the tree(s) where the feature is used.
2. Measure the **reduction in impurity** (e.g., entropy decrease) from that split.
3. Weight this reduction by the number of samples passing through the node.
4. Sum across all trees (for ensembles).
5. Normalize so all importances add up to **1.0**.

3. Interpretation

- **High value** → Feature is influential in the model's decision process.
- **Low value** → Feature has little effect on predictions (maybe can be dropped).
- Example from Heart dataset:

```
cholesterol       0.28
age               0.22
max_heart_rate    0.19
...
```

4. How to Visualize

python Copy Edit

```python
import matplotlib.pyplot as plt
import pandas as pd

importances = clf.feature_importances_
feat_importances = pd.Series(importances, index=X.columns)
feat_importances.sort_values().plot(kind='barh', figsize=(8,6))
plt.title("Feature Importances")
plt.show()
```

- **Horizontal bar charts** are common for clarity.

---

5. Caveats

- Feature importance in trees is **not causal** — a high score doesn't mean changing the feature will change the outcome.
- In correlated features, importance may be split between them.
- For more robust interpretation, use **permutation importance** or **SHAP values**.

8.What are the pros/cons of random forests?

✅ Advantages

1. **High accuracy** — Often outperforms a single decision tree.
2. **Robust to overfitting** — Bagging and random feature selection reduce variance.
3. **Works well with missing data** — Can still give good results with partial data.
4. **Handles large feature sets** — Useful for high-dimensional problems.
5. **Feature importance** — Can rank features by their contribution.
6. **Versatile** — Works for both classification and regression.
7. **Nonlinear decision boundaries** — Captures complex relationships.

❌ Disadvantages

1. **Less interpretable** — Harder to explain than a single decision tree.
2. **Computationally expensive** — More training and prediction time.
3. **Memory usage** — Needs more storage for multiple trees.
4. **Not ideal for very sparse high-dimensional data** — Linear models may perform better.
5. **Can still overfit** if the number of trees is too small or features aren't well-tuned.

Summary Table

Aspect	Random Forest	Single Decision Tree
Accuracy	Higher	Lower (more variance)
Interpretability	Lower	Higher
Overfitting Risk	Lower	Higher
Training Speed	Slower	Faster