1. Can we use Bagging for regression problems?

Yes. Bagging can be used for regression. In **Bagging Regressor**, predictions from multiple models are **averaged** instead of voted.

---

2. Difference between multiple model training and single model training

- **Single model training:** One model learns from the entire dataset.
- **Multiple model training:** Several models are trained (often on different samples), and their outputs are combined to improve performance.

---

3. Feature randomness in Random Forest

At each split, Random Forest selects a **random subset of features** instead of all features.
This:

- Reduces correlation between trees
- Improves generalization
- Reduces overfitting

---

4. What is OOB (Out-of-Bag) Score?

OOB score is an **internal validation method** in Bagging/Random Forest.
Each model is tested on data **not used during its training**.

---

5. Measuring feature importance in Random Forest

Feature importance can be measured by:

- **Mean decrease in impurity (Gini / MSE)**
- **Permutation importance**

Higher importance = greater impact on predictions.

---

6. Working principle of a Bagging Classifier

1. Create multiple bootstrap samples
2. Train a base classifier on each sample
3. Combine predictions using **majority voting**

---

7. Evaluating Bagging Classifier performance

Performance can be evaluated using:

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix
- OOB score
- Cross-validation

---

8. How does a Bagging Regressor work?

- Multiple regression models are trained on bootstrap samples
- Final output = **average of predictions**

---

9. Main advantage of ensemble techniques

They:

- Improve accuracy
- Reduce variance
- Reduce overfitting

- Are more robust than single models

---

10. Main challenge of ensemble methods

- High computational cost
- Reduced interpretability
- Longer training time

---

11. Key idea behind ensemble techniques

Combining multiple weak or base models produces a stronger, more reliable model.

---

12. What is a Random Forest Classifier?

A Random Forest Classifier is an ensemble of **decision trees** that:

- Uses bagging
- Uses feature randomness
- Predicts using **majority voting**

---

13. Main types of ensemble techniques

1. **Bagging**
2. **Boosting**
3. **Stacking**

---

14. What is ensemble learning?

Ensemble learning is a technique where **multiple models are combined** to improve prediction performance.

---

15. When should we avoid ensemble methods?

Avoid when:

- Dataset is very small
- Interpretability is critical
- Computational resources are limited

---

16. How does Bagging reduce overfitting?

By training models on different data samples and averaging results, it **reduces variance**.

---

17. Why is Random Forest better than a single Decision Tree?

Because it:

- Reduces overfitting
- Is more accurate
- Is more stable

---

18. Role of bootstrap sampling in Bagging

Bootstrap sampling:

- Creates diverse datasets
- Allows models to learn different patterns
- Improves ensemble strength

---

19. Real-world applications of ensemble techniques

- Fraud detection
- Medical diagnosis
- Stock price prediction
- Recommendation systems
- Image recognition

---

20. Difference between Bagging and Boosting

Bagging	Boosting
Models trained independently	Models trained sequentially
Reduces variance	Reduces bias
Equal weight to models	Higher weight to strong models
Example: Random Forest	Example: AdaBoost

21. Bagging Classifier (Decision Tree)

```
X, y = make_classification(n_samples=1000, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y)

model = BaggingClassifier(
    estimator=DecisionTreeClassifier(),
    n_estimators=100,
    random_state=42
)
model.fit(X_train, y_train)

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

---

22. Bagging Regressor (MSE)

```
X, y = make_regression(n_samples=1000, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
model = BaggingRegressor(
    estimator=DecisionTreeRegressor(),
    n_estimators=100,
    random_state=42
)
model.fit(X_train, y_train)

pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, pred))
```

---

23. Random Forest (Breast Cancer) + Feature Importance

```
data = load_breast_cancer()
X, y = data.data, data.target

rf = RandomForestClassifier(random_state=42)
rf.fit(X, y)

for name, imp in zip(data.feature_names, rf.feature_importances_):
    print(name, ":", imp)
```

---

24. RF Regressor vs Decision Tree

```
X, y = make_regression(n_samples=1000, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y)

dt = DecisionTreeRegressor().fit(X_train, y_train)
rf = RandomForestRegressor(random_state=42).fit(X_train, y_train)

print("DT MSE:", mean_squared_error(y_test, dt.predict(X_test)))
print("RF MSE:", mean_squared_error(y_test, rf.predict(X_test)))
```

---

25. OOB Score (Random Forest)

```
rf = RandomForestClassifier(oob_score=True, bootstrap=True,
random_state=42)
rf.fit(X, y)

print("OOB Score:", rf.oob_score_)
```

---

26. Bagging Classifier with SVM

```
model = BaggingClassifier(
    estimator=SVC(probability=True),
    n_estimators=10,
    random_state=42
)
model.fit(X_train, y_train)

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

---

27. RF with Different Trees

```
for n in [10, 50, 100]:
    rf = RandomForestClassifier(n_estimators=n, random_state=42)
    rf.fit(X_train, y_train)
    print(n, "trees accuracy:",
          accuracy_score(y_test, rf.predict(X_test)))
```

---

28. Bagging + Logistic Regression (AUC)

```
model = BaggingClassifier(
    estimator=LogisticRegression(max_iter=1000),
    n_estimators=50,
    random_state=42
)
model.fit(X_train, y_train)

probs = model.predict_proba(X_test)[:,1]
print("AUC:", roc_auc_score(y_test, probs))
```

---

29. RF Regressor Feature Importance

```
rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)

print(rf.feature_importances_)
```

---

30. Bagging vs Random Forest

```
bag = BaggingClassifier(random_state=42).fit(X_train, y_train)
rf  = RandomForestClassifier(random_state=42).fit(X_train, y_train)

print("Bagging:", accuracy_score(y_test, bag.predict(X_test)))
print("RF:", accuracy_score(y_test, rf.predict(X_test)))
```

---

31. RF + GridSearchCV

```
param_grid = {
    "n_estimators": [50, 100],
    "max_depth": [None, 5, 10]
}

grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best Params:", grid.best_params_)
```

---

32. Bagging Regressor (Different Estimators)

```
for n in [10, 50, 100]:
    model = BaggingRegressor(n_estimators=n, random_state=42)
    model.fit(X_train, y_train)
    print(n, "MSE:",
          mean_squared_error(y_test, model.predict(X_test)))
```

---

33. Misclassified Samples

```
rf = RandomForestClassifier().fit(X_train, y_train)
pred = rf.predict(X_test)

misclassified = np.where(pred != y_test)
print("Misclassified indices:", misclassified)
```

---

34. Bagging vs Single Tree

```
dt = DecisionTreeClassifier().fit(X_train, y_train)
bag = BaggingClassifier().fit(X_train, y_train)

print("DT:", accuracy_score(y_test, dt.predict(X_test)))
print("Bagging:", accuracy_score(y_test, bag.predict(X_test)))
```

---

35. Confusion Matrix

```
cm = confusion_matrix(y_test, rf.predict(X_test))
print(cm)
```

---

36. Stacking Classifier

```
estimators = [
    ('dt', DecisionTreeClassifier()),
    ('svm', SVC(probability=True)),
]

stack = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression()
)

stack.fit(X_train, y_train)
print("Stacking Accuracy:",
      accuracy_score(y_test, stack.predict(X_test)))
```

---

37. Top 5 Important Features

```
importances = rf.feature_importances_
indices = np.argsort(importances)[-5:]

for i in indices:
    print(data.feature_names[i], importances[i])
```

---

38. Precision, Recall, F1

```
pred = rf.predict(X_test)

print("Precision:", precision_score(y_test, pred))
print("Recall:", recall_score(y_test, pred))
print("F1:", f1_score(y_test, pred))
```

---

39. Effect of max_depth

```
for d in [2, 5, 10, None]:
    rf = RandomForestClassifier(max_depth=d, random_state=42)
    rf.fit(X_train, y_train)
    print("Depth:", d,
          accuracy_score(y_test, rf.predict(X_test)))
```

---

40. Bagging with Different Base Estimators

```
for est in [DecisionTreeRegressor(), KNeighborsRegressor()]:
    model = BaggingRegressor(estimator=est)
    model.fit(X_train, y_train)
    print(type(est).__name__,
          mean_squared_error(y_test, model.predict(X_test)))
```

---

41. RF ROC–AUC

```
probs = rf.predict_proba(X_test)[:,1]
print("ROC-AUC:", roc_auc_score(y_test, probs))
```

---

42. Cross-Validation (Bagging)

```
scores = cross_val_score(BaggingClassifier(), X, y, cv=5)
print("CV Accuracy:", scores.mean())
```

---

43. Precision–Recall Curve

```
precision, recall, _ = precision_recall_curve(y_test, probs)

plt.plot(recall, precision)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.show()
```

---

44. Stacking (RF + Logistic)

```
estimators = [
    ('rf', RandomForestClassifier()),
]

stack = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression()
)

stack.fit(X_train, y_train)
print("Accuracy:",
      accuracy_score(y_test, stack.predict(X_test)))
```

---

45. Bootstrap Levels in Bagging

```
for frac in [0.5, 0.7, 1.0]:
    model = BaggingRegressor(
        max_samples=frac,
        random_state=42
```

```
)
model.fit(X_train, y_train)
print("Samples:", frac,
      mean_squared_error(y_test, model.predict(X_test)))
```