Random Forest – Complete Beginner-to-Advanced Revision Guide

---

1. Introduction

What is Random Forest?

Random Forest (RF) is an **ensemble learning algorithm** used for **classification and regression**. It builds a large number of **decision trees** and combines their predictions to improve accuracy, robustness, and generalisation.

It belongs to the family of **bagging (Bootstrap Aggregating)** methods.

Core Intuition

> "Many weak, noisy trees together make a strong, stable model."

- Each tree is trained on a different random subset of data
- Each split considers a random subset of features
- Final prediction = aggregation of all trees

Real-World Applications

- Credit risk modelling
- Medical diagnosis
- Fraud detection
- Feature importance analysis
- Remote sensing

---

2. Mathematical Foundations

2.1 Decision Trees Recap

A decision tree splits data recursively to maximise **purity** at each node.

Classification Criteria

Gini Impurity

$$ G = 1 - \sum_{k=1}^K p_k^2 $$

Entropy

$$ H = -\sum_{k=1}^K p_k \log_2 p_k $$

Information Gain

$$ IG = H(\text{parent}) - \sum_{j} \frac{N_j}{N} H(\text{child}_j) $$

2.2 Regression Trees

Split criterion: **Mean Squared Error (MSE)**

$$ \text{MSE} = \frac{1}{n} \sum (y - \hat{y})^2 $$

2.3 Bootstrap Sampling

Random Forest uses **sampling with replacement**.

Probability a sample is *not* selected in one bootstrap:

$$ \left(1 - \frac{1}{n}\right)^n \approx e^{-1} \approx 0.368 $$

≈ **36.8%** samples are **Out-Of-Bag (OOB)**.

2.4 Ensemble Prediction

Classification

$$ \hat{y} = \text{mode}{T_1(x), T_2(x), \dots, T_M(x)} $$

Regression

$$ \hat{y} = \frac{1}{M} \sum_{m=1}^M T_m(x) $$

---

3. Randomness in Random Forest

3.1 Data Randomness (Bagging)

- Each tree sees a different dataset
- Reduces variance

3.2 Feature Randomness

At each split, choose a random subset of features:

- Classification: $\sqrt{d}$
- Regression: $d/3$

---

4. Bias–Variance Tradeoff

- Single tree: low bias, high variance
- Random Forest: low bias, **much lower variance**

---

5. Model Training

Training Steps

1. Draw bootstrap sample
2. Grow decision tree
3. Random feature selection at each split
4. Repeat for M trees

Important Hyperparameters

- `n_estimators`
- `max_depth`
- `min_samples_split`
- `max_features`

---

6. Model Evaluation

Evaluation Metrics

Classification: - Accuracy - Precision - Recall - F1-score - ROC-AUC

Regression: - MSE - RMSE - MAE

Out-Of-Bag (OOB) Error

Uses OOB samples as validation set:

$$ \text{OOB Error} = \frac{1}{n} \sum \mathbb{1}(y_i \neq \hat{y}_i^{OOB}) $$

---

7. Interpretation

Feature Importance (MDI)

Based on impurity reduction:

$$ FI_j = \sum_{t \in T_j} p(t) \Delta i(t) $$

Permutation Importance

Measures performance drop after shuffling a feature.

---

8. Assumptions

- Observations are i.i.d.
- Trees are weakly correlated
- Enough trees are used

---

9. Common Pitfalls & Misconceptions

- ❌ Thinking more trees always overfit (they don’t)
- ❌ Ignoring class imbalance
- ❌ Blindly trusting feature importance

- ❌ Overusing deep trees
- ❌ Forgetting to tune `max_features`

---

10. Python Implementation

10.1 From Scratch (Conceptual)

```
import numpy as np

# Pseudocode-like illustration
for tree in range(n_trees):
    sample = bootstrap_sample(X, y)
    tree = train_decision_tree(sample)
    forest.append(tree)
```

10.2 Using scikit-learn

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    max_features='sqrt',
    oob_score=True,
    random_state=42
)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("OOB Score:", model.oob_score_)
```

10.3 Feature Importance Plot

```
import matplotlib.pyplot as plt

importances = model.feature_importances_
plt.bar(range(len(importances)), importances)
plt.title("Feature Importance")
plt.show()
```

---

11. Advanced Topics

- Extremely Randomised Trees (ExtraTrees)
- Random Forest for Survival Analysis
- Quantile Regression Forests

---

12. When NOT to Use Random Forest

- Very small datasets
- Highly interpretable models required
- Extrapolation beyond data range

---

13. Best Practices

- Tune `max_depth` and `max_features`
- Use OOB score
- Handle class imbalance
- Use permutation importance

---

14. Summary

Random Forest is: - Powerful - Robust - Low-tuning - Industry-proven

It is often the **first strong baseline** in ML projects.

End of Revision Guide