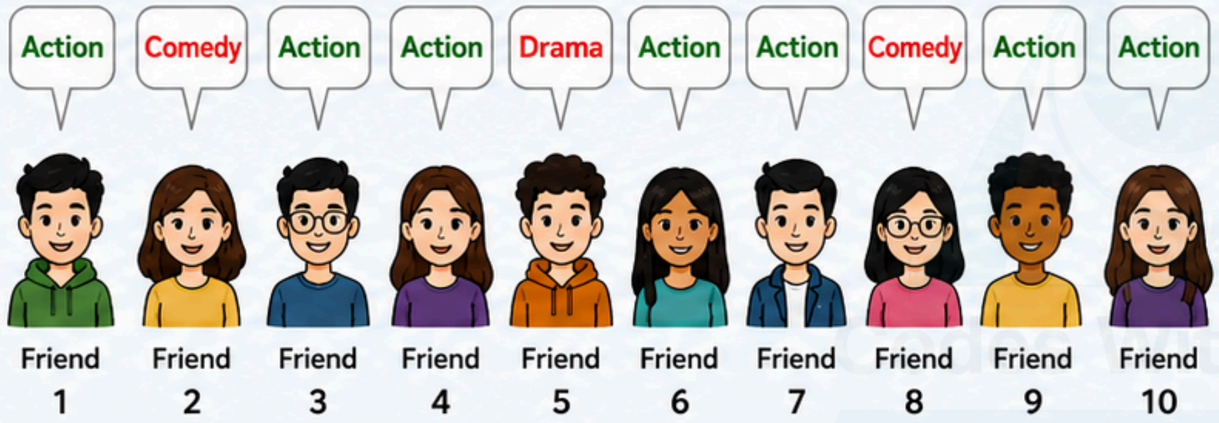
Bagging vs Boosting – Simple Analogy

Goal: Decide which movie to watch

BAGGING

Ask 10 friends independently
and take the majority vote.

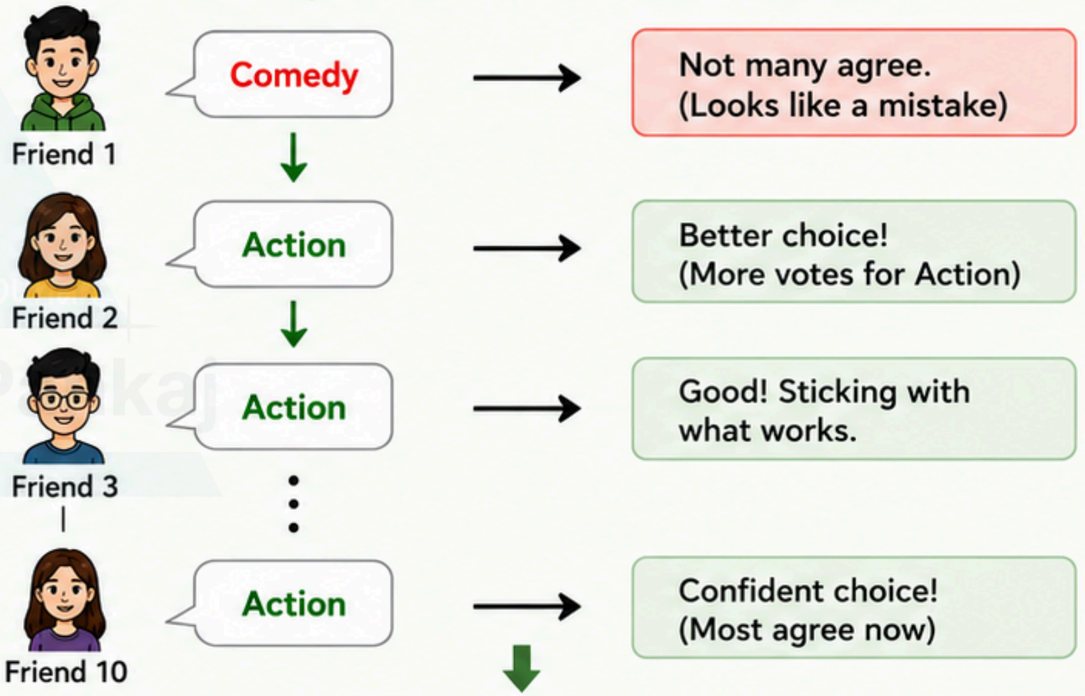


Majority Vote:
Action (7 votes)
We watch an Action movie!

Each friend decides independently (no help from others).

BOOSTING

Ask friends one by one.
Each new friend learns from the mistakes
of the previous ones and tries to fix them.



Final Decision:
Action
We watch an Action movie!

Each friend learns from previous mistakes and improves the decision.

What is Bagging? (Bootstrap Aggregating)

Simple Meaning:

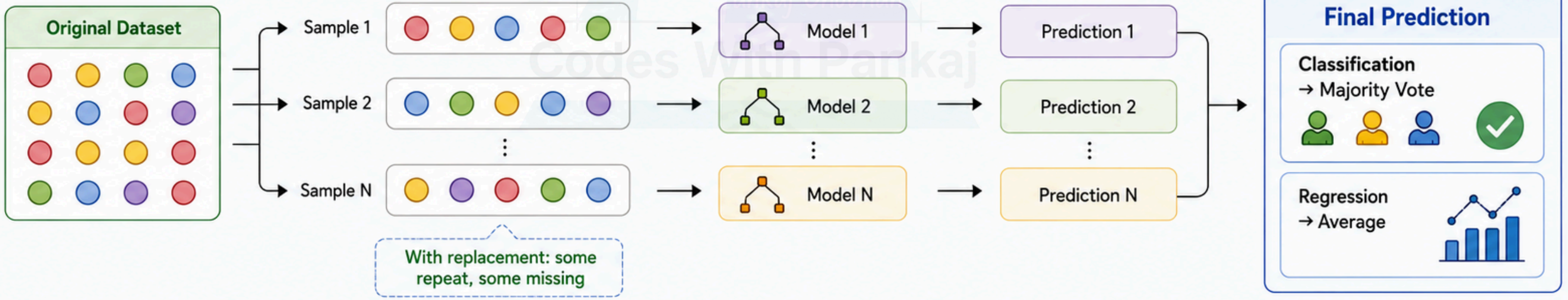
Train many models separately on slightly different versions of the data, then combine their answers (average or majority vote).

Why it works:

- **It reduces overfitting** (when a model memorizes the data too much and performs badly on new data).
- It makes the final prediction more stable and reliable.

How it works (very simply):

1. Start with original data
2. Take random samples with replacement (Bootstrap samples)
3. Train a separate model on each sample
4. Get predictions from all models
5. Combine predictions for final output



Popular Example: Random Forest = Bagging + Decision Trees

Random Forest

What is Boosting?

Simple Meaning:

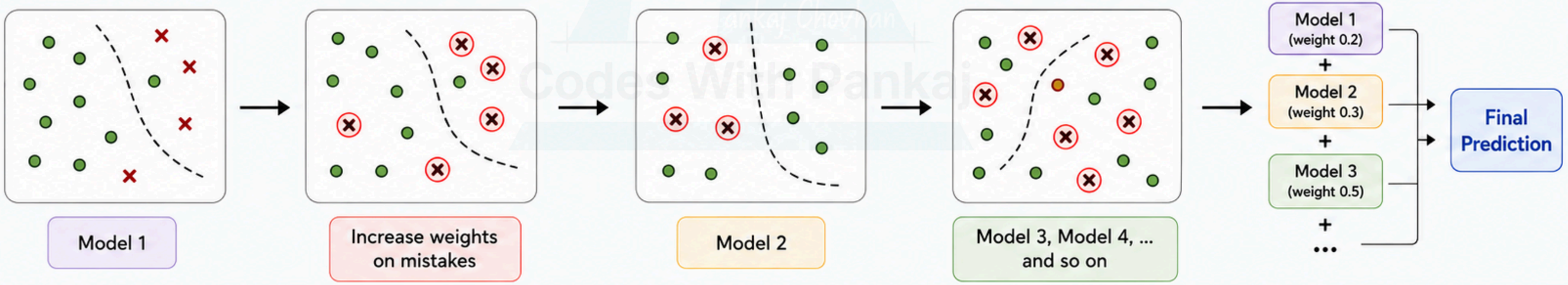
Build models one after another.
Each new model focuses on the mistakes made by the previous models.

Why it works:

- It reduces bias (systematic errors) and can create very powerful models.
- Models learn from each other's weaknesses.

How it works (very simply):

1. Train first model on the data.
2. Find where it made mistakes → Give those mistakes higher importance (higher weights).
3. Train second model focusing more on the hard examples.
4. Repeat this process. Each model tries to correct the previous one.
5. Final prediction = Weighted combination of all models.



Popular Examples:

1. AdaBoost
2. Gradient Boosting (XGBoost, LightGBM, CatBoost)

Key Differences

Feature	Bagging	Boosting
How models are trained	Independently (in parallel)	Sequentially (one after another)
Focus	Reduce variance (overfitting)	Reduce bias + variance
Data sampling	Random samples (with replacement)	Same data, but weights change
Final decision	Average / Majority vote	Weighted sum
Speed	Faster (can train in parallel)	Slower (must train sequentially)
Risk	Less chance of overfitting	Can overfit if not careful

Bagging =
Team of independent learners → vote or average.
Great for reducing overfitting.

Boosting =
Team of learners where each one learns from the mistakes of the previous.
Great for reducing bias.

Step-by-Step Examples

Bagging Example (Predicting if a student will Pass or Fail)

1 You have 1000 students' data (marks, attendance, etc.).

2 Create 5 different random samples (each with 1000 rows, but some students repeated, some missing).

3 Train 5 Decision Trees — one on each sample.

4 For a new student: Each tree makes a prediction.

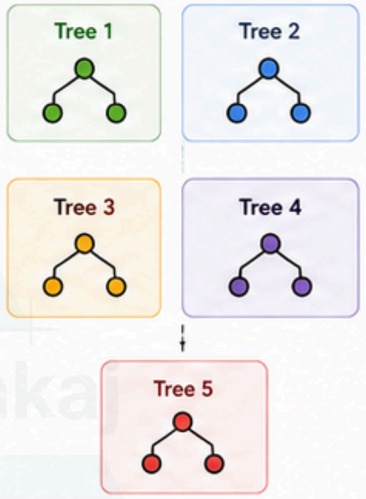
Original Dataset (1000 students)

ID	Marks	Attendance	...	Result
1	78	90%	...	Pass
2	45	60%	...	Fail
...	...	...	...	...
1000	88	92%	...	Pass

→

Sample 1 (1000 rows)		Sample 2 (1000 rows)		Sample 3 (1000 rows)		Sample 4 (1000 rows)		Sample 5 (1000 rows)	
ID	...	ID	...	ID	...	ID	...	ID	...
25	...	3	...	10	...	8	...	8	...
7	...	115	...	42	...	42	...	200	...
7	...	115	...	115	...	42	...	30	...
...		...		...		...		...	
991	...	870	...	998	...	998	...	1000	...

→



→

New Student (Unknown)

- Tree 1 says **Pass**
- Tree 2 says **Fail**
- Tree 3 says **Pass**
- Tree 4 says **Pass**
- Tree 5 says **Fail**

Final Answer:

Majority = Pass (3 out of 5)

→ Pass | Fail | Pass | Pass | Fail → **Pass (Majority)**

This is how **Random Forest** works.

Bagging vs Boosting - Complete Python Example

1 Step 1: Install Required Libraries (Run once)

```
Bash | pip install numpy pandas scikit-learn xgboost matptol1ib seaborn
```

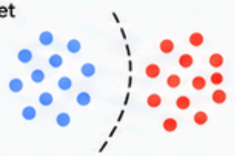
2 Step 2: Complete Python Code

```
# ============================================
# BAGGING vs BOOSTING - Complete Example
# ============================================

import numpy as np
import pandas as pd
import matptol1ib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
                            confusion_matrix

# ------------------ Bagging ------------------
from sklearn.ensemble import RandomForestClassifier   # Bagging

# ------------------ Boosting -----------------
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier

import warnings
warnings.filterwarnings('ignore')

# ============================================
# Step 1: Load Dataset
# ============================================
print("Step 1: Loading Dataset...\n")

data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

print(f"Dataset Shape: {X.shape}")
print(f"Target Names: {data.target_names}")
print("0 = Malignant, 1 = Benign\n")

# ============================================
# Step 2: Split Data
# ============================================
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.25,
                                   random_state=42)
print(f"Training samples: {X_train.shape[0]}")
print(f"Testing samples:  {X_test.shape[0]}\n")

# ============================================
# Step 3: BAGGING - Random Forest
# ============================================
print("Step 3: Training Bagging (Random Forest)...")

rf_model = RandomForestClassifier(
    n_estimators=100,      # Number of trees
    max_depth=5,
    random_state=42
)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"✅ Random Forest (Bagging) Accuracy: {rf_accuracy:.4f}
    f" ({rf_accuracy*100:.2f}%)")

# ============================================
# Step 4: BOOSTING - Gradient Boosting & XGBoost
# ============================================
print("\nStep 4: Training Boosting Models...")

# 4.1 Gradient Boosting
gb_model = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)
gb_model.fit(X_train, y_train)
gb_pred = gb_model.predict(X_test)
gb_accuracy = accuracy_score(y_test, gb_pred)
print(f"✅ Gradient Boosting Accuracy: {gb_accuracy:.4f}
    f" ({gb_accuracy*100:.2f}%)")

# 4.2 XGBoost (Most Popular Boosting)
xgb_model = XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42,
    eval_metric='logloss'
)
xgb_model.fit(X_train, y_train)
xgb_pred = xgb_model.predict(X_test)
xgb_accuracy = accuracy_score(y_test, xgb_pred)
print(f"✅ XGBoost Accuracy:          {xgb_accuracy:.4f}
    f" ({xgb_accuracy*100:.2f}%)")

# ============================================
# Step 5: Comparison
# ============================================

print("\n" + "="*50)
print("FINAL COMPARISON")
print(""="*50)

results = pd.DataFrame({
  'Model': ['Random Forest (Bagging)',
            'Gradient Boosting',
            'XGBoost (Boosting)'],
  'Accuracy': [rf_accuracy, gb_accuracy, xgb_accuracy]
})

print(results.round(4))

best_model = results.loc[results['Accuracy'].idxmax()]
print(f"\n🏆 Best Model: {best_model['Model']} with
      f"   best_model['Accuracy']:.4f} accuracy")

# ====================================
# Step 6: Detailed Report (Best Model)
# ====================================

print("\nDetailed Classification Report (XGBoost):")
print(classification_report(y_test, xgb_pred,
          target_names=data.target_names))
```

Dataset Used

Breast Cancer Wisconsin Dataset

- 569 samples
- 30 features
- Binary Classification:
  0 = Malignant, 1 = Benign



How to Run the Complete Code

1. Copy all the code above
2. Paste in a new Jupyter Notebook or Python file (bagging_boosting.py)
3. Run all cells

Expected Output Example

```
Step 1: Loading Dataset...

Dataset Shape: (569, 30)
Target Names:  ['malignant' 'benign']
0 = Malignant, 1 = Benign

Training samples:  426
Testing samples:   143

Step 3: Training Bagging (Random Forest)...
✅ Random Forest (Bagging) Accuracy: 0.9650 (96.50%)
Step 4: Training Boosting Models...

✅ Gradient Boosting Accuracy: 0.9580 (95.80%)
✅ XGBoost Accuracy:           0.9720 (97.20%)

==================================================
FINAL COMPARISON
==================================================
-   Model                      Accuracy
0   Random Forest (Bagging)     0.9650
1   Gradient Boosting           0.9580
2   XGBoost (Boosting)          0.9720

🏆 Best Model: XGBoost (Boosting) with 0.9720 accuracy

Detailed Classification Report (XGBoost):
              precision    recall  f1-score   support

   malignant       0.96      0.95      0.95        51
      benign       0.98      0.98      0.98        92

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```