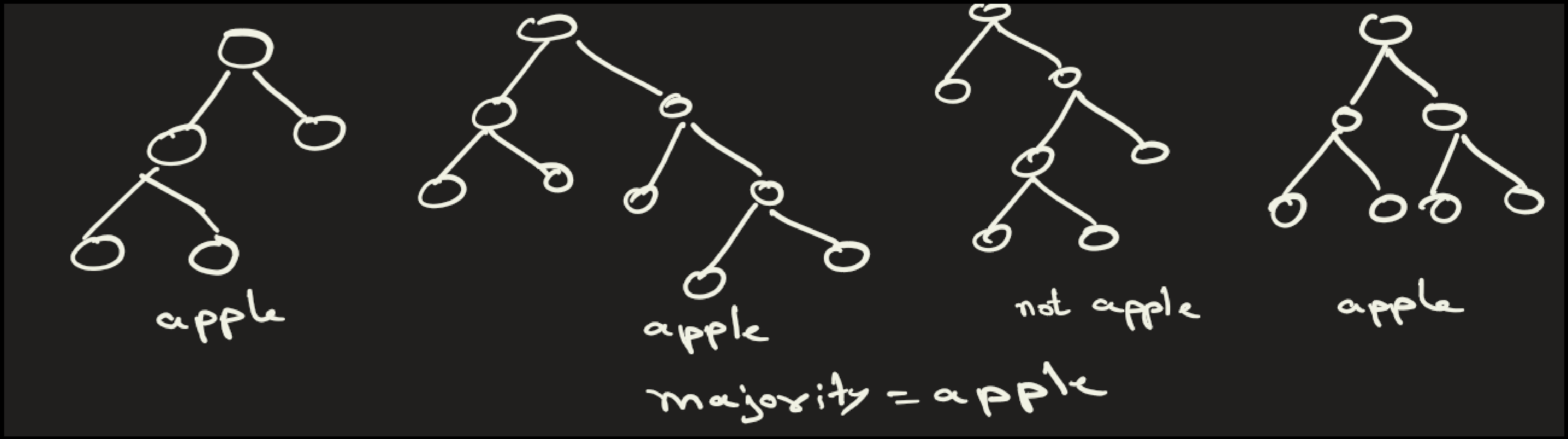
Random Forest: Idea

Random Forest is an **ensemble learning algorithm** that uses **many decision trees** to make predictions.
Instead of relying on a single tree (which may overfit), Random Forest builds **multiple trees** and combines their results.
It's like asking 100 experts instead of trusting just one.

It can be used for both classification and regression.



Random Forest: Algorithm

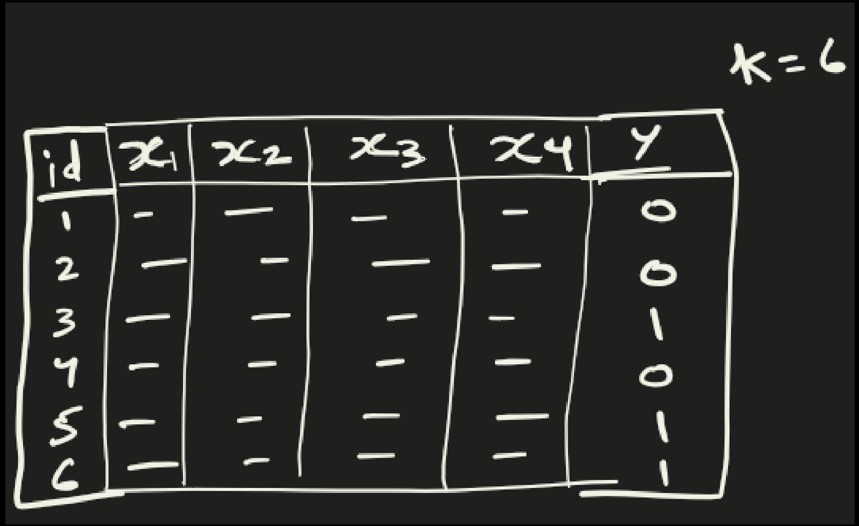
Step 1: Create Random Samples of Data and Features

From a dataset with *k* total records, we repeatedly select:

- **n random data points** (with replacement) → bootstrap sample
- **m random features** out of all available features

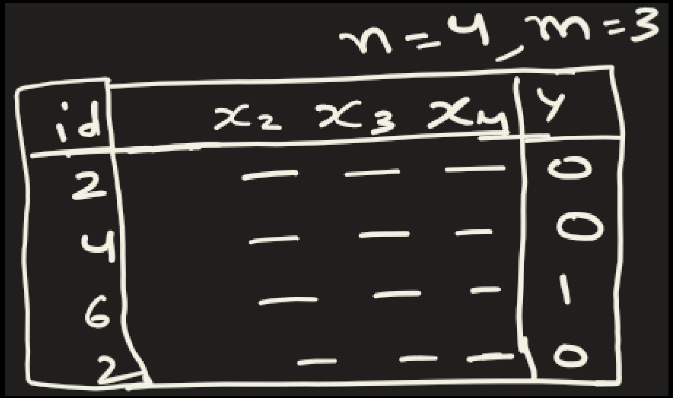
Each sample + feature subset will be used to build one decision tree.
This randomness ensures that every tree is slightly different.

Example: Below we show 3 random samples from k=6 data points with 4 features. Each sample has n=4 data points and m=3 features.



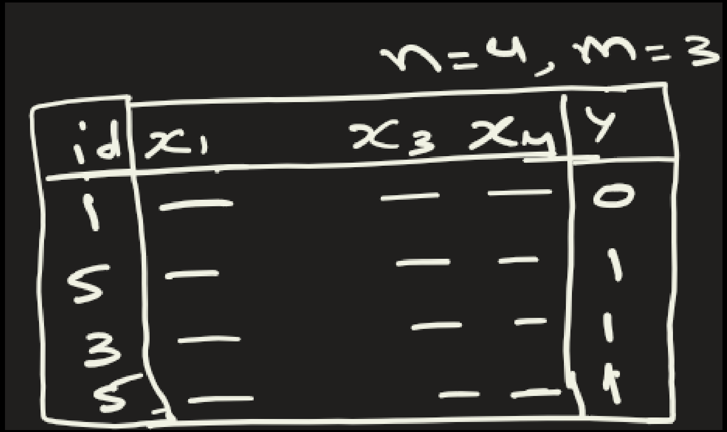
$k=6$

id	x_1	x_2	x_3	x_4	y
1	–	–	–	–	0
2	–	–	–	–	0
3	–	–	–	–	1
4	–	–	–	–	0
5	–	–	–	–	1
6	–	–	–	–	1



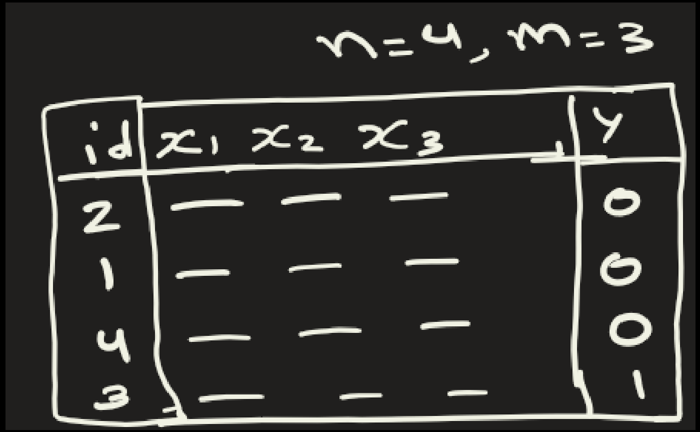
$n=4, m=3$

id	x_2	x_3	x_4	y
2	–	–	–	0
4	–	–	–	0
6	–	–	–	1
2	–	–	–	0



$n=4, m=3$

id	x_1	x_3	x_4	y
1	–	–	–	0
5	–	–	–	1
3	–	–	–	1
5	–	–	–	1



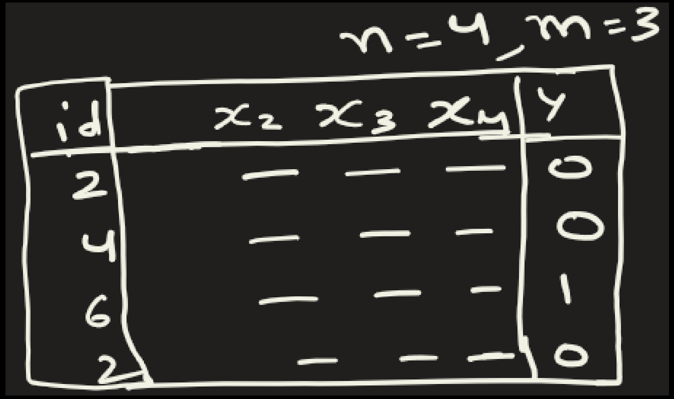
$n=4, m=3$

id	x_1	x_2	x_3	y
2	–	–	–	0
1	–	–	–	0
4	–	–	–	0
3	–	–	–	1

Random Forest: Algorithm

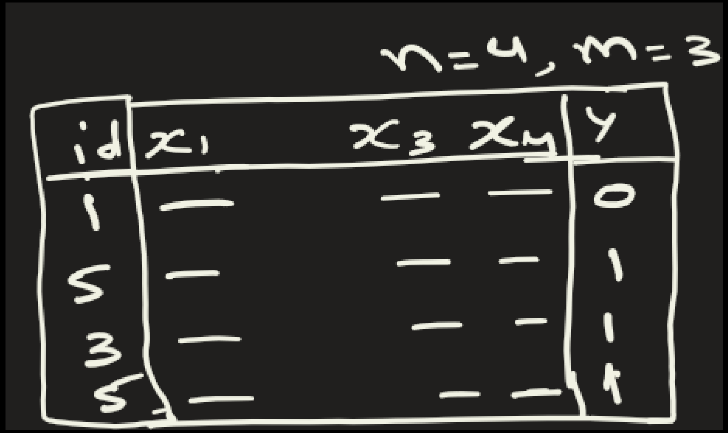
Step 2: Build a Decision Tree for Each Sample

For each bootstrap sample grow a decision tree.
At every split, evaluate **only the selected subset of features**, not all features.



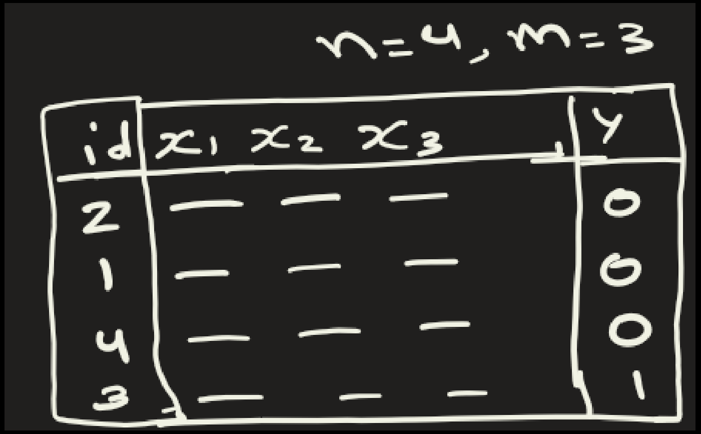
$n=4, m=3$

id	x_2	x_3	x_4	Y
2	—	—	—	0
4	—	—	—	0
6	—	—	—	1
2	—	—	—	0



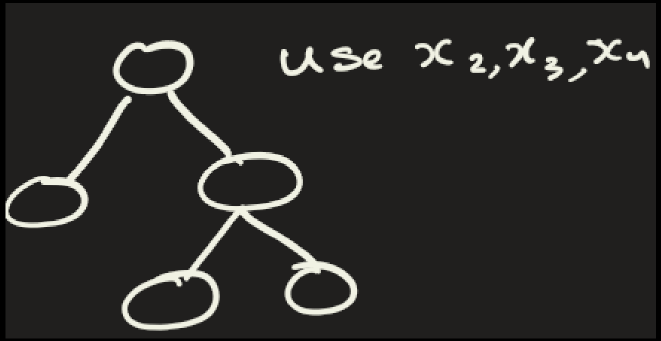
$n=4, m=3$

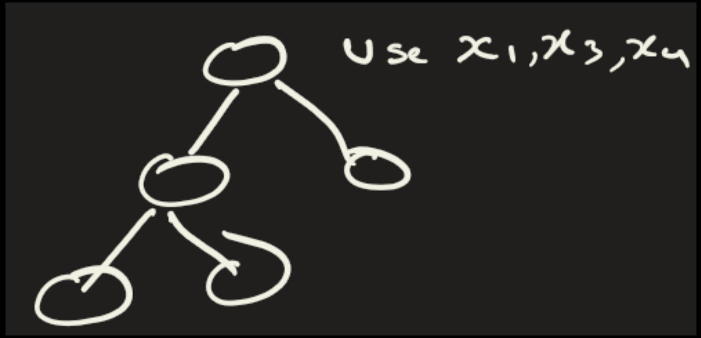
id	x_1	x_3	x_4	Y
1	—	—	—	0
5	—	—	—	1
3	—	—	—	1
5	—	—	—	1

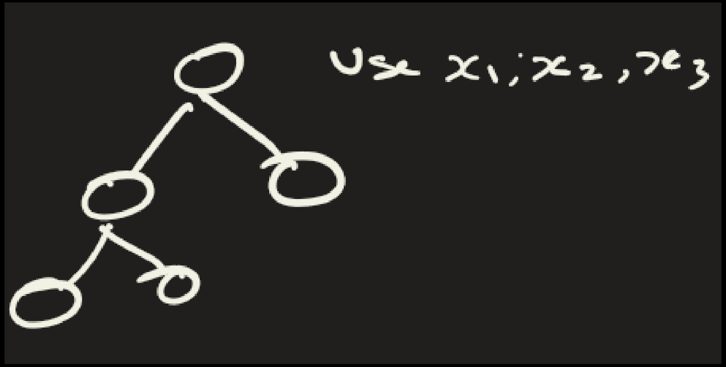


$n=4, m=3$

id	x_1	x_2	x_3	Y
2	—	—	—	0
1	—	—	—	0
4	—	—	—	0
3	—	—	—	1





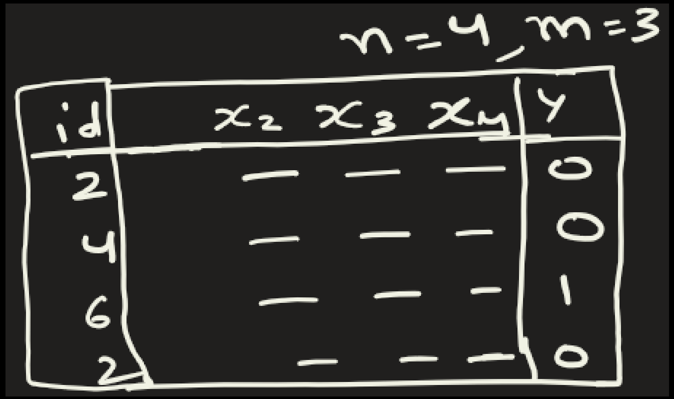


Random Forest: Algorithm

Step 3: Generate Predictions from All Trees

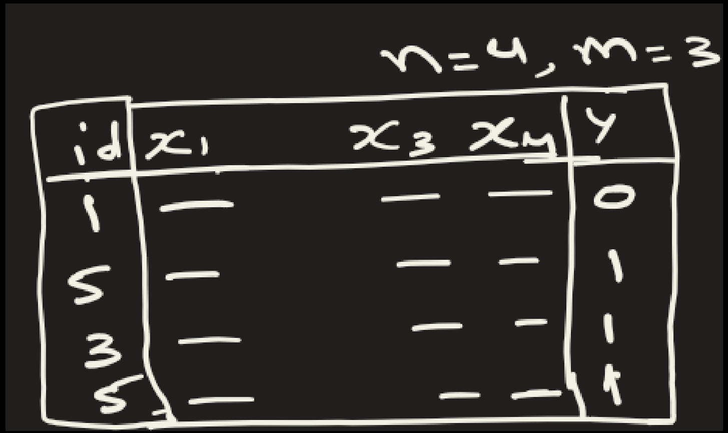
Each decision tree makes its own prediction:

- A class label (for classification)
- A numerical value (for regression)



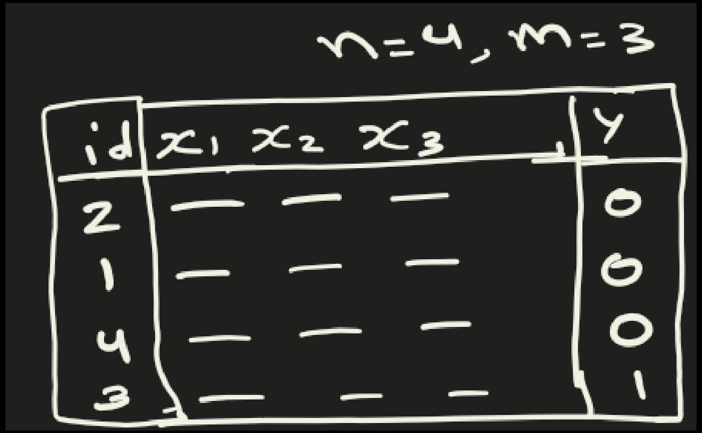
$n=4, m=3$

id	x_2	x_3	x_4	Y
2	—	—	—	0
4	—	—	—	0
6	—	—	—	1
2	—	—	—	0



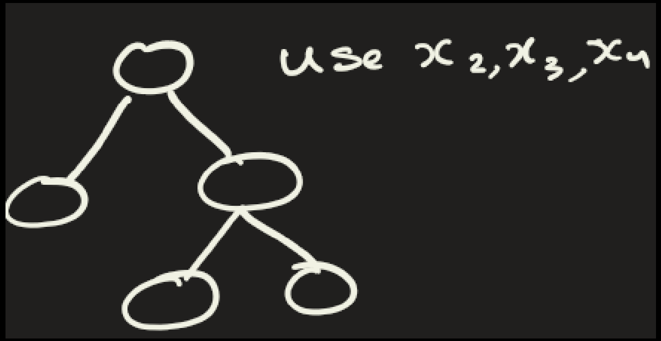
$n=4, m=3$

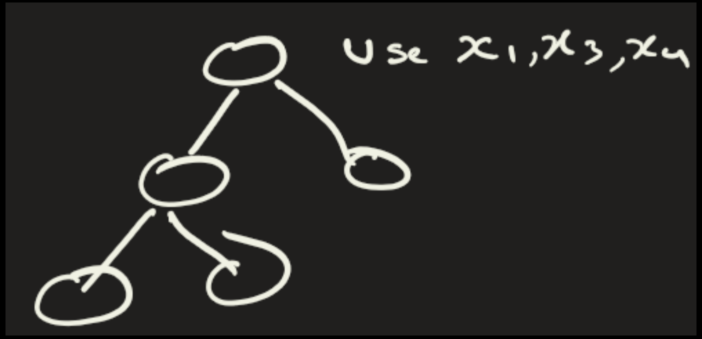
id	x_1	x_3	x_4	Y
1	—	—	—	0
5	—	—	—	1
3	—	—	—	1
5	—	—	—	1

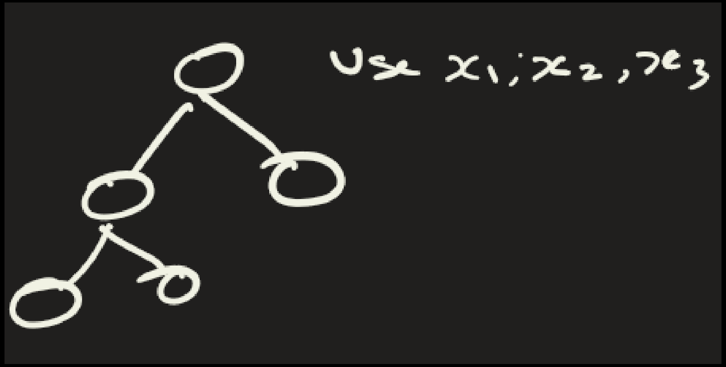


$n=4, m=3$

id	x_1	x_2	x_3	Y
2	—	—	—	0
1	—	—	—	0
4	—	—	—	0
3	—	—	—	1





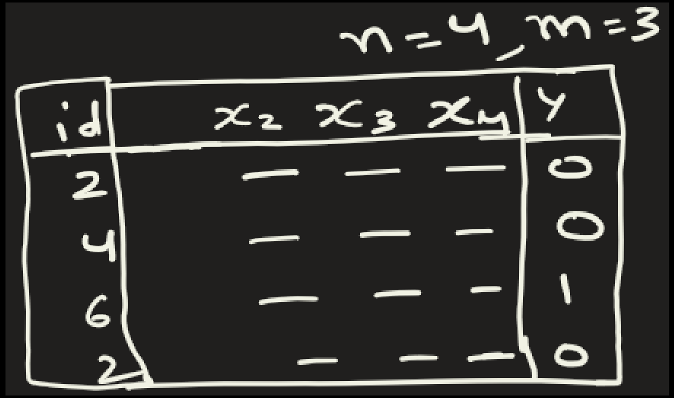


Random Forest: Algorithm

Step 4: Combine All Predictions to Form the Final Output

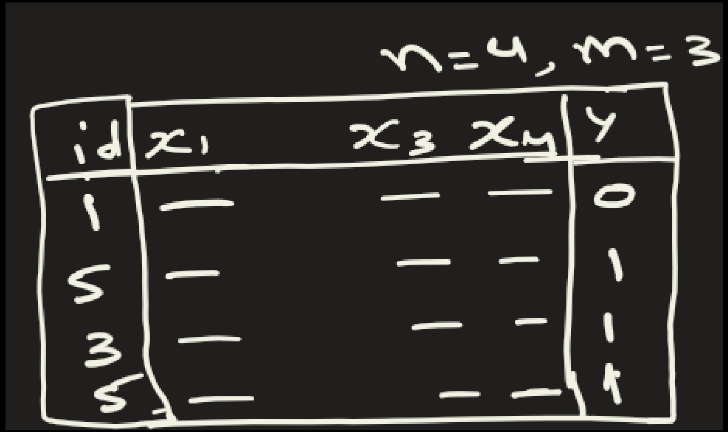
- **Classification:** Use **majority voting** → the class predicted by most trees becomes the final prediction.
- **Regression:** Use **averaging** → take the mean of all tree predictions.

Note: Bootstrapping in step1 and Aggregation in step4 is commonly referred to as **Bagging.** This reduces variance.



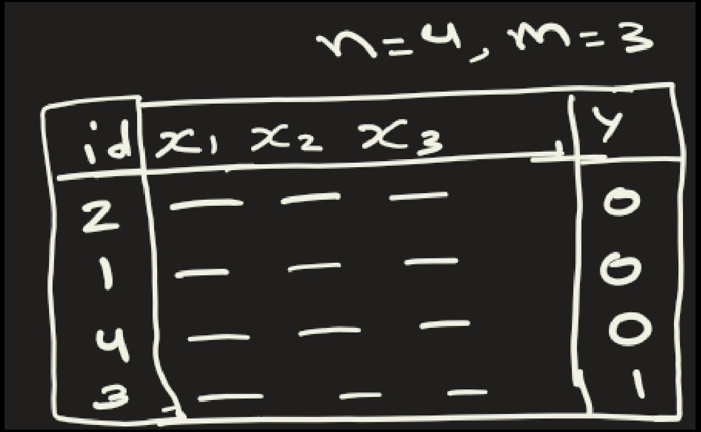
$n=4, m=3$

id	x_2	x_3	x_4	y
2	—	—	—	0
4	—	—	—	0
6	—	—	—	1
2	—	—	—	0



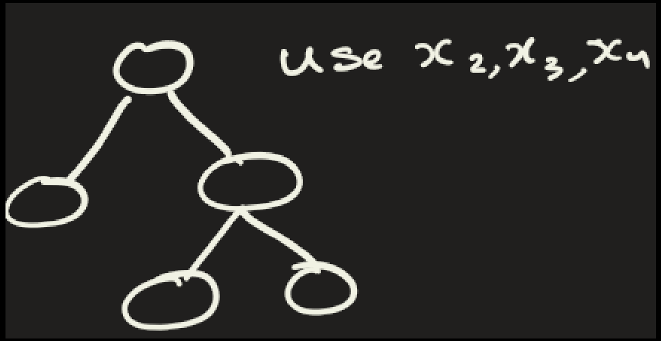
$n=4, m=3$

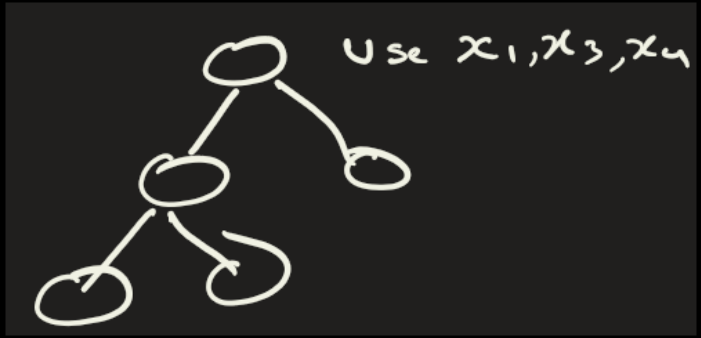
id	x_1	x_3	x_4	y
1	—	—	—	0
5	—	—	—	1
3	—	—	—	1
5	—	—	—	1

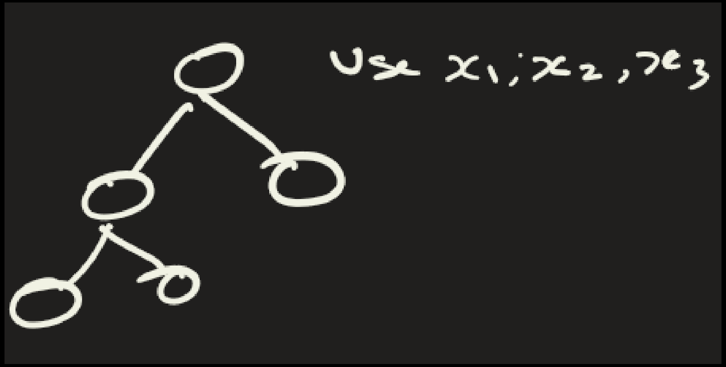


$n=4, m=3$

id	x_1	x_2	x_3	y
2	—	—	—	0
1	—	—	—	0
4	—	—	—	0
3	—	—	—	1



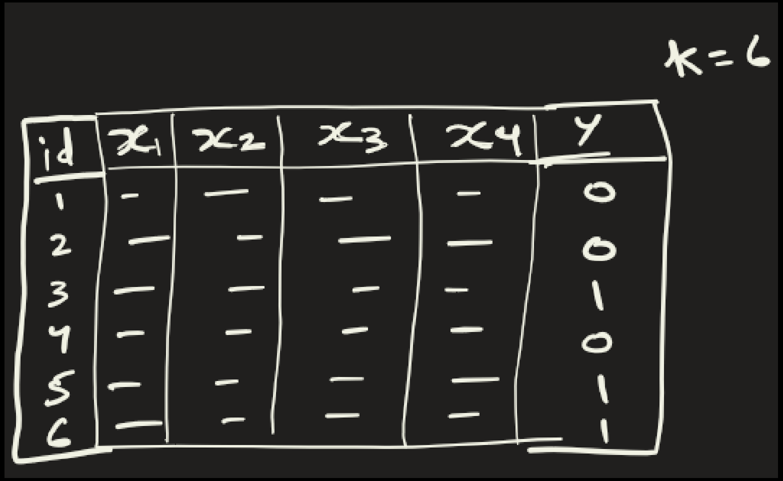




Random Forest: Key points

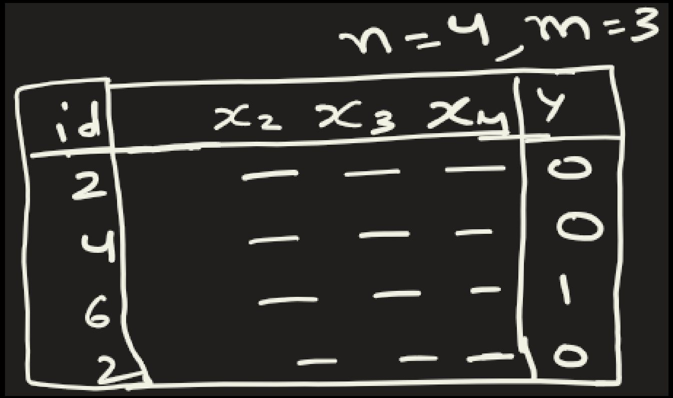
Why is it called random?

Because we select random samples and random features.
In example below, we select 3 random samples with 3 features in each



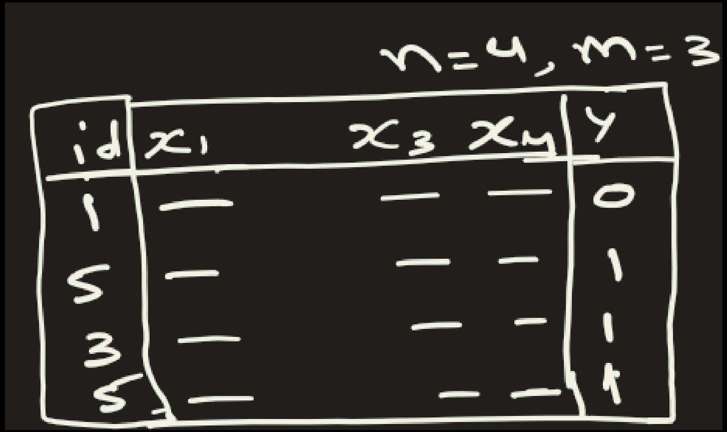
$k=6$

id	x_1	x_2	x_3	x_4	y
1	—	—	—	—	0
2	—	—	—	—	0
3	—	—	—	—	1
4	—	—	—	—	0
5	—	—	—	—	1
6	—	—	—	—	1



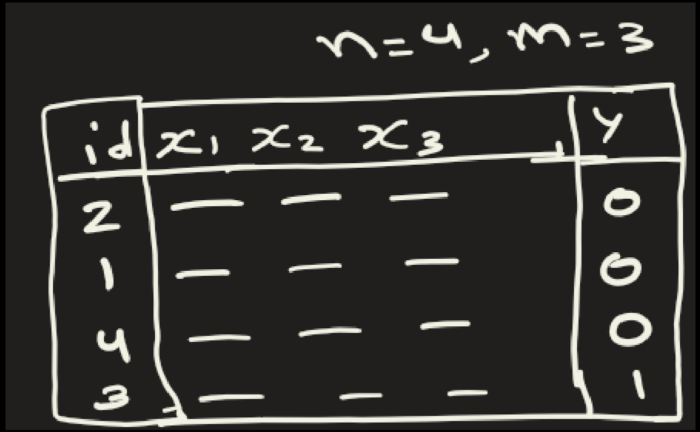
$n=4, m=3$

id	x_2	x_3	x_4	y
2	—	—	—	0
4	—	—	—	0
6	—	—	—	1
2	—	—	—	0



$n=4, m=3$

id	x_1	x_3	x_4	y
1	—	—	—	0
5	—	—	—	1
3	—	—	—	1
5	—	—	—	1



$n=4, m=3$

id	x_1	x_2	x_3	y
2	—	—	—	0
1	—	—	—	0
4	—	—	—	0
3	—	—	—	1

Random Forest: Key points

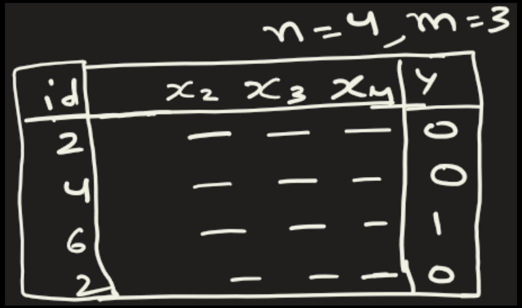
Why Bootstrapping? (Random Sampling of Data)

1) Reduce Overfitting

If every tree is trained on the **same dataset**, they will all look similar and may overfit the training data.
Different data → different splits → different trees
Bootstrapping ensures **every tree sees a slightly different dataset**, so each tree learns something different.

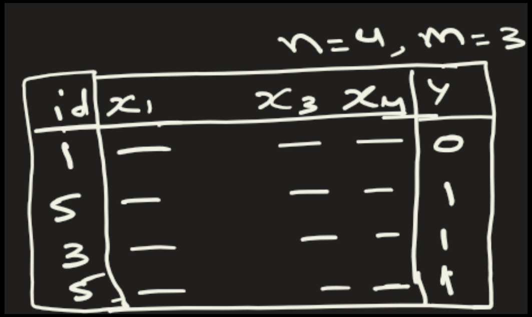
2) **Provide Built-in Model Validation (OOB Error)**

Since some data points are *not selected* in each bootstrap sample, those points act as **Out-of-Bag data**.
OOB data helps estimate accuracy without a separate test set.



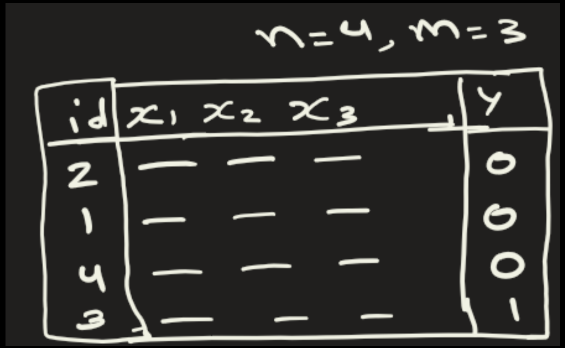
$n=4, m=3$

id	x_2	x_3	x_4	Y
2	—	—	—	0
4	—	—	—	0
6	—	—	—	1
2	—	—	—	0



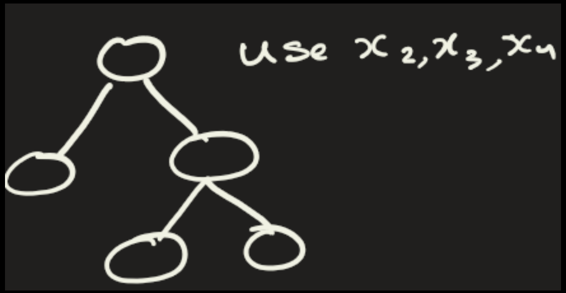
$n=4, m=3$

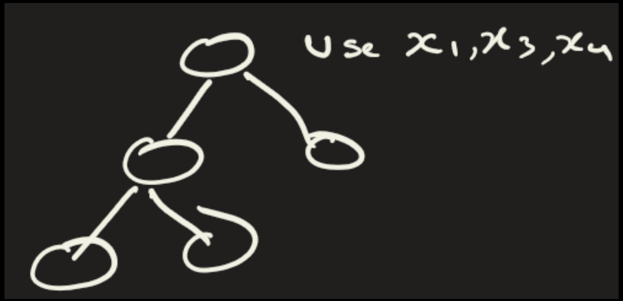
id	x_1	x_3	x_4	Y
1	—	—	—	0
5	—	—	—	1
3	—	—	—	1
5	—	—	—	1

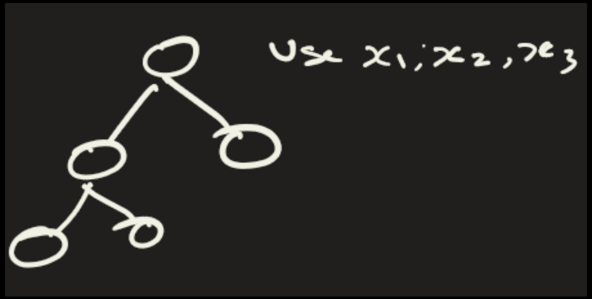


$n=4, m=3$

id	x_1	x_2	x_3	Y
2	—	—	—	0
1	—	—	—	0
4	—	—	—	0
3	—	—	—	1







Random Forest: Key points

Why Random Feature Selection at Each Split?

1) Break Feature Dominance
If one strong feature exists, all trees will use it at the top split. This would make trees very similar.
Random feature selection forces trees to explore **different feature combinations**, increasing variety.

2) **Reduce Correlation Between Trees**
For a strong ensemble, models must be:
Accurate individually
Uncorrelated with each other
Random feature selection reduces correlation, giving better combined predictions.

3) **Improve Generalization**
Trees trained on different feature subsets:
Capture different data patterns
Avoid overfitting
Perform better on unseen data

Random Forest: Key points

How many features should we select ?

Researchers have found it to be square root or log of total number of features works well.

Random Forest: Key points

The combination of both **Random Sampling of Data and Random Feature Selection at Each Split** :

- **Creates many diverse, uncorrelated trees**
- **Reduces overfitting**
- **Improves accuracy and stability**

Random Forest: Key points

Random Forest Classifier Parameters:

- **n_estimators**: Number of trees in the forest. More trees generally lead to better performance, but at the cost of computational time. Start with a value of 100 and increase as needed.

- **max_depth**: Maximum depth of each tree. Deeper trees can capture more complex patterns, but also risk overfitting. Experiment with values between 5 and 15, and consider lower values for smaller datasets.

- **max_features**: Number of features considered for splitting at each node. A common value is 'sqrt' (square root of the total number of features). Adjust based on dataset size and feature importance.

- **criterion**: Function used to measure split quality ('gini' or 'entropy').Gini impurity is often slightly faster, but both are generally similar in performance.

- **min_samples_split**: Minimum samples required to split a node. Higher values can prevent overfitting, but too high can hinder model complexity. Start with 2 and adjust as needed.

- **min_samples_leaf**: Minimum samples required to be at a leaf node. Similar to min_samples_split, but focused on leaf nodes. Start with 1 and adjust as needed.

Random Forest: Coding

Python code on Heart disease

EXTRA

Python code on Heart disease