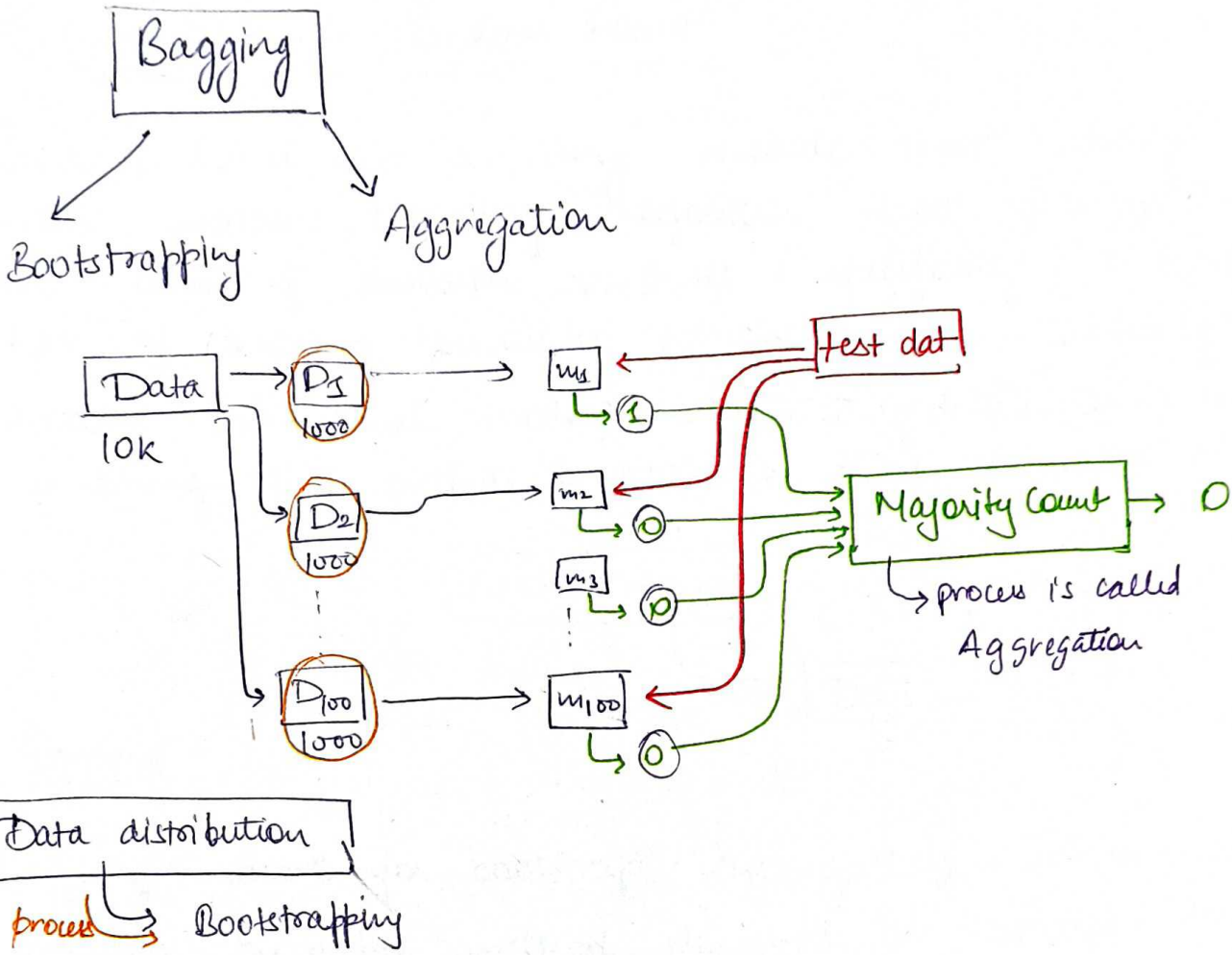
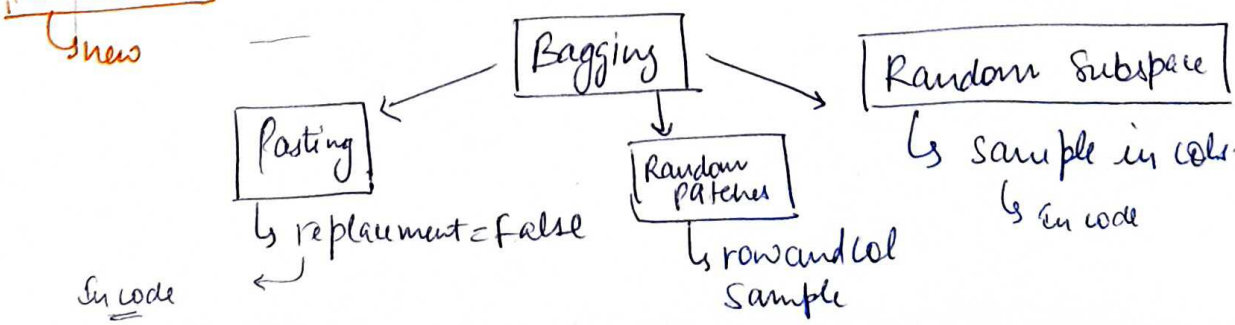
Bagging Ensemble



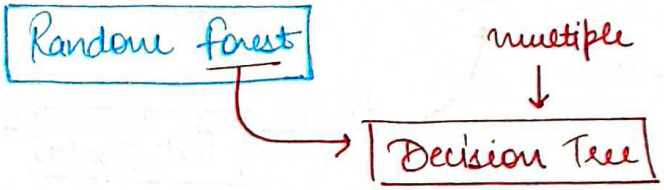
Bias Variance



Random Forest

Introduction to Random Forest

Random Forest is a very versatile and widely used machine learning algorithm that belongs to the class of ensemble methods. Specifically, it is a type of bagging technique, which involves training many individual models (in this case, decision trees) and combining their outputs to make a final prediction.



Bagging

Bagging, short for bootstrap aggregating, is a machine learning ensemble method designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also helps to avoid overfitting. The key principle of bagging is to generate multiple subsets of the original data (with replacement), train a seperate model for each subset and then combine the results.

Random Forest Intuition

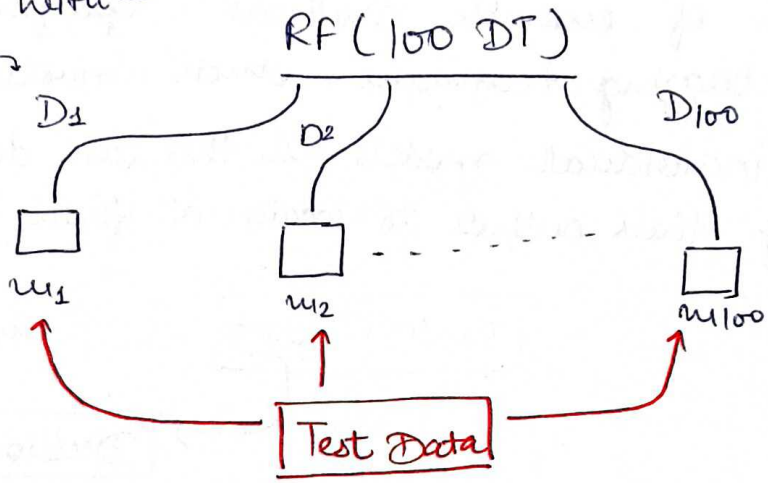
Random Forest → base model → Decision Tree

Data → 100 rows / 5 cols (classification)

↓

rows = 500 rows with replacement

RF (100 DT)



60 → 0
40 → 1
} → 0 majority count

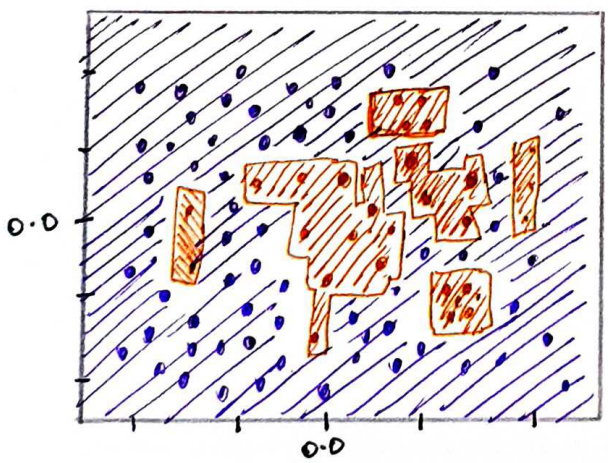
* In Regression, we find mean of the prediction accuracy of the model individualy.

Why Random Forest Works well?

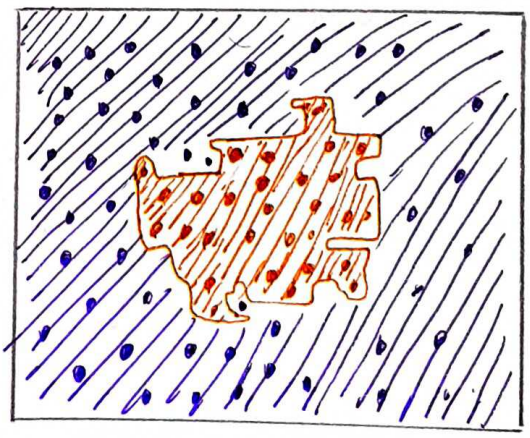
↓

[Wisdom of the Crowd]

Decision Tree



Random Forest



Bias - Variable Tradeoff

↓

Bias ∝ $\frac{1}{\text{Variance}}$

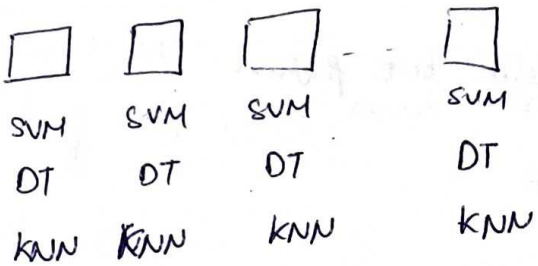
LB, HV

↓

Random Forest → LB, LV

Bagging Vs Random Forest

1) **Bagging** → any ML algorithm



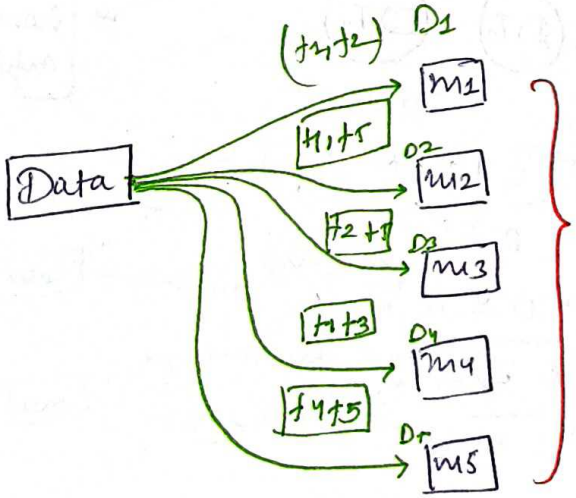
Random Forest → Decision Tree Model

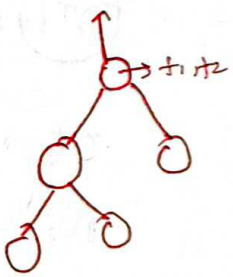
Col samp = 50%

[feature decided before tree made]

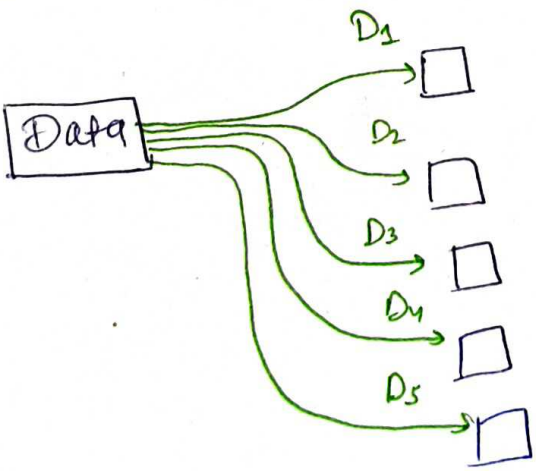
2) Bagging →

f1 | f2 | f3 | f4 | f5

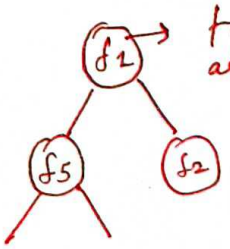




Random Forest →



Col sample = 50%



feature select at node level

* select cols 2 at node level.

Feature Importance

Data → f1 f2 f3 f4 f5

↓

RF

↓

f1 = 0.2
f2 → 0.3
f3 → 0.25
f4 → 0.15
f5 → 0.10

} select best feature

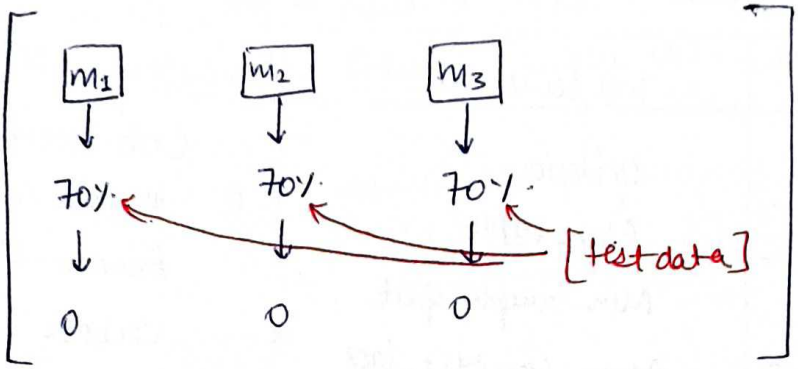
DT1 DT2 DT3

* how to find feature importance explained in decision Tree

DT1 → f1 → 0.1
DT2 → f1 → 0.3
DT3 → f1 → 0.4

0.8/3

Why Ensemble Techniques work?



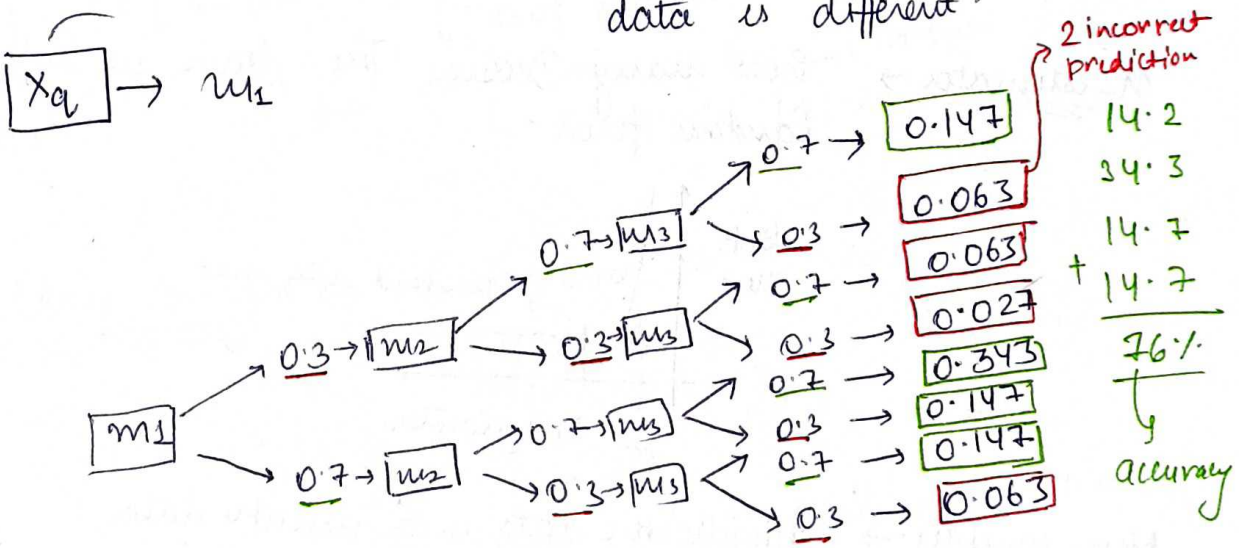
* So, we check whether the accuracy of ensemble learning greater than single model ($EL > m_1$) ($EL > m_2$)

→ **Assumption**

↳ m_1, m_2 and m_3 are independent model

↓

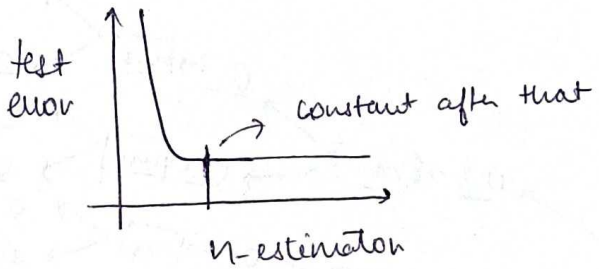
* If data is same but model is different
* If model is same but data is different.



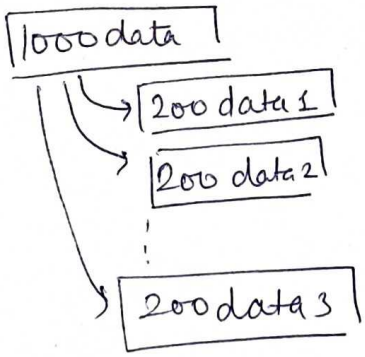
Random Forest Hyperparameters

Forest Level HP	Tree Level HP	Miscellaneous HP
n_estimator	Criterion	Oob_score
Max features	Max_depth	n_jobs
bootstrap	Min_sample_split	Random_state
Max_samples	Min_samples_leaf	Verbose
	Min_weight_fraction_leaf	Warm_start
	Max_leaf_nodes	Class weight
	Min_impurity_decrease	
	CLp_alpha	

n_estimator → How many Decision Tree train in side Random forest.



Max_samples → Divide the data into multiple data



Row sample

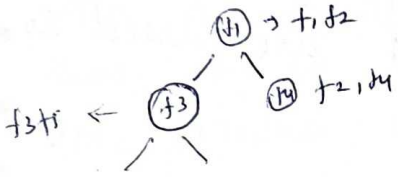
Bootstrap → Row sampling with Replace or without replace

Bootstrap → True → repeat value in dataset

Bootstrap → false → Unique value in dataset

Max_features → column sampling

5 columns → f1 f2 f3 f4 f5



Max_feature → sqrt(5) → No. of feature

↘ log(5) ↗

Warm_start → training time is higher

RF(n_estim= 10) → 20 } not restart the process
it start the process where it stop
the process (11-20)

True ↗

Class_weight → imbalanced dataset

95% 5%

↓ ↓

1 0

class weight give equal weight both even data is imbalanced

OOB Score

"OOB" stands for "out of bag". In the content of machine learning an out-of-bag score is a method of measuring the prediction error of random forests bagging classifiers, and other ensemble methods that use bootstrap aggregation (bagging) where sub-samples of the training dataset are used to train individual models.

Here's how it works:

1.) Each tree in the ensemble is trained on a distinct bootstrap sampling, some samples from the dataset will be left out during the training of each tree. These samples are called "out-of-bag" samples.

2.) The out of bag samples can then be used as a validation set. We can pass them through the tree that didn't see them during training and obtain predictions.

3.) These predictions are the then compared to the actual values to compute an "out-of-bg score". which can be thought of as an estimate of the prediction error on unseen data.

One of the advantages of the out-of-bag score is that it allows us to estimate the prediction error without needing a separate validation set. This can be particularly useful when the dataset is small and patitioning it into training and validation set might leave too few samples for effective learning.

Extremely Randomized Trees

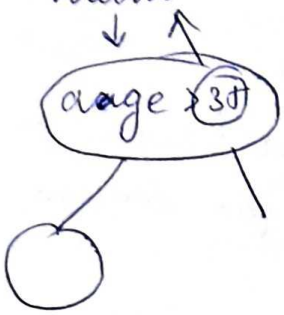
Extra Trees is short for "Extremely Randomized Trees". It's a modification of the Random Forest algorithm that changes the way the splitting points for decision tree branches are chosen.

In traditional decision tree algorithms (and therefore in Random Forests), the optimal split point for each feature is calculated, which involves a degree of computation. For a given node, the feature and the corresponding optimal split point that provide the best split are chosen. On the other hand, in the Extra Trees algorithm, for each feature under consideration, a split point is chosen completly at random. The best performing feature and its associated randomness to the model, hence the name "Extremely Randomized Trees"

Because of this difference, Extra Trees tend to have more branches (be deeper) than Random Forests, and the splits are made arbitarily. This can sometimes lead to models that perform better, especially on tasks where the data may not have clear optimal split points. However, like all models, whether Extra Trees will outperform Random Forests (or any other algo) depends on the specific dataset and task

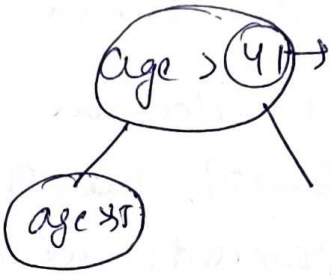
age	insurance
27	N
43	Y

ascending order → mean

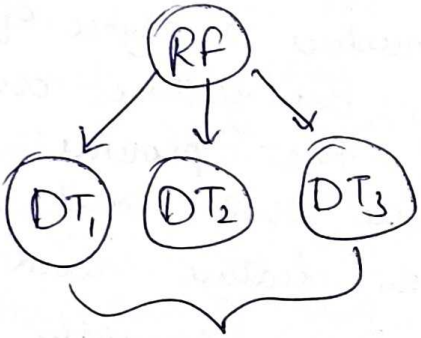


← Decision Tree

Extremly Randomised Trees



Random → high bias, training accuracy ↓, accuracy change



Extra Trees
Extra Randomness
→ independent (decorrelation)

* Sometimes good work

Advantages

Robustness to Overfitting Random Forests are less prone to overfitting compared to individual decision trees, because they average the results from many different trees, each of which might overfit the data in a different way.

Handling Large Datasets: They can handle large datasets with high dimensionality effectively.

Less pre-processing: Random Forest can handle both categorical and numerical variables without the need for scaling or normalization. They can also handle missing values.

Variable Importance: They provide insights into which features are most important in the prediction.

Parallelizable: The training of individual tree can be parallelized, as they are independent of each other. This speeds up the training process.

Non-Parametric: Random Forests are non-parametric, meaning they make no assemptions about the function form of the transformation from inputs to output. This makes them very flexible and able to model complex, non-linear relationship.

Disadvantages

- **Model Interpretability**: One of the biggest drawbacks of Random Forest is that they lack the interpretability of simpler models like linear regression or decision trees. While you can rank features by their importance, the model as a whole is essentially a black box.
- **Performance with Unbalanced Data**: Random Forest can be biased towards the majority class when dealing with unbalanced datasets. This can sometimes be migrated by balancing the dataset prior to training.
- **Predictive Performance**: Although Random Forests generally perform well, they may not always provide the best predictive performance. Gradient Boosting machine for instance, often outperform Random Forests. If the ~~data~~ ~~be~~ relationship within the data are linear, a linear model will likely perform better than a Random Forests.
- **Inefficiency with Sparse Data**: Random Forests might not be the best choice for sparse data or text data where linear model or other algo might be more suitable.

- **Parameter Tuning:** Although Random Forests requires less tuning than some other models, there are still several parameters (like the number of trees, tree depth etc) that can affect model performance and need to be optimized.

- **Difficulty with High Cardinality Features:** Random Forests can struggle with higher cardinality categorical features (features with a large number of distinct values). These types of features can lead to trees that are biased towards the variable with more levels and may cause overfitting.

- **Can't Extrapolate!-** This is because they do not predict beyond the range of the training data, and that they may not predict as accurately as other regression models.