# Ensemble Learning: Bagging and RandomForest

upGrad knowledgehut

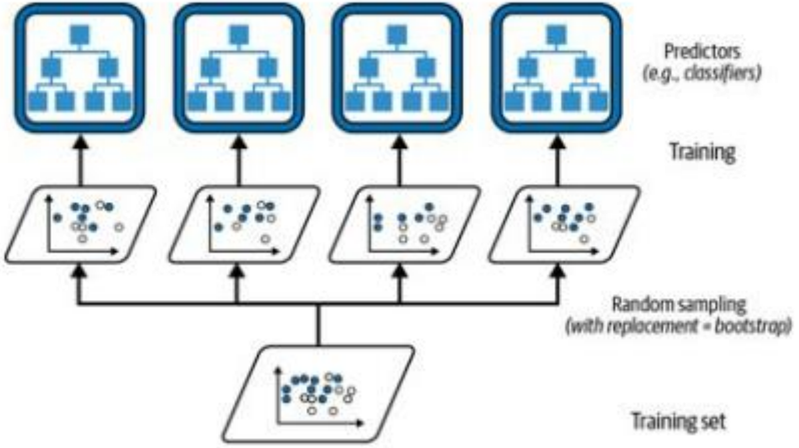
# Learning Objectives

- Describe the ensemble learning
- Discuss the construction of random forest
- Demonstrate random forest in Python using sklearn
- Explore hyperparameter and tuning for random forest
- Explain the feature importance in random forest

# Introduction to Ensemble Learning

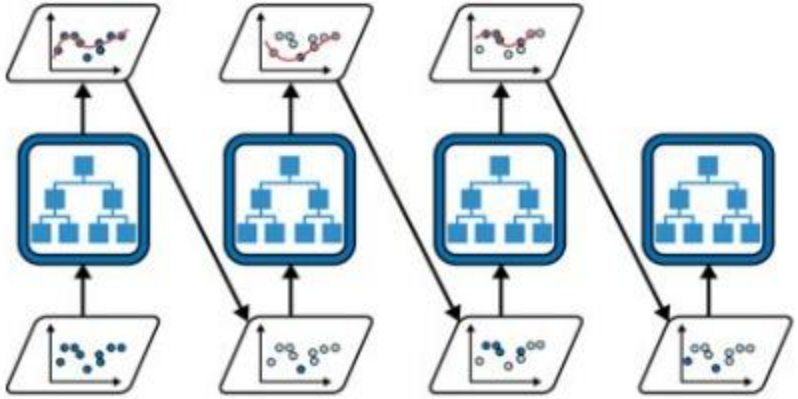
# Bagging

Combine multiple learners by individually training various models, then combine the results via an aggregation procedure.



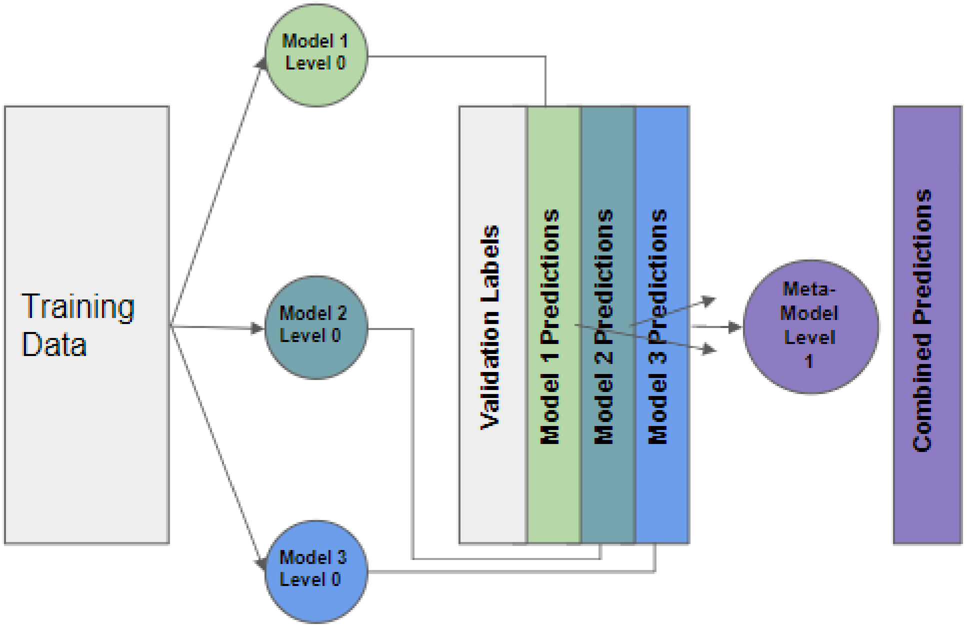
# Boosting

Sequentially add models together to an ensemble, each one correcting the mistakes of the previous one.



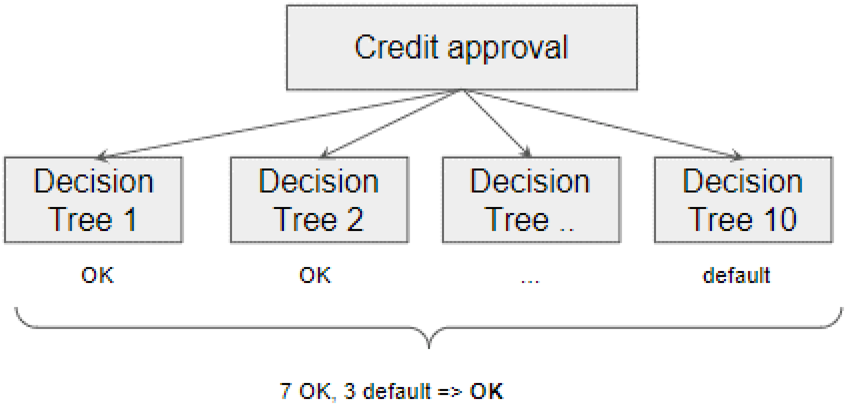
# Stacking

**Level-0 models**: Well-performing models trained on training data.
**Level-1 model**: Meta-model that learns the best combination based on level-0 predictions and true labels.



# Construction of Random Forest

# Construction Random Forest

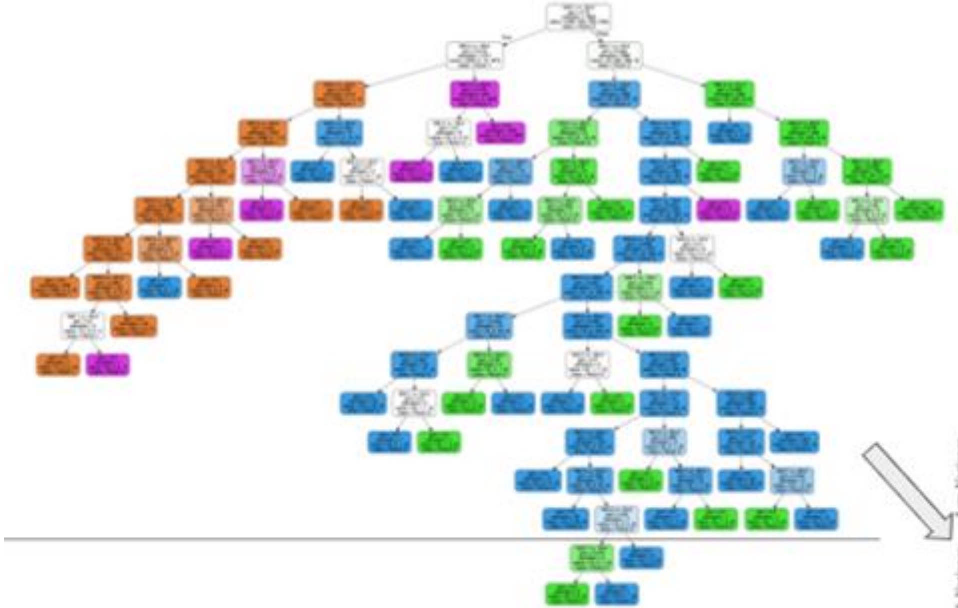


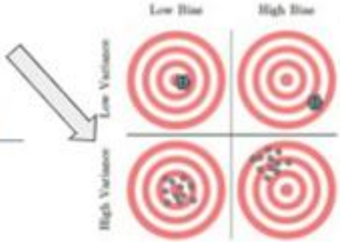

How is a random forest model created?

Many models are less wrong than single models.

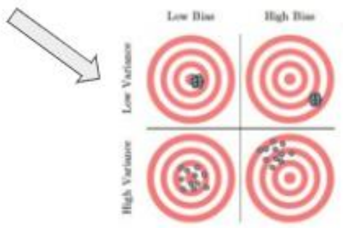
Goal: Control variance of decision trees.

# Controlling Variance in Decision Trees



# Add Variance to Decision Trees


One tree has high bias, and almost no variance.
Many trees have lower bias, higher variance than a single tree.

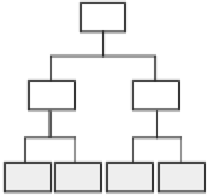
# Bootstrapping

N = 3

| | debt | assets | price | status |
|---|---|---|---|---|
| 0 | 500 | 2500 | 1250 | OK |
| 1 | 250 | 4500 | 1500 | OK |
| 2 | 500 | 2500 | 1250 | OK |
| 3 | 1000 | 4000 | 4500 | default |

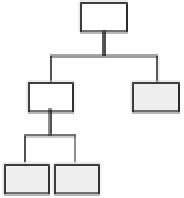
N = 1
sample(features, 2)

| | debt | assets | status |
|---|---|---|---|
| 0 | 500 | 2500 | OK |
| 1 | 250 | 4500 | OK |
| 2 | 500 | 2500 | OK |
| 3 | 1000 | 4000 | default |



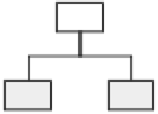

N = 2
sample(features, 2)

| | debt | price | status |
|---|---|---|---|
| 0 | 500 | 1250 | OK |
| 1 | 250 | 1500 | OK |
| 2 | 500 | 1250 | OK |
| 3 | 1000 | 4500 | default |



N = 3
sample(features, 2)

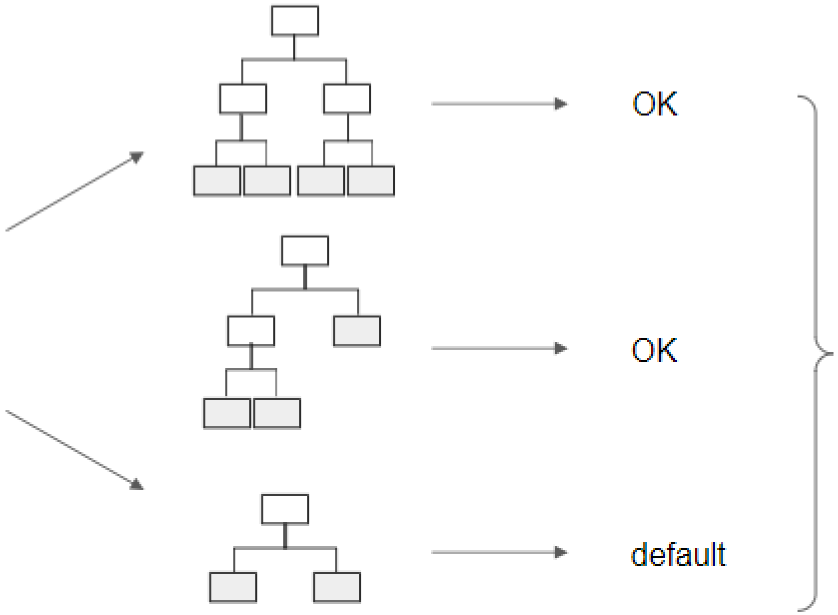
| | assets | price | status |
|---|---|---|---|
| 0 | 2500 | 1250 | OK |
| 1 | 4500 | 1500 | OK |
| 2 | 2500 | 1250 | OK |
| 3 | 4000 | 4500 | default |



# Aggregation

Prediction

| | debt | assets | price |
|---|---|---|---|
| 4 | 0 | 5000 | 5000 |




Combined Outcome (Vote)

**OK**

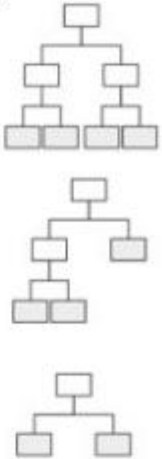
# Bagging


Bootstrapping

| | debt | assets | status |
|---|---|---|---|
| 0 | 500 | 2500 | OK |
| 1 | 250 | 4500 | OK |
| 2 | 500 | 2500 | OK |
| 3 | 1000 | 4000 | default |

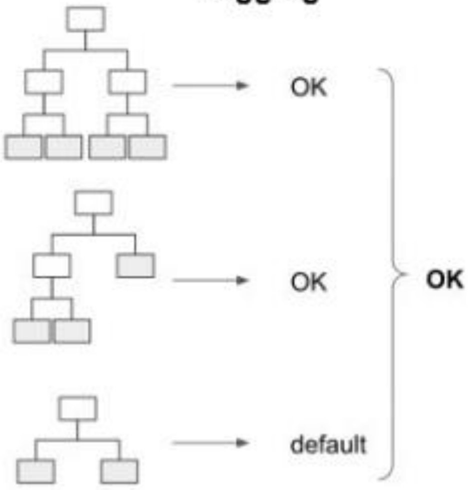
| | debt | price | status |
|---|---|---|---|
| 0 | 500 | 1250 | OK |
| 1 | 250 | 1500 | OK |
| 2 | 500 | 1250 | OK |
| 3 | 1000 | 4500 | default |

| | assets | price | status |
|---|---|---|---|
| 0 | 2500 | 1250 | OK |
| 1 | 4500 | 1500 | OK |
| 2 | 2500 | 1250 | OK |
| 3 | 4000 | 4500 | default |



Aggregating

Bagging



# Hyper Parameters and Tuning for Random Forest

# Important Hyper Parameters

## Number of decision trees

- Specifies the number of independent decision trees in your ensemble.
- Higher value usually result in better / more stable predictions, since errors average out.

## Maximum depth of trees

- Specifies the maximum depth a tree in the ensemble can have.
- Rule of thumb: Deeper trees give better accuracy but increase the risk of overfitting.

## Minimum leaf size

- Determines the smallest size of a leaf node in the ensemble.
- Too many leaves can cause overfitting and poor model generalization.

# Tunning Hyperparameter

## Typical approach

- Split data in training and validation set to find best hyperparameters with cross validation.

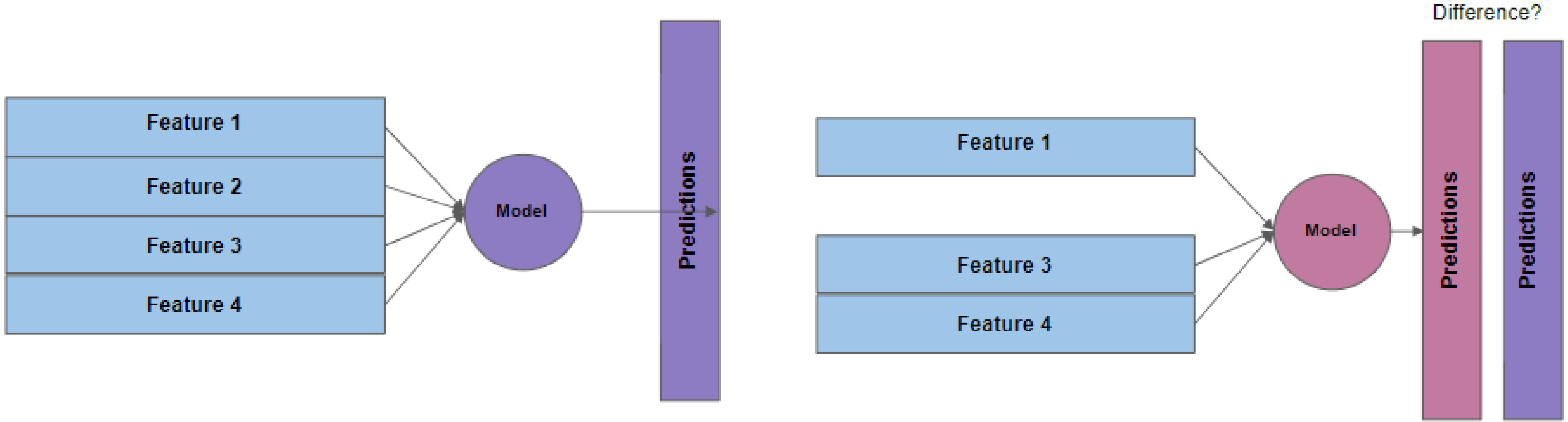
## Random Forest

- Bootstrapping phase generates hold-out data automatically (no tree sees the full dataset).

# Feature Importance in Random Forest

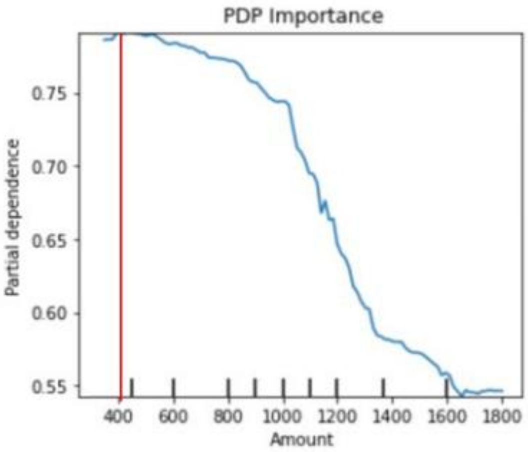
# Interpretation and Feature Importance

Which feature led to the prediction?

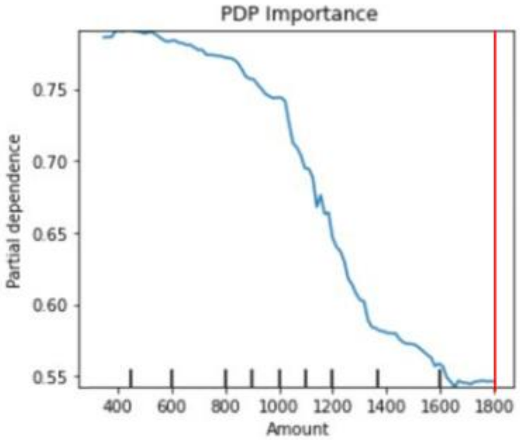


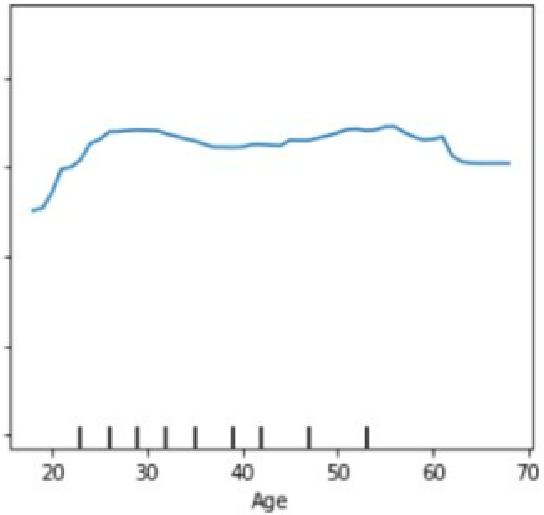

**Global methods:**
Describe how features affect the prediction on average

# Partial Dependencies Plot




Graphical representation showing the marginal effect of one feature on the outcome.



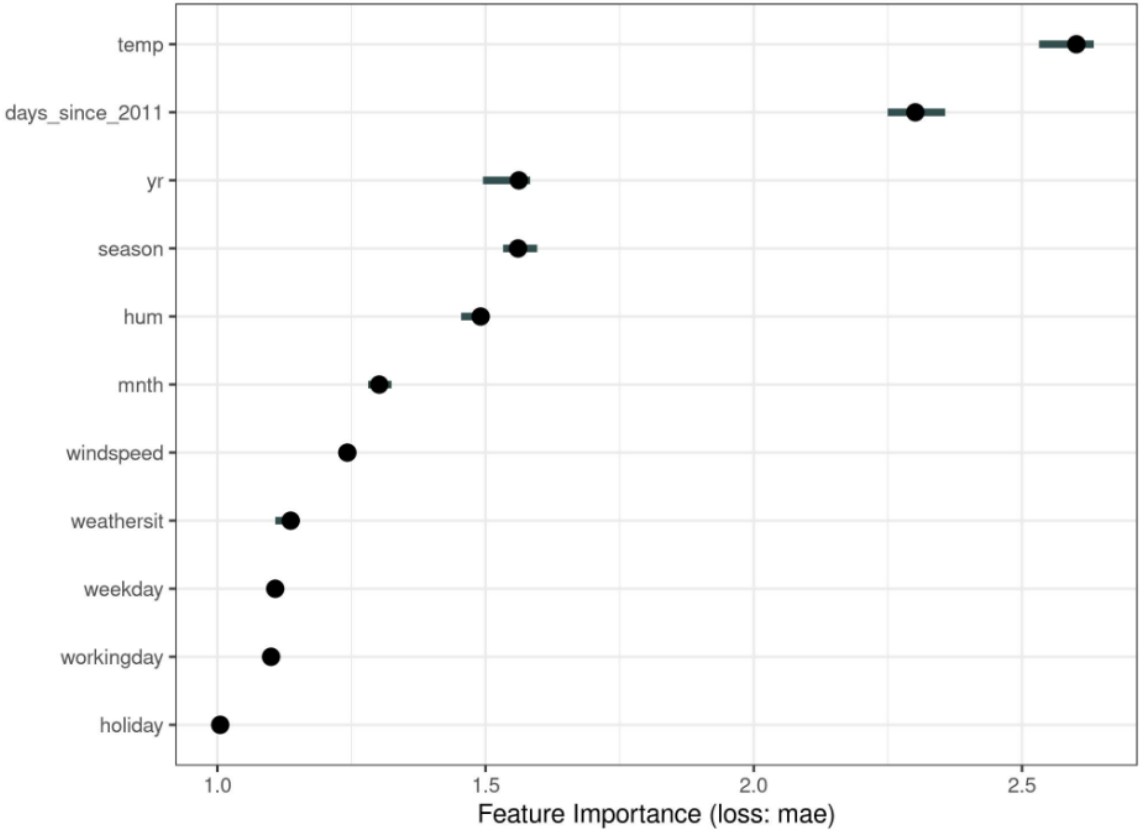
# Permutation Feature Importance

**The permutation feature importance algorithm based on Fisher, Rudin, and Dominici (2018):**

Input: Trained model $\hat{f}$, feature matrix $X$, target vector $y$, error measure $L(y, \hat{f})$.

1. Estimate the original model error $e_{orig} = L(y, \hat{f}(X))$ (e.g. mean squared error)
2. For each feature $j \in \{1, \ldots, p\}$ do:
   - Generate feature matrix $X_{perm}$ by permuting feature j in the data X. This breaks the association between feature j and true outcome y.
   - Estimate error $e_{perm} = L(Y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.
   - Calculate permutation feature importance as quotient $FI_j = e_{perm}/e_{orig}$ or difference $FI_j = e_{perm} - e_{orig}$
3. Sort features by descending FI.

# Permutation Feature



# Thank you