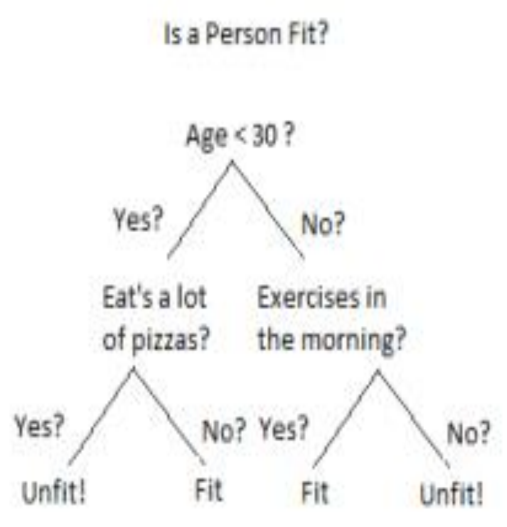
Unit 3

Tree Based Models

Decision Trees

- Supervised Machine Learning Algorithm
- The data is continuously split according to a certain parameter
- Two Entities: Decision nodes and leaves.
- The leaves are the decisions or the final outcomes.
- And the decision nodes are where the data is split.
- Types of Decision Trees :Classification Trees & Regression Trees.
- Classification Trees: Outcome variable is a class label
- Regression Trees: Outcome variable is a continuous number
- Algorithms: CART (classification and Regression Trees),

  ID3 (Iterative Dichotomiser 3), c4, c5, CHAID



Terminologies with DT

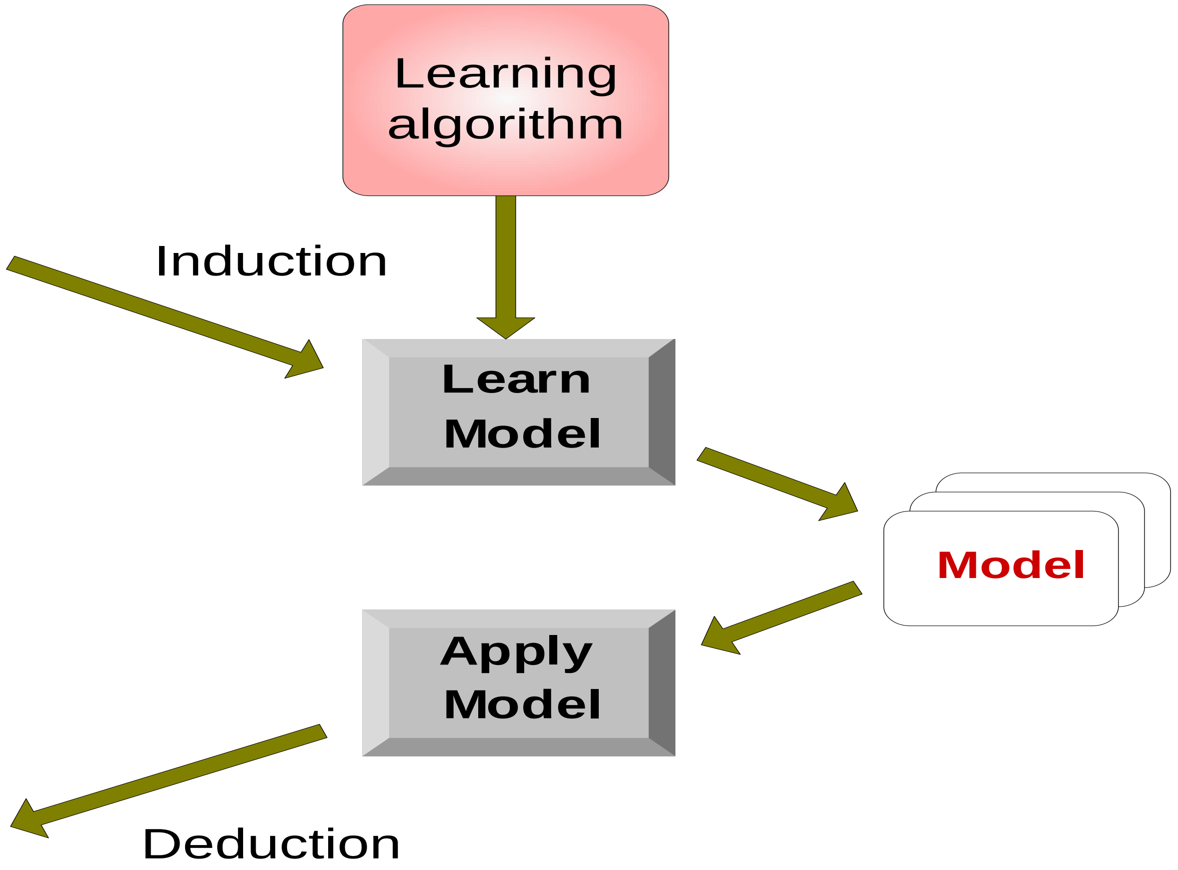
- **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
- **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.

- **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
- **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.

Illustrating Classification Learning

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



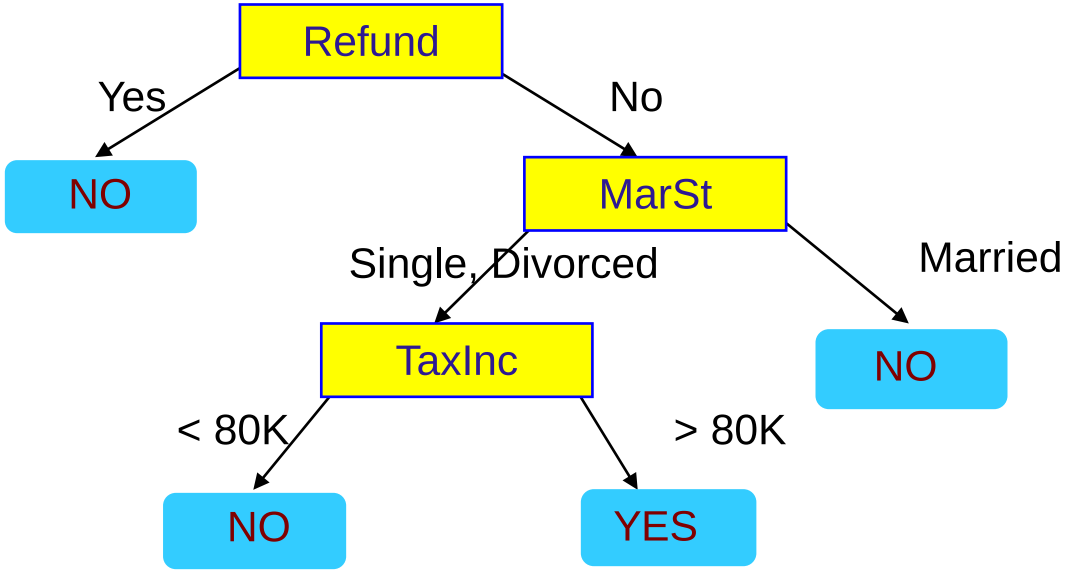
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Example of a Decision Tree

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

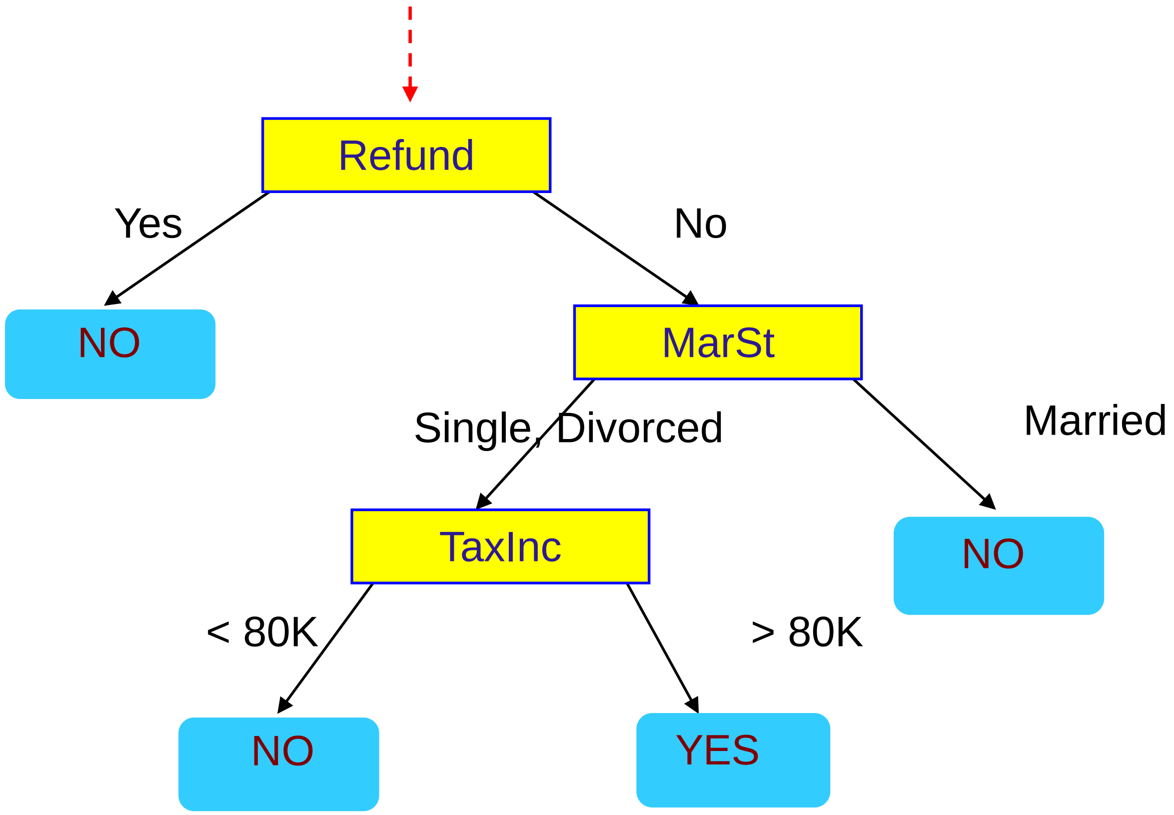
Training Data



Model: Decision Tree

Apply Model to Test Data

Start at the root of tree



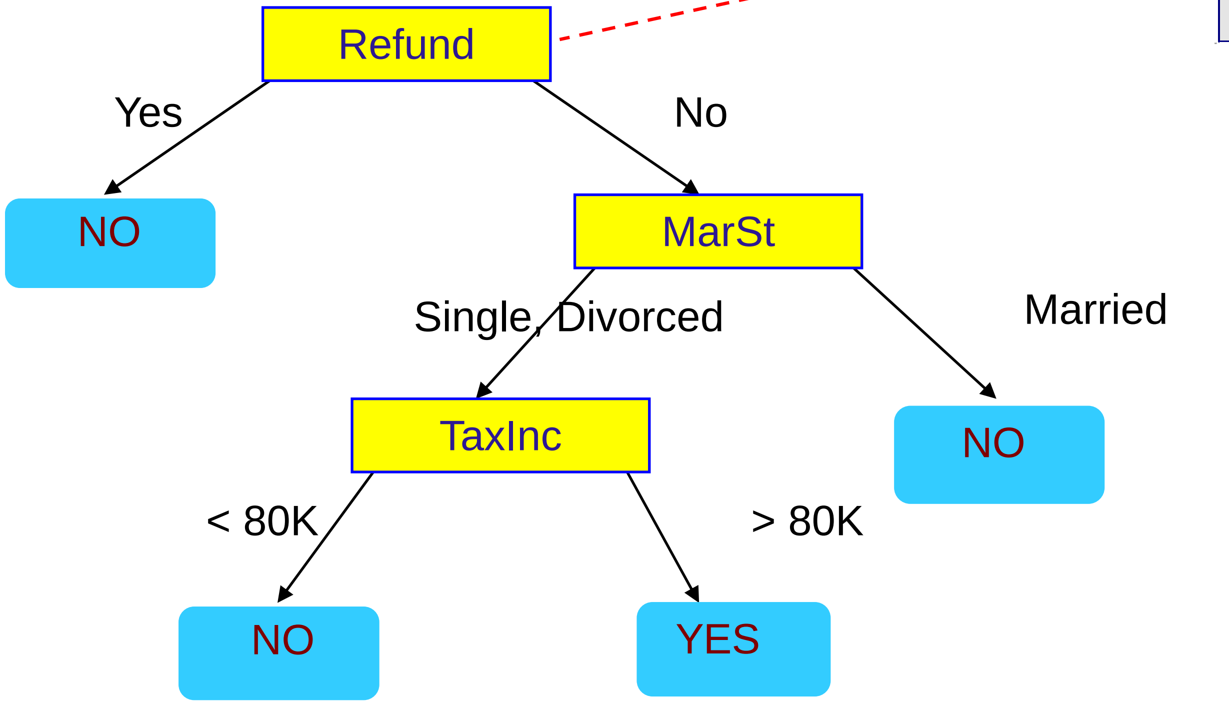
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

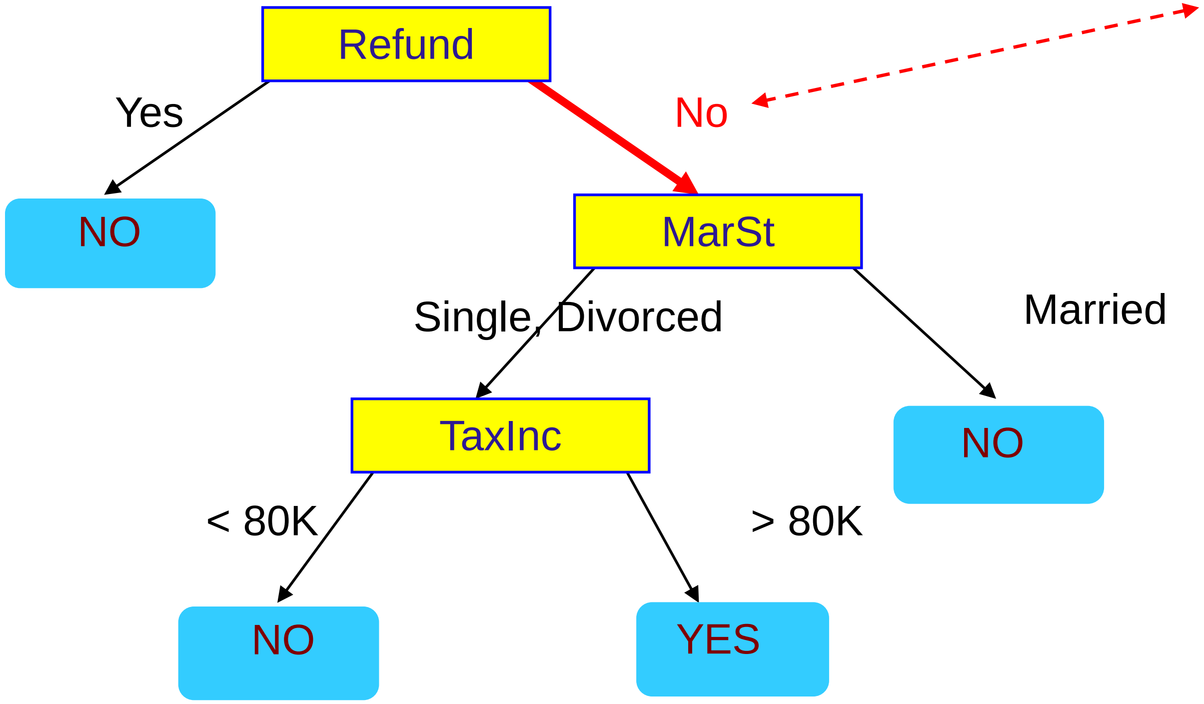
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

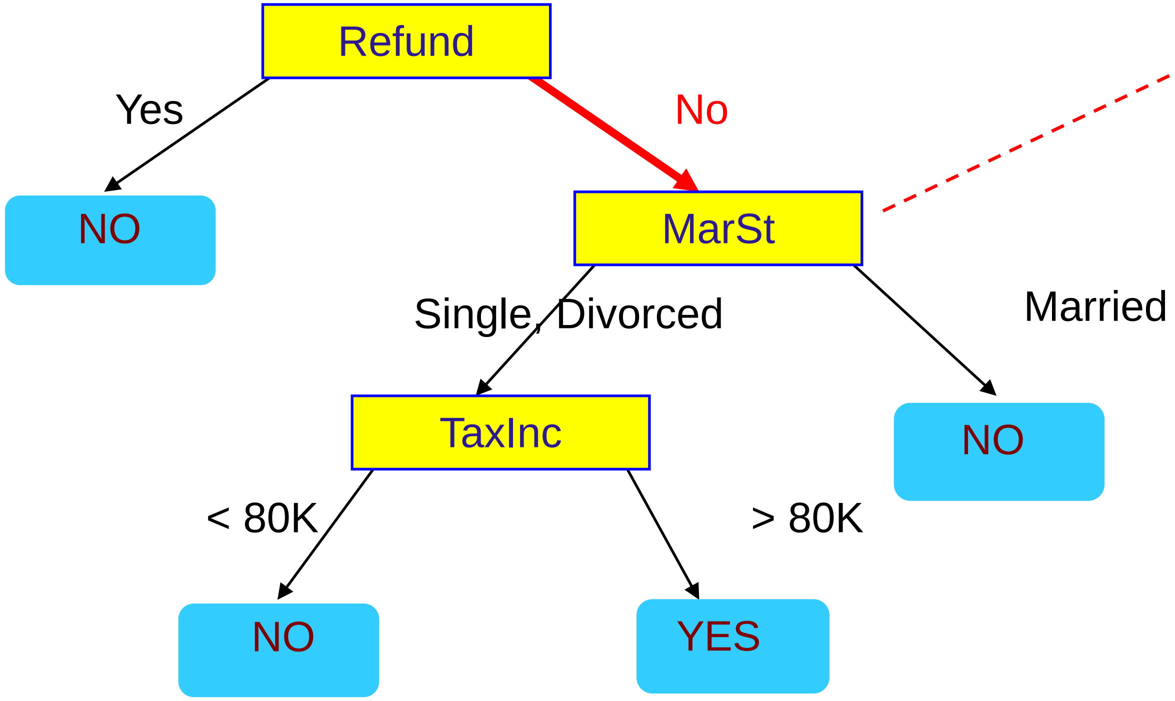
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

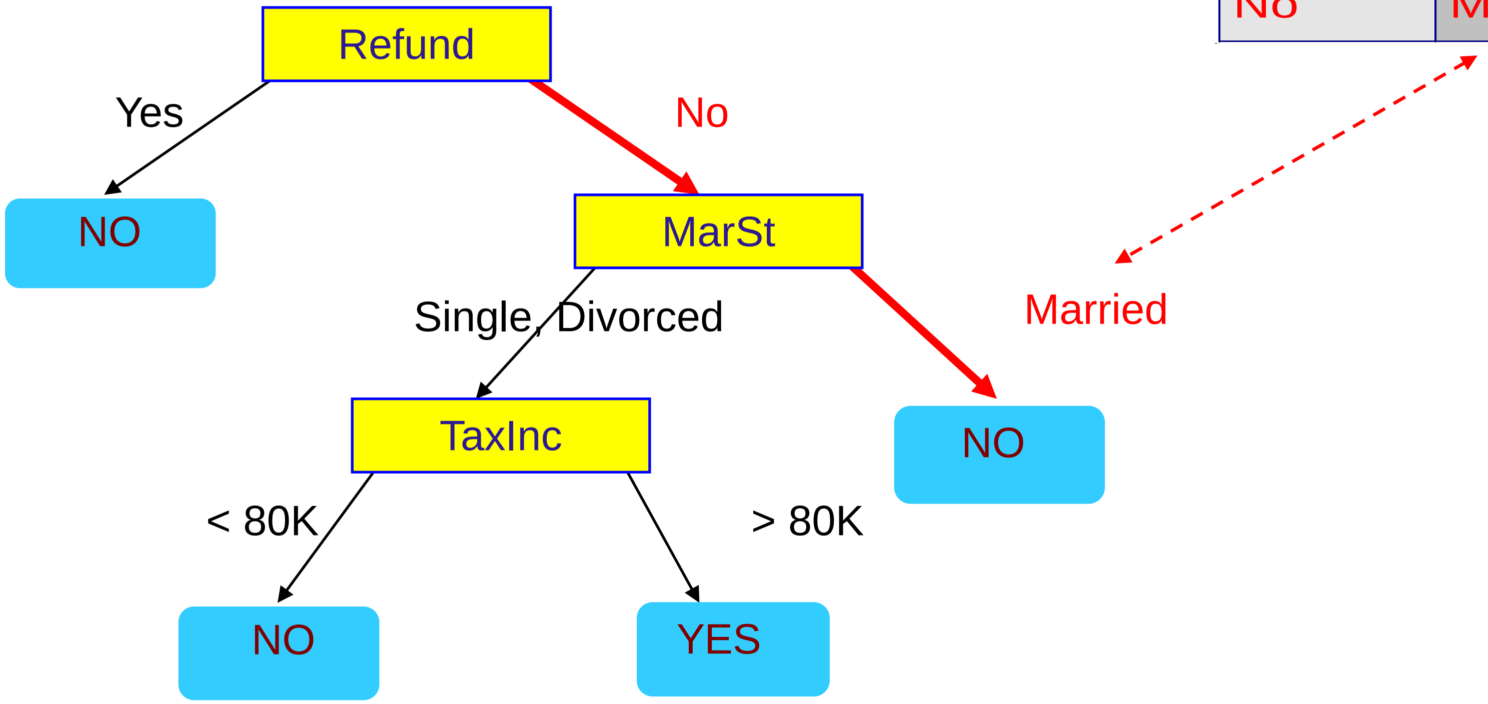
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

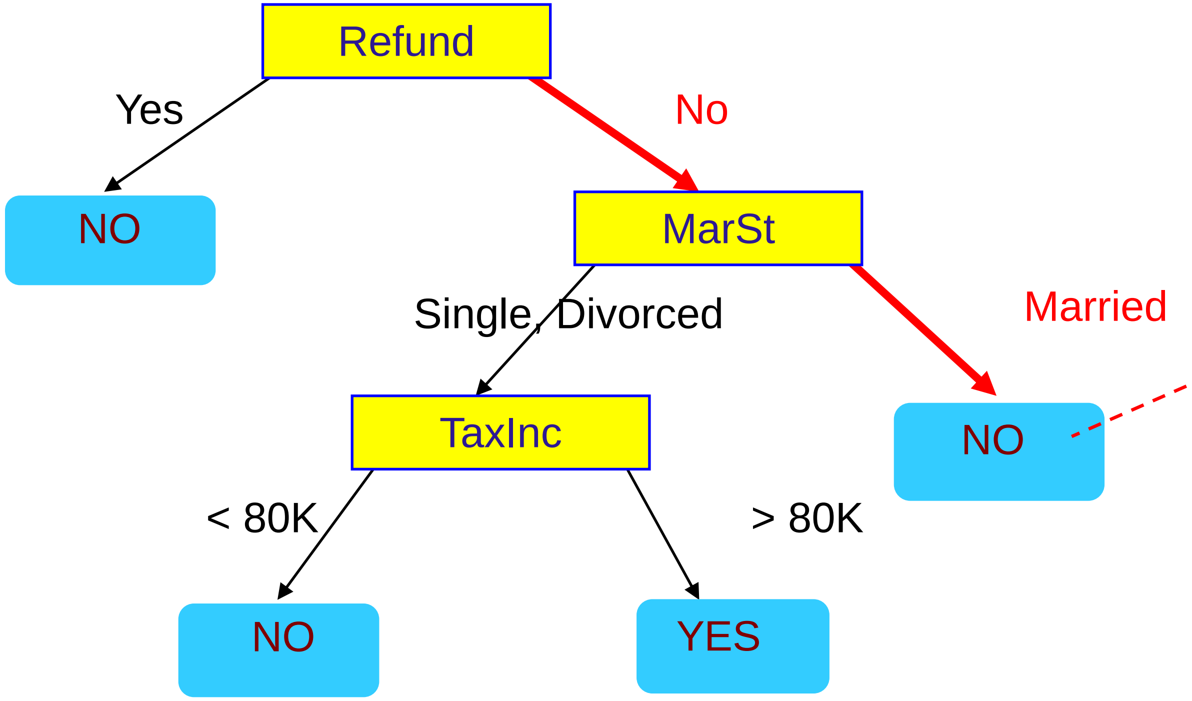
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to "No"

Algorithm

Data: Training data with m features and n instances

Result: Induced Decision Tree

Begin with an empty tree;

while stopping criterion is not satisfied do

        select best feature;

        split the training data on the best feature;

repeat last two steps for the child nodes resulting from the split;

end

How many possible Decision trees?

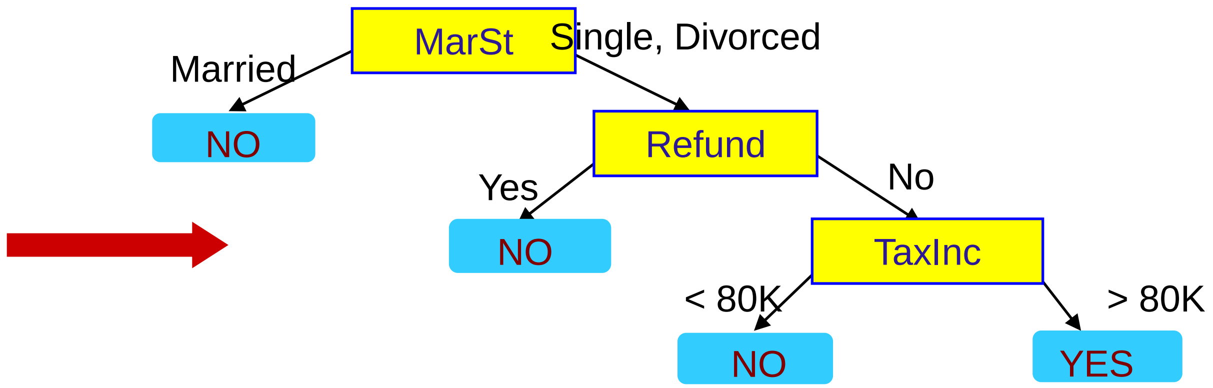
- If we consider only binary features and a binary classification task, then for a training dataset having m features there are at most 2^m possible instances. In this case there are 2^{2^m}

- The search space is very large
- Suppose we have 10 features 2^1024 possible decision trees

Non-Uniqueness

- Decision trees are not unique:
  - Given a set of training instances *T*, there generally exists a number of decision trees that are consistent with (or fit) *T*

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



How to start with split?

- Which feature is the best?
  - Measure the impurityness using
    - Entropy
    - Gini Index
- Information Gain - select the feature with the maximum information gain

Concept

- Entropy
  - *Entropy is a measure of disorder.*
  - *Entropy is an indicator of how messy your data is.*
  - It is the measure of the amount of uncertainty or randomness in data
  - It is the predictability of certain event
  - When there is no randomness, entropy is 0.
  - In particular, lower values imply less uncertainty while higher values imply high uncertainty.

$$\mathbf{H}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Concept

- Information Gain
  - It is the effective change in entropy after deciding on a particular attribute
  - It measures the relative change in entropy with respect to the independent variables.

$$IG(S,A) = H(S) - H(S,A)$$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Algorithm

- Create root node for the tree
- If all examples are positive, return leaf node ‘positive’
- Else if all examples are negative, return leaf node ‘negative’
- Calculate the entropy of current state H(S)
- For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by H(S, x)
- Select the attribute which has maximum value of IG(S, x)
- Remove the attribute that offers highest IG from the set of attributes
- Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

Dataset

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	**High**	Weak	No
D2	Sunny	Hot	**High**	**Strong**	No
D3	Overcast	Hot	**High**	Weak	Yes
D4	Rain	Mild	**High**	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	**Strong**	No
D7	Overcast	Cool	Normal	**Strong**	Yes
D8	Sunny	Mild	**High**	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	**Strong**	Yes
D12	Overcast	Mild	**High**	**Strong**	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	**High**	**Strong**	No

Working

- The initial step is to calculate H(S), the Entropy of the current state.
- For the given dataset, we can see in total there are 5 No's and 9 Yes's.

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$= 0.940$$

$= -(0.64 \log_2 0.64) - (0.36 \log_2 0.36)$

$= -0.64 * \log(0.64)/\log 2 - (0.36 * \log 0.36/\log 2)$

$= (-0.64 * -0.64) + (-0.36 * -1.47)$

$= 0.41 + 0.53 = 0.94$

- Entropy is 0 if all members belong to the same class, and 1 when half of them belong to one class and other half belong to other class that is perfect randomness. Here it's 0.94 which means the distribution is fairly random.

-contd

- Let’s start with wind.

$$IG(S,Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

- where ‘x’ are the possible values for an attribute. Here, attribute ‘Wind’ takes two possible values in the sample data, hence x = {Weak, Strong}

1. $H(S_{weak})$
2. $H(S_{strong})$
3. $P(S_{weak})$
4. $P(S_{strong})$
5. $H(S) = 0.94$ which we had already calculated in the previous example

-contd

- Week=8, strong=6
- P(x) is the probability of the event x.

$$P(S_{weak}) = \frac{Number\ of\ Weak}{Total}$$
$$= \frac{8}{14}$$
$$P(S_{strong}) = \frac{Number\ of\ Strong}{Total}$$
$$= \frac{6}{14}$$

- Now out of the 8 Weak examples, 6 of them were ‘Yes’ for Play Golf and 2 of them were ‘No’ for ‘Play Golf’. So, we have,

$$Entropy(S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)$$
$$= 0.811$$

-contd

- Similarly, out of 6 Strong examples, we have 3 examples where the outcome was 'Yes' for Play Golf and 3 where we had 'No' for Play Golf

$$Entropy(S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$
$$= 1.000$$

- Remember, here half items belong to one class while other half belong to other. Hence we have perfect randomness.
- Now we have all the pieces required to calculate the Information Gain,

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) * H(S_{weak}) - P(S_{strong}) * H(S_{strong})$$
$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$
$$= 0.048$$

-contd

- Information gain for wind is **0.048**. Now we must similarly calculate the Information Gain for all the features.

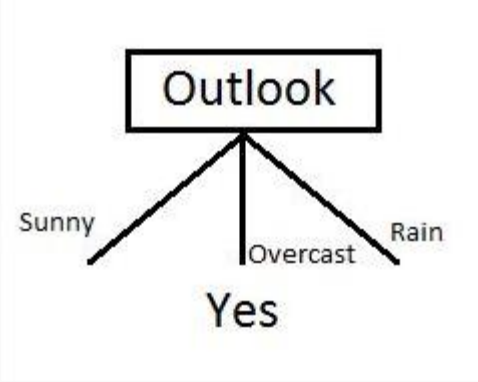
$$IG(S, Outlook) = 0.246$$

$$IG(S, Temperature) = 0.029$$

$$IG(S, Humidity) = 0.151$$

$$IG(S, Wind) = 0.048 \text{ (Previous example)}$$

- IG(S, Outlook) has the highest information gain of 0.246, **hence we chose Outlook attribute as the root node**. At this point, the decision tree looks like.



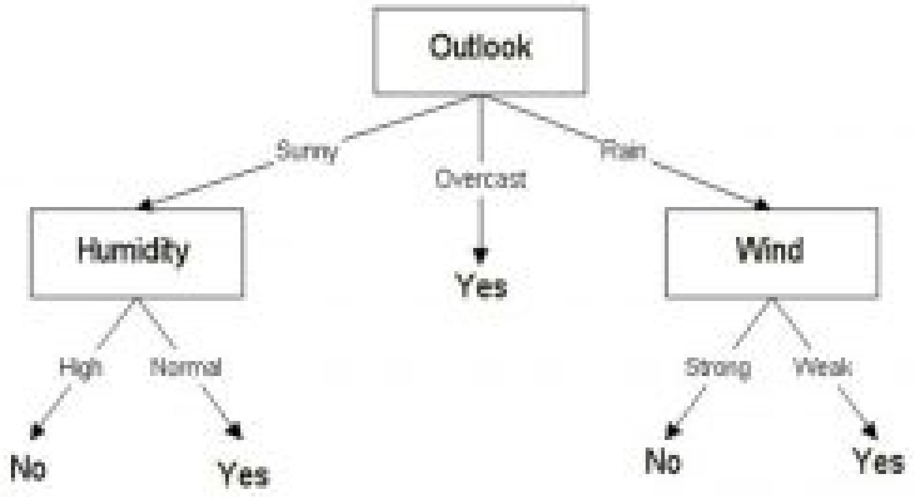
-contd

- we’ve got three of them remaining Humidity, Temperature, and Wind.
- And, we had three possible values of Outlook: Sunny, Overcast, Rain.
- The Overcast node already ended up having leaf node ‘Yes’, so we’re left with two subtrees to compute: Sunny and Rain.
- Next step would be computing $H(S_{sunny})$
- Sunny table looks like this

$$H(S_{sunny}) = \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.96$$

Temperature	Humidity	Wind	Play Golf
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Final Tree


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Gini Index

- Gini index

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Algorithms that uses Gini/Entropy

- CART – Gini Index
- ID3/c4.5 - Entropy

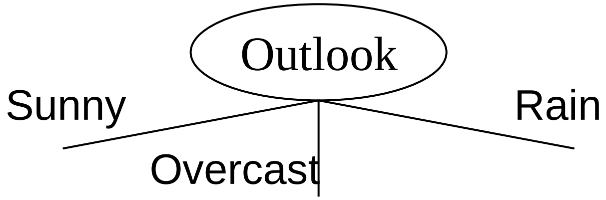
How to Specify Test Condition?

- Depends on attribute types
  - Categorical
  - Continuous

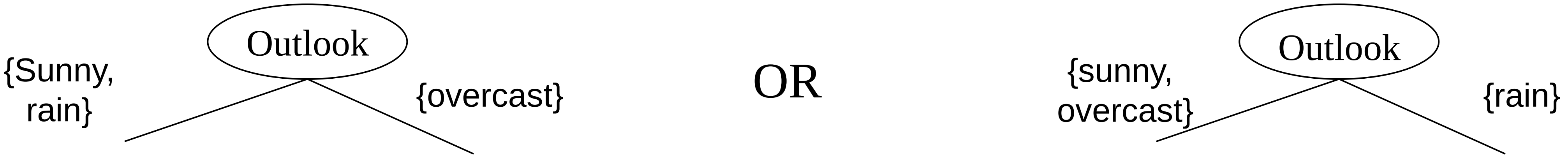
- Depends on number of ways to split
  - Binary split
  - Multi-way split

Splitting Based on categorical Values

- Multi-way split: Use as many partitions as values



- Binary split: Divide values into two subsets



Need to find optimal partitioning!

Splitting Based on Continuous Attributes

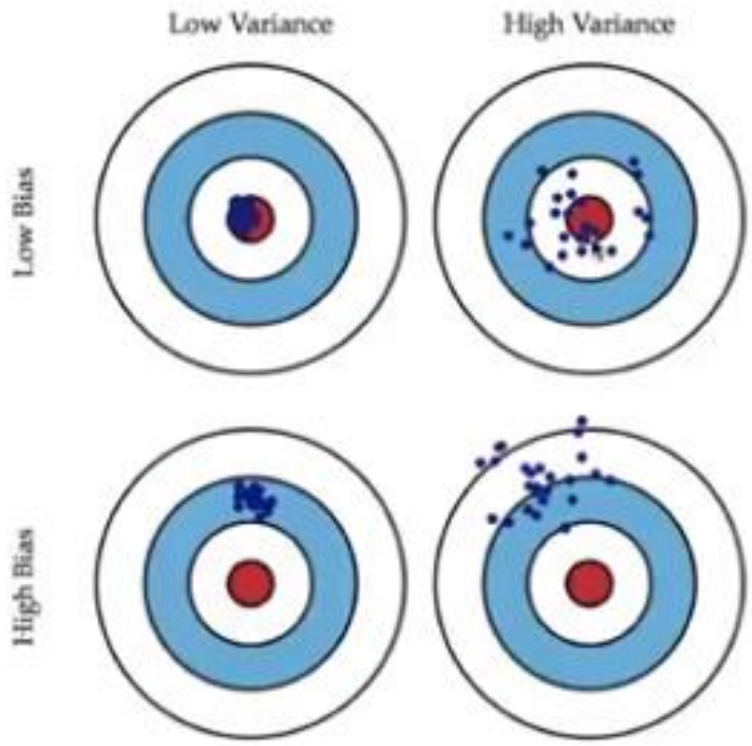
- Different ways of handling
  - Multi-way split:
    - Static – discretize once at the beginning
    - Dynamic – repeat on each new partition
  - Binary split: $(A < v)$ or $(A \geq v)$
    - How to choose v?

Need to find optimal partitioning!

Can use GAIN or GINI !

Bias and variance

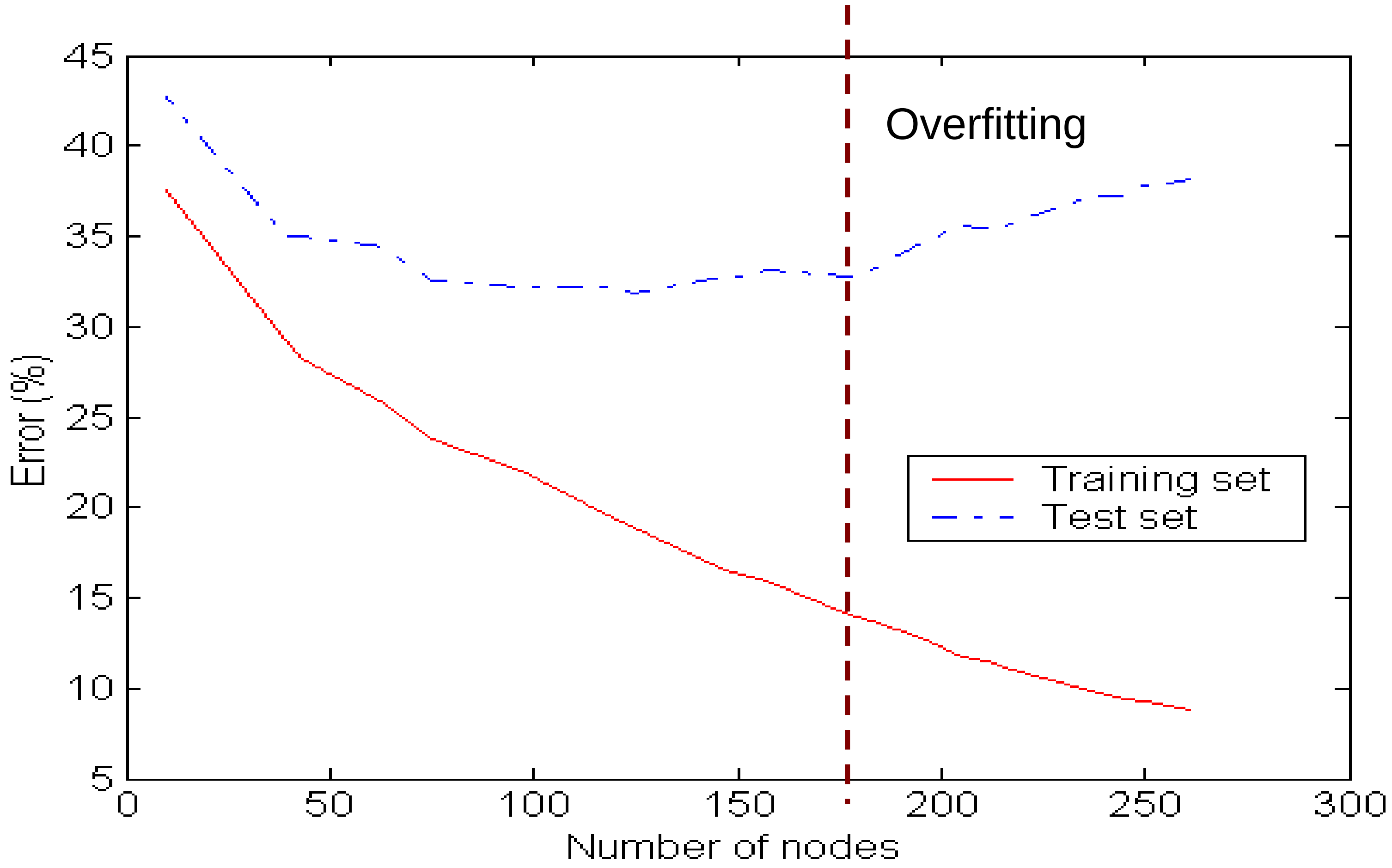
- Bias- how much on an average are the predicted values different from the actual value
- Variance – how different will the predictions of the model be at the same point if different samples are taken from the same population



Overfitting and Underfitting

- **Overfitting:**
  - Given a model space H, a specific model $h \in H$ is said to overfit the training data if there exists some alternative model $h' \in H$, such that h has smaller error than h' over the training examples, but h' has smaller error than h over the entire distribution of instances i.e, your machine recognition is worse
  - The model is too complex, which creates a noise in the model. Suppose if the model ends up making one leaf for each observation then, it is overfitting
- **Underfitting:**
  - The model is too simple, so that both training and test errors are large

Detecting Overfitting



Underfitting in Decision Tree Learning

- Underfitting happens when the decision tree is too simple
- Solution
  - Add more features

Overfitting in Decision Tree Learning

- Overfitting results in decision trees that are more complex than necessary
  - Tree growth went too far
  - Number of instances gets smaller as we build the tree (e.g., several leaves match a single example)

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Avoiding Tree Overfitting – Solution 1

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., GINI or GAIN)

Avoiding Tree Overfitting – Solution 2

- Post-pruning
  - Split dataset into training and validation sets
  - Grow full decision tree on training set
  - While the accuracy on the validation set increases:
    - Evaluate the impact of pruning each subtree, replacing its root by a leaf labeled with the majority class for that subtree
    - Replace subtree that most increases validation set accuracy (greedy approach)

DT Advantages

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Good accuracy
- Data type is not a constraint. Less data cleaning is required
- Non-parametric method – DT has no assumptions about the distribution space

DT Disadvantages

- Axis-parallel decision boundaries
- Redundancy
- Need to retrain with new data

When does DT work better?

- If there is a high non-linearity & complex relationship between dependent & independent variables, a tree model will outperform

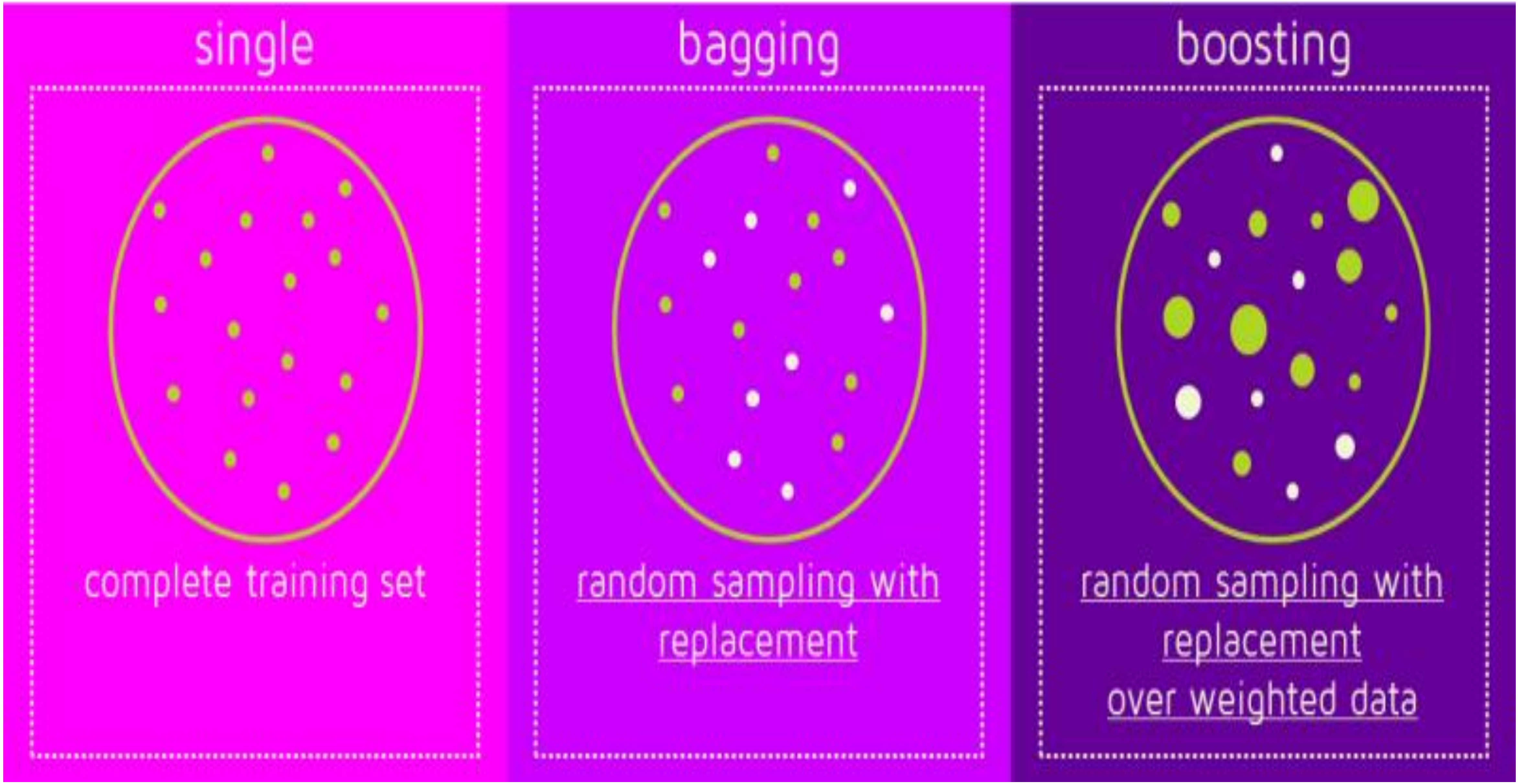
DT in Python

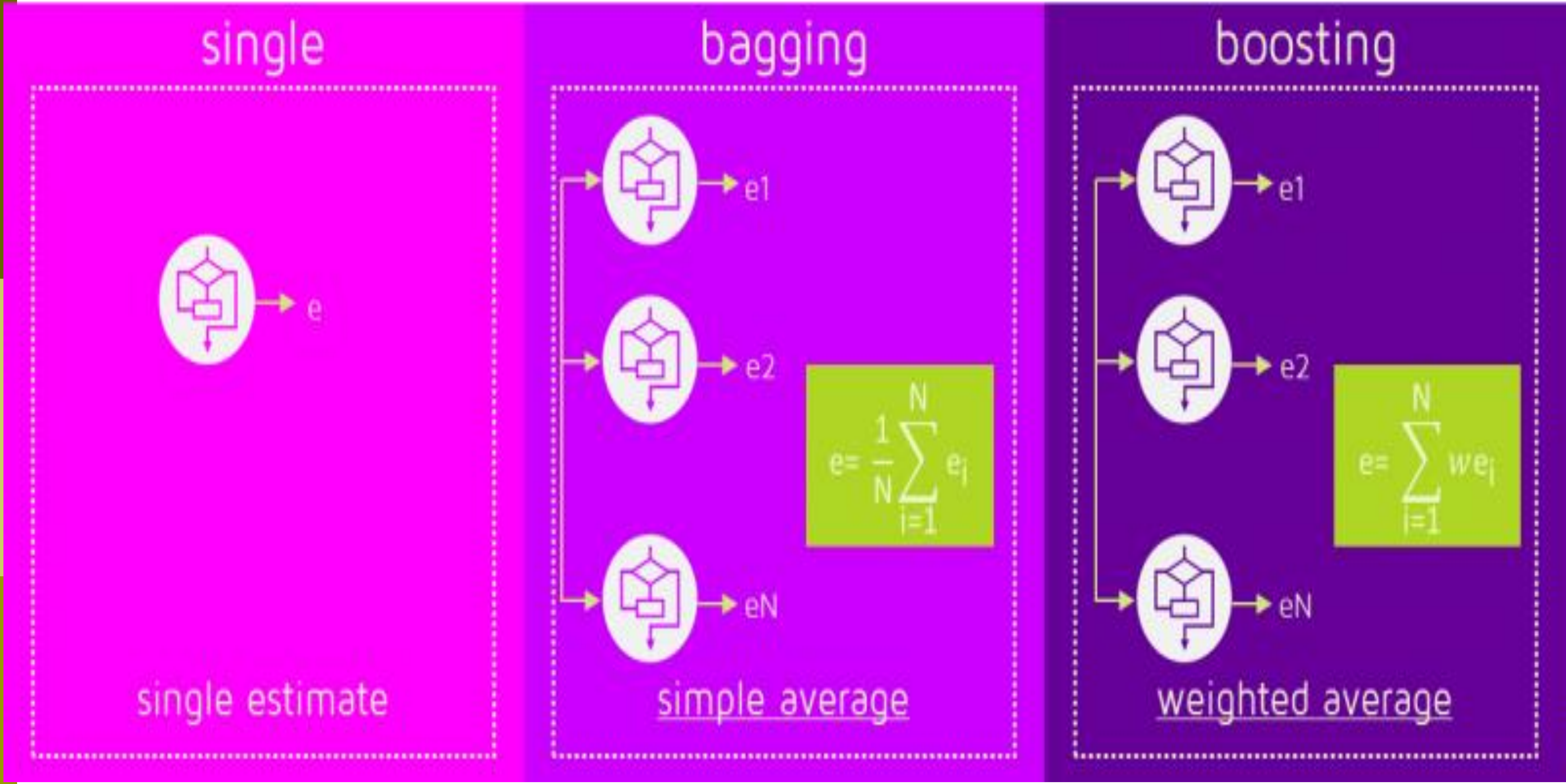
- DecisionTreeClassifier() – classification
- DecisionTreeRegressor()- Regression Tree
- Model.fit()
- Model.predict()
- DecisionTree Parameters:
  - Criterion(gini/entropy)
  - max_depth
  - min_samples_split

Ensemble methods

- Ensemble means group
- Ensemble methods can be used to boost your accuracy
- Different ensemble methods
  - Bagging – BaggingClassifier, RandomForest etc.
  - Boosting – adaboost, stochasticGradientBoost etc

- **Bagging**. Building multiple models (typically of the same type) from different subsamples of the training dataset.
- **Boosting**. Building multiple models (typically of the same type) each of which learns to fix the prediction errors of a prior model in the chain.


single
complete training set
bagging
random sampling with replacement
boosting
random sampling with replacement over weighted data

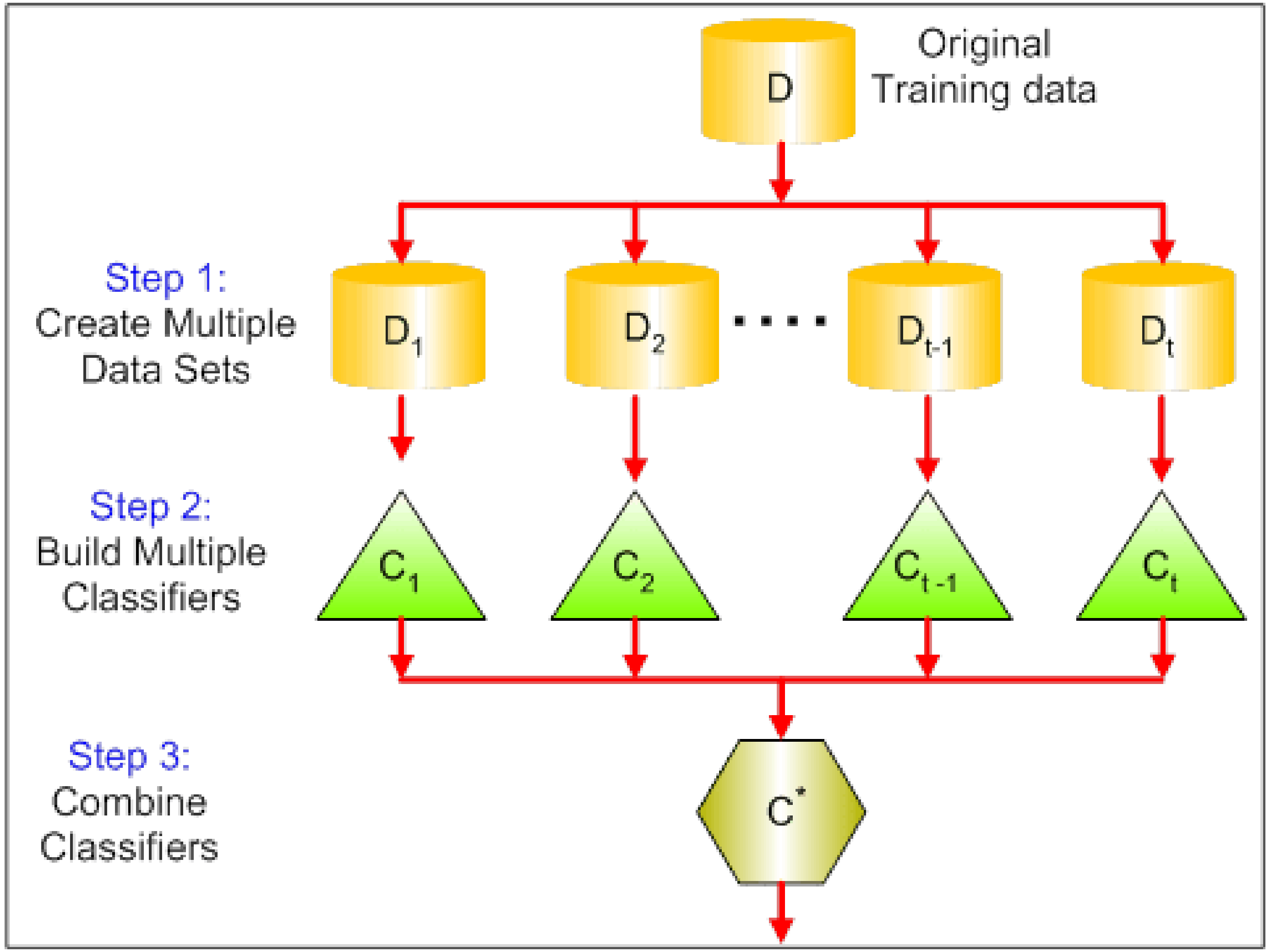

single
bagging
boosting
e
e1
e2
eN
$e = \frac{1}{N}\sum_{i=1}^{N} e_i$
e1
e2
eN
$e = \sum_{i=1}^{N} we_i$
single estimate
simple average
weighted average

Bagging

- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

- For classification, a *committee* of trees each cast a vote for the predicted class.
- The basic idea:
  - randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*
- Bagging is a technique used to reduce the variance of our predictions by combining the result of multiple classifiers modeled on different sub-samples of the same data set.

Bagging


D
Original
Training data
Step 1:
Create Multiple
Data Sets
D1
D2
Dt-1
Dt
Step 2:
Build Multiple
Classifiers
C1
C2
Ct -1
Ct
Step 3:
Combine
Classifiers
C*

Random Forest

- Random forest construct multiple decision trees at the training time
- Random forest can be the best method when overfitting happens in the decision tree implementation

- Random forest developed by aggregating trees
- Can be used for classification or regression
- Avoids overfitting
- Can deal with large number of features
- Helps with feature selection based on importance of variables
- User friendly: only 2 free parameters
  - Trees- ntree, default 500
  - Variables randomly sampled as candidates at each split – mtry,
  - Default is sqrt(p) for classification and p/3 for regression

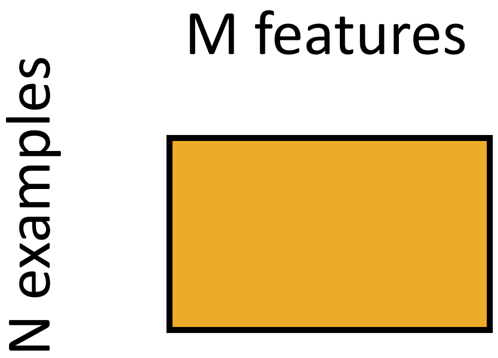
- P – represents the features in the dataset
- 3 steps
  - Draw ntree bootstrap samples
  - For each bootstrap sample, grow un-pruned tree by choosing best split based on a random sample of mtry predictors at each node.
  - Predict new data using majority votes for classification and average for regression based on ntree trees.

How Random forest is constructed?

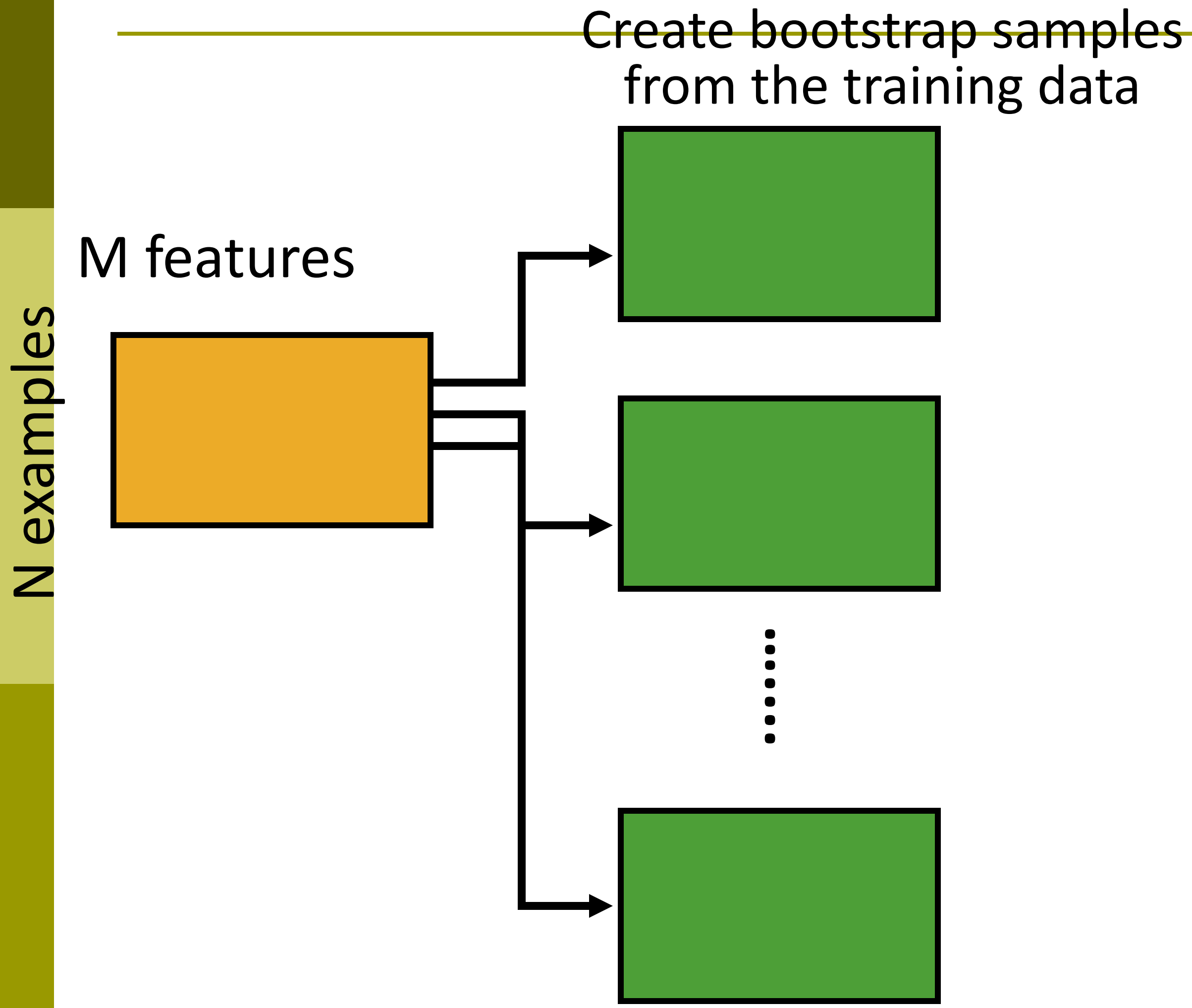
- Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
- If there are M input variables, a number m<M is specified such that at each node, m variables are selected at random out of the M. The best split on these m is used to split the node. The value of m is held constant while we grow the forest.
- Each tree is grown to the largest extent possible and there is no pruning.
- Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

Random Forest Classifier

Training Data

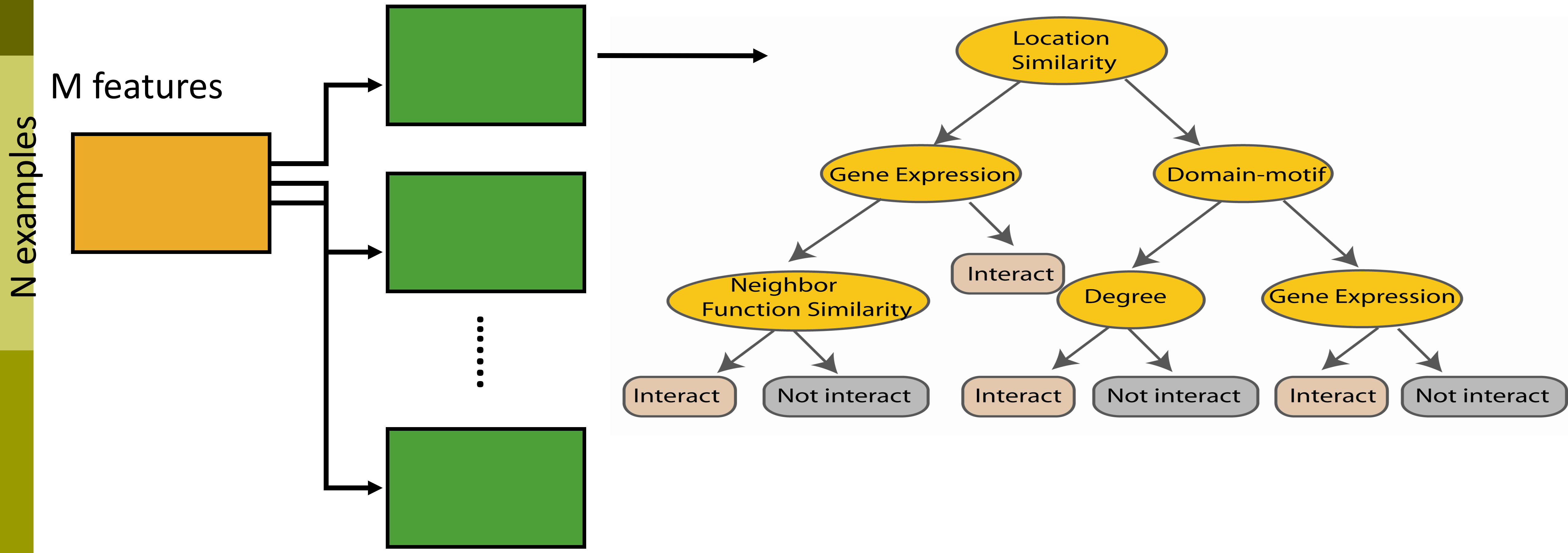


Random Forest Classifier



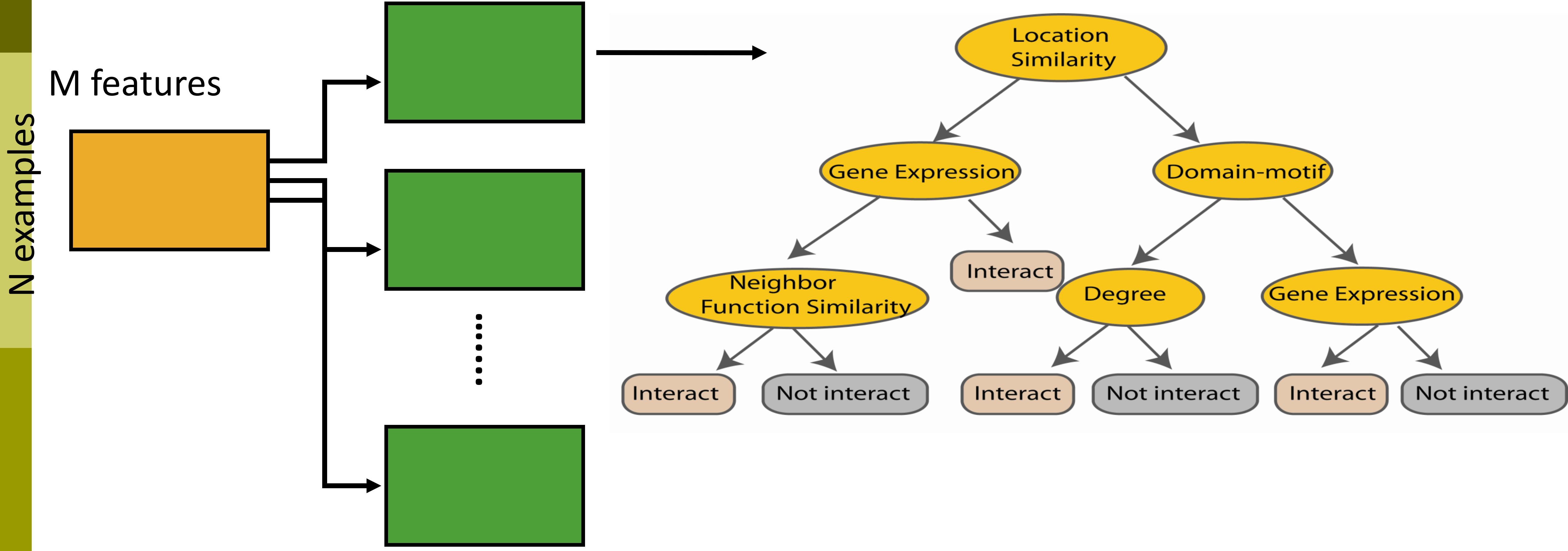
Random Forest Classifier

Construct a decision tree



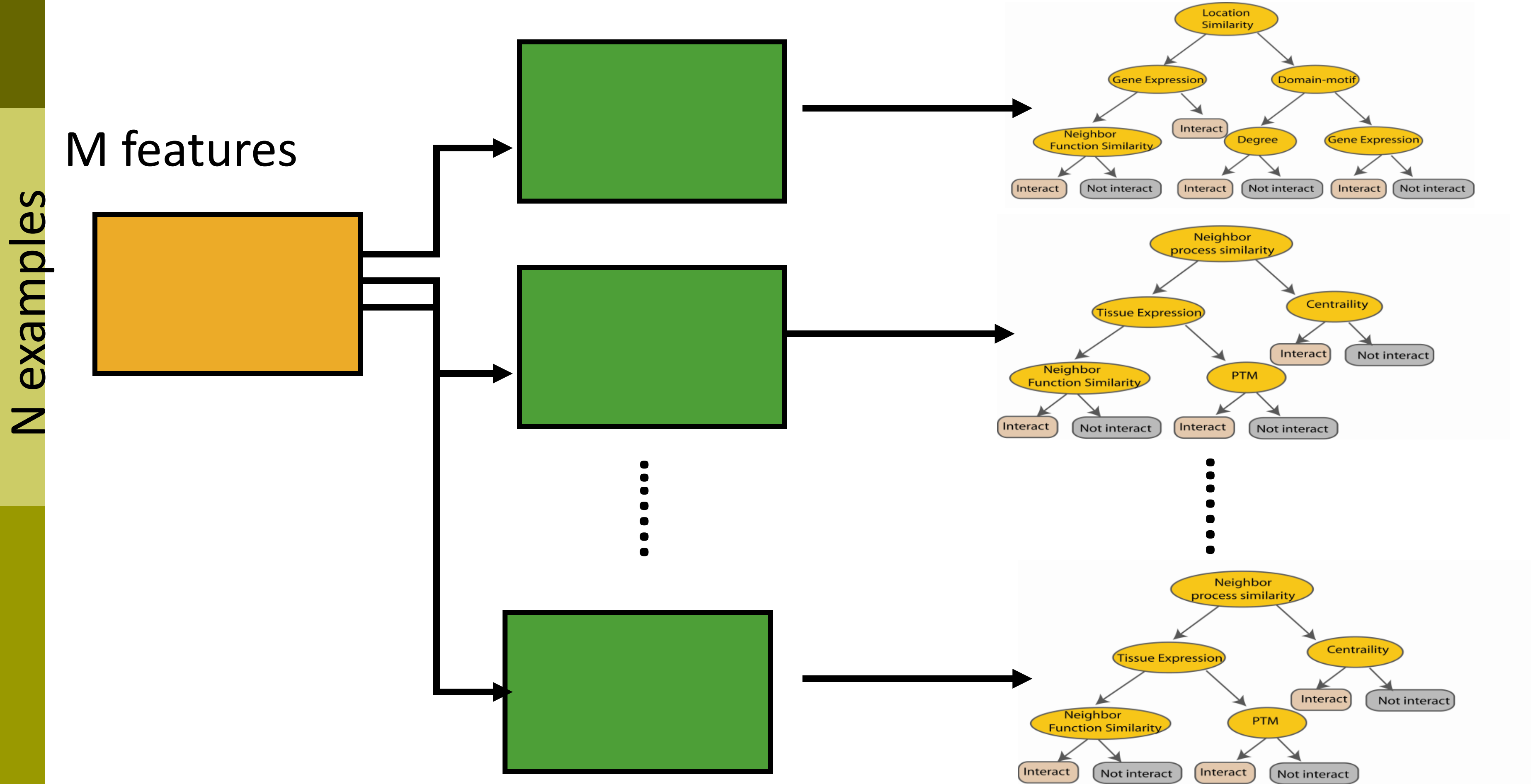
Random Forest Classifier

At each node in choosing the split feature
choose only among $m<M$ features

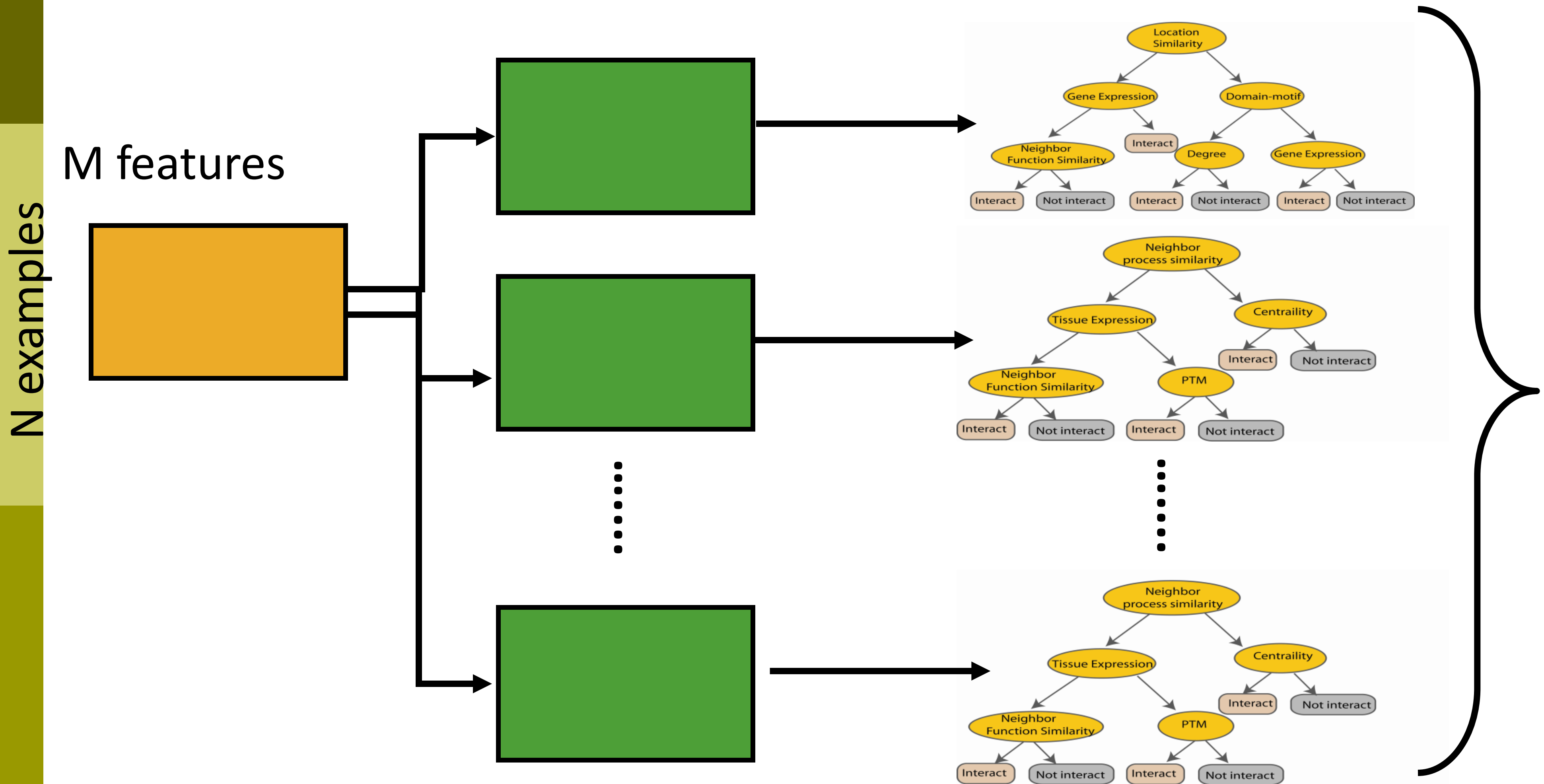


Random Forest Classifier

Create decision tree from each bootstrap sample



Random Forest Classifier



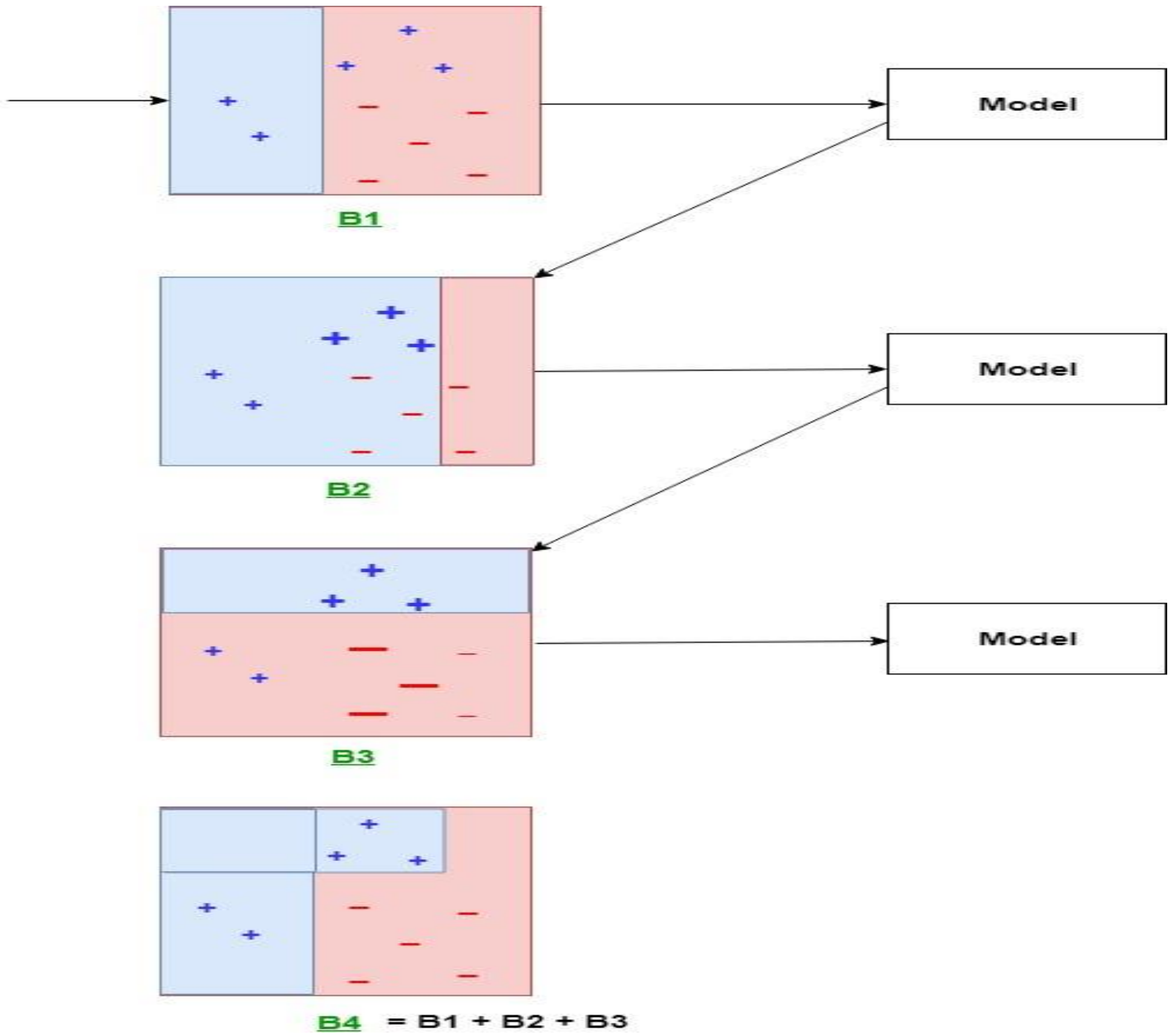
Take the majority vote

In simple…

- Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Boosting

- Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence.
- It is
- The two most common boosting ensemble machine learning algorithms are:
  - AdaBoost
  - Stochastic Gradient Boosting



Similarities and Differences – Bagging and Boosting

- Similarities
  - Both are ensemble methods
  - Both generate several training data sets by random sampling
  - Both make the final decision by averaging the N learners
  - Both are good at reducing variance and provide higher stability

Similarities and Differences – Bagging and Boosting

- Differences
  - Bagging- built independent models
  - Boosting- improves from the previous model
  - Bagging – average on all the models
  - Boosting – weighted average on the models
  - Bagging- solves overfitting
  - Boosting – reduces bias

Tuning Parameters

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

Hypertuning Parameters for GridSearchCV

- from sklearn.model_selection import GridSearchCV
- # Create the parameter grid based on the results of random search
```
param_grid = {
'bootstrap': [True],
'max_depth': [80, 90, 100, 110],
'max_features': [2, 3],
'min_samples_leaf': [3, 4, 5],
'min_samples_split': [8, 10, 12],
'n_estimators': [100, 200, 300, 1000]
}# Create a based model
rf = RandomForestRegressor()# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
cv = 3, n_jobs = -1, verbose = 2)
```

Reading Resources - Decision Trees

- https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/
- https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/#
- Decision Trees and Random forest – towardsdatascience.com