Supervised Learning
Classification

Decision Tree and Random Forest

Agenda

- Understanding Terminologies
  - Information Theory
  - Entropy
  - Conditional Entropy
  - Information Gain
- Decision Tree Algorithms
  - ID3 (Iterative Dichotomiser)
  - C4.5
  - C5.0

Agenda

- Decision Trees for Classification
  - Business Problem
  - Measures of Purity of node
    - Entropy
    - Gini Index
    - Classification error
- Construction of Decision Tree
- Model Performance Measures
  - Confusion Matrix
  - Cross Entropy
  - ROC - AUC Score

Agenda

- Overfitting in Decision Tree
- Ensemble Learning
    - Random Forest Classifier
- Feature Importance
    - Gini importance
    - Mean decrease in accuracy

Understanding Terminologies

Information theory

Information theory is based on the intuition that

Event	Information Gain	Example
Most likely event	No information	The sun rose this morning
Likely event	Little information	The sun rose at 6:30 a.m. this morning
Unlikely event	Maximum information	There was a solar eclipse this morning

Reference: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

Information theory

- Let I(x) denote the information of an event X
- It is the self information of an event X at x

$$I(x) = -\ln P(x)$$

- Since we have considered natural log its units is nat
- For log with base 2, we use units called bits or shannons

Shannon's entropy

- Entropy is the measure of information for classification problem, i.e. it measures the heterogeneity of a feature

- The entropy of a feature is calculated as

$$E = -\sum_{i=1}^{c} p_c \log_2 p_c$$

where p_c is the probability of occurrence of the class

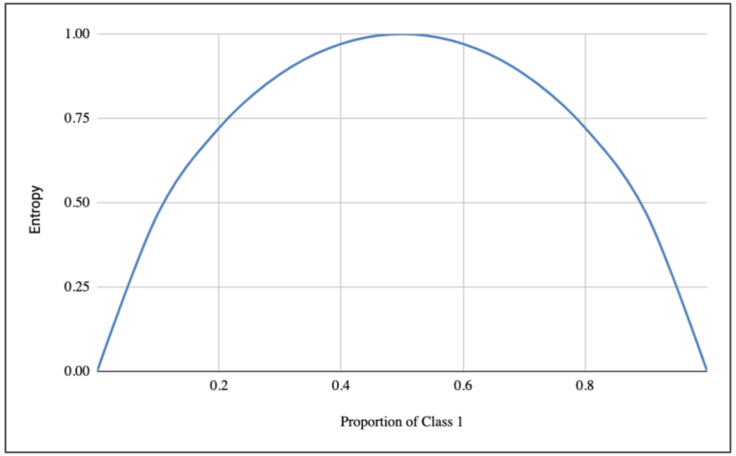
- A lower entropy is always prefered

- Entropy is always non-negative

Shannon's entropy

Consider a feature with two class the entropy for various proportion of classes is given below

Class 1	0	0.1	0.3	0.5	0.7	0.9	1
Class 2	1	0.9	0.7	0.5	0.3	0.1	0
Entropy	0	0.46	0.88	1	0.88	0.46	0



Shannon's entropy

Obtain the entropy of the given data

Obesity		Total
Not-Obese	Obese	
20	15	35

Entropy(Obesity) = - P(Not-obese) $\log_2$ (P(Not-obese)) -P(Obese) $\log_2$ (P(Obese))

Entropy(Obesity) = - 20/35 $\log_2$ (20/35) - 15/35 $\log_2$ (15/35)

Entropy(Obesity) = 0.985

Conditional entropy

The conditional entropy of one feature given other is calculated as from the contingency table of the two features.

$$E(T|X) = \sum_{x \in X} P(c)E(c)$$

It is the sum if the of the product of the probability of occurrence of the each class and the entropy of it.

Conditional entropy

To obtain the conditional entropy of Obesity given the person is a smoker

Entropy(Obesity|Smoker) = P(Not-obese) E(Not-obese|Smoker=Yes,No)

+ P(Obese) E(obese|Smoker=Yes,No)

Entropy(Obesity|Smoker) = (20/35) Entropy(15,5) + (15/35) Entropy(7,8)

Entropy(Obesity|Smoker) = (0.571) (0.811)+ (0.428) (0.997)

Entropy(Obesity|Smoker) = 0.890

		Obesity	
		Not - Obese	Obese
Smoker	Yes	15	7
	No	5	8
Total		20	15

Information gain

- Information gain is the decrease in entropy at a node

$$\text{Information Gain }(T, X) = \text{Entropy }(T) - \text{Entropy }(T|X)$$

- To construct the decision tree, the feature with highest information gain is chosen
- Information gain is always positive

Information gain

The information gain in the feature Obesity due to Smoker is

Information Gain (Obesity, Smoker) = Entropy(Obesity) - Entropy(Obesity|Smoker)

...from slides 9 and 11

= 0.985 - 0.890

= 0.095

Can Information Gain be negative?

After the split of data, the purity of data will be higher as a result the entropy will always be lower. Thus, the information gain is always positive.

Decision Tree Algorithms

Decision tree algorithms

- The decision tree algorithms are:
    - ID3 (Iterative Dichotomiser)
    - C4.5
    - C5.0
- Hunt's algorithm forms the basic to many of the decision tree algorithms like ID3, C4.5, CART and so on
- It grows a decision tree in a recursive manner by partitioning the samples into successively purer subsets

Hunt's algorithm

Let S_n be the training samples associated with node n and y_c be the class labels

The algorithm is as follows

1. If all samples belong to the same class y_c, then node n is a leaf node with label y_c
2. If S_n has samples with more than one class, an attribute value is selected to partition the samples into smaller subsets such that the samples in the subsets belong to the same class

Decision tree algorithms

ID3 Algorithm

- Invented by Ross Quinlan
- Handles only categorical data
- May not converge to optimal decision tree
- May overfit

C4.5 Algorithm

- Extension to ID3 algorithm
- Handles both categorical and numeric data
- Handles the missing data marked by '?'

C5.0 Algorithm

- Works faster than C4.5 algorithm
- More memory efficient than C4.5 algorithm
- Creates smaller trees with similar efficiency

Decision Trees for Classification

Business problem: loan approval

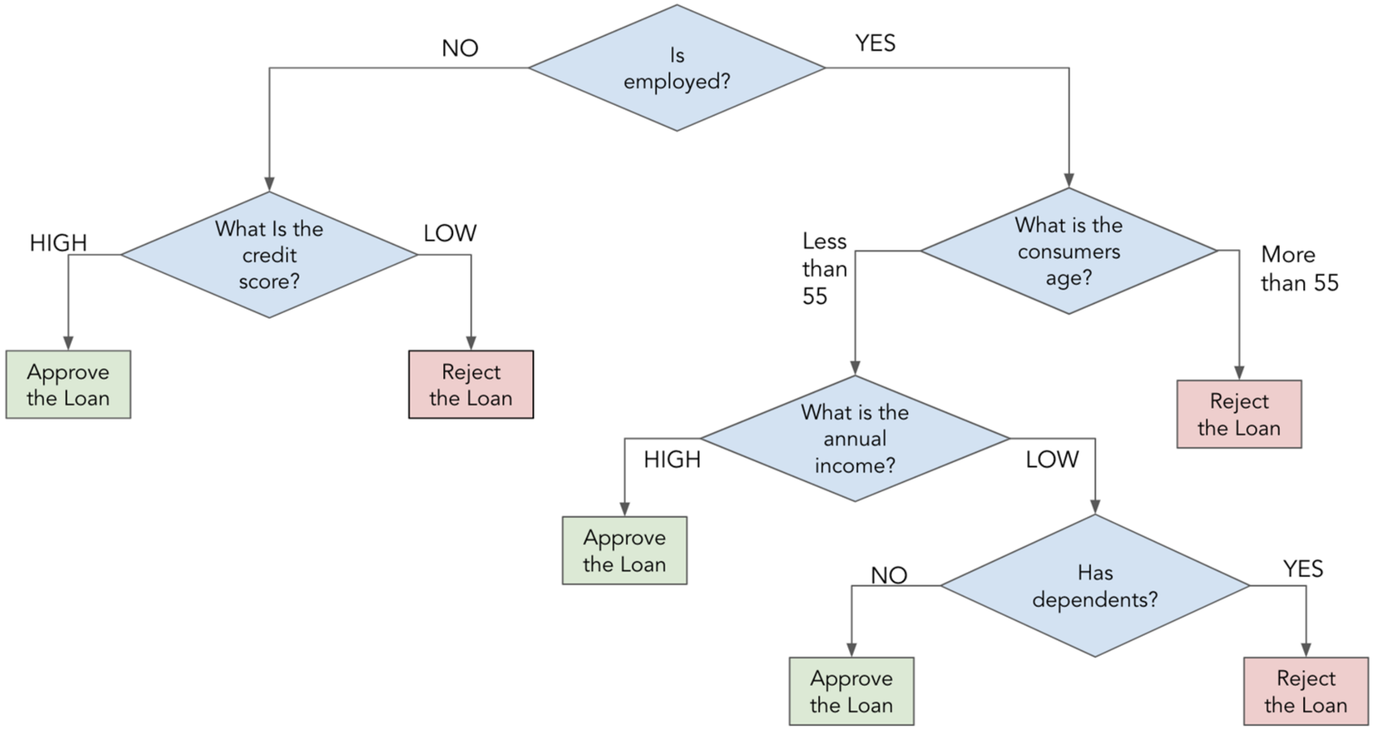
It is important to know the credibility of a consumer before lending a loan. It can be achieved by knowing answers to questions such as

- Is he employed?
- Is he nearing retirement?
- What his his annual income?
- Does he have any dependents?
- What is his age?
- What is his credit score?

These series of questions can be organised in a hierarchical structure.

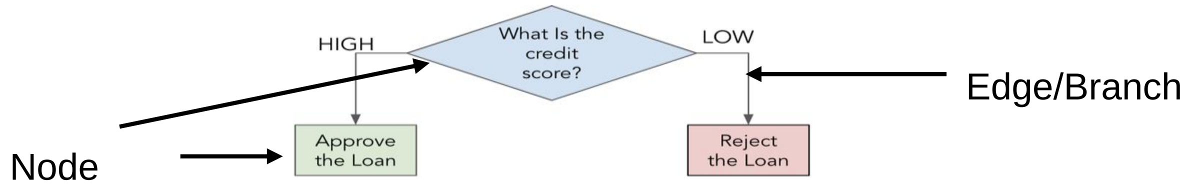
Business problem: loan approval

We create a flowchart like hierarchical structure to decide whether to approve the loan.



Decision Tree

- Decision tree is a classifier that results in flowchart-like structure with nodes and edges



- Each node denotes a condition on an attribute value
  (Condition: High or Low for the credit score)

- Each branch represents the outcome of the condition

- The outcome nodes are called the child nodes
  (Outcome: Approve or Reject the loan)

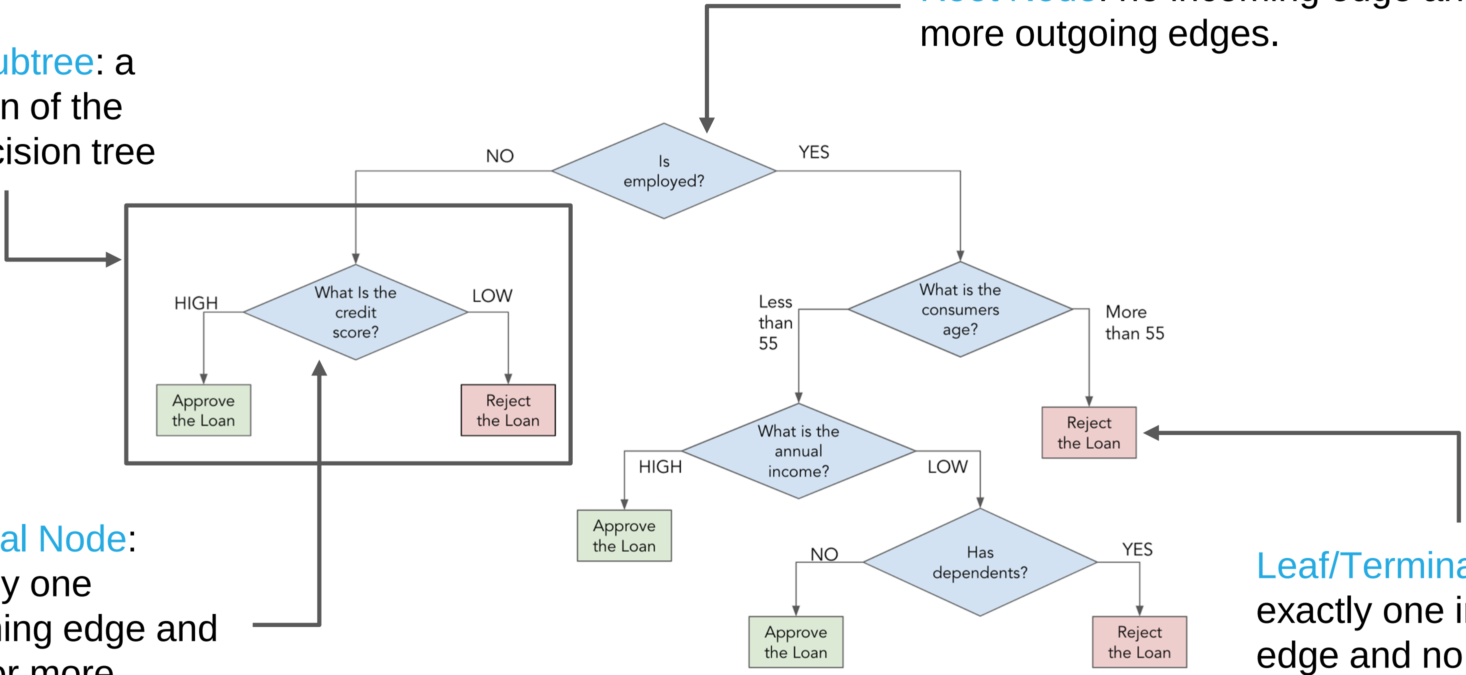
Terminologies

Root Node: no incoming edge and zero or more outgoing edges.

Branch/subtree: a subsection of the entire decision tree

Internal Node: exactly one incoming edge and zero or more outgoing edges.

Leaf/Terminal Node: exactly one incoming edge and no outgoing edge.



Measure of Purity of Node

Pure nodes

Consider the feature Credit Score, observe that for Credit Score = High, the loan proposal is approved

Credit Score = High	
Approved	Rejected
4	0

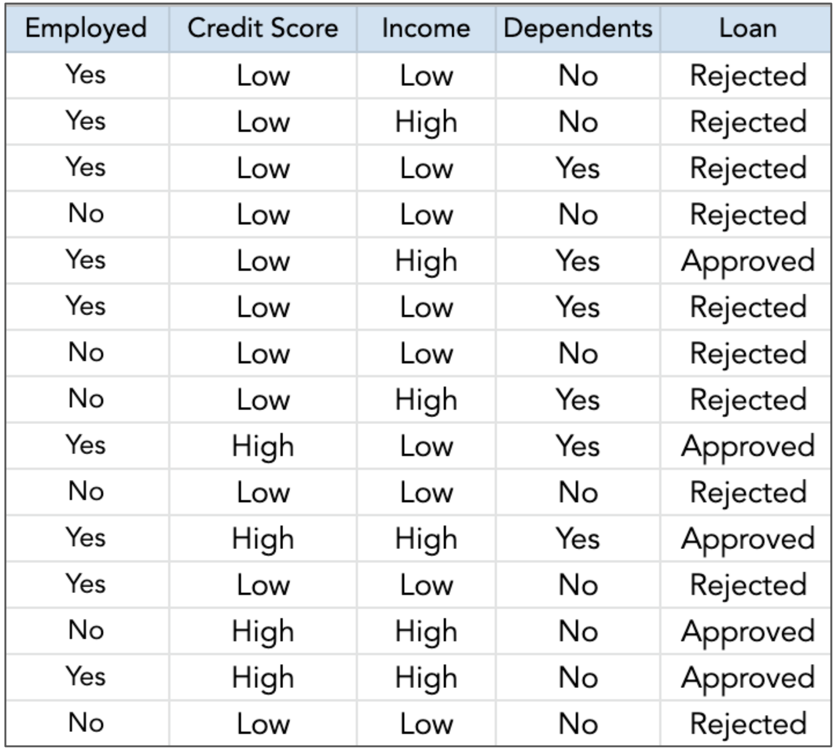
That to say the child nodes are pure or homogeneous.

(Homogeneous values in the target variable are all same)

Employed	Credit Score	Income	Dependents	Loan
Yes	Low	Low	No	Rejected
Yes	Low	High	No	Rejected
Yes	Low	Low	Yes	Rejected
No	Low	Low	No	Rejected
Yes	Low	High	Yes	Approved
Yes	Low	Low	Yes	Rejected
No	Low	Low	No	Rejected
No	Low	High	Yes	Rejected
Yes	High	Low	Yes	Approved
No	Low	Low	No	Rejected
Yes	High	High	Yes	Approved
Yes	Low	Low	No	Rejected
No	High	High	No	Approved
Yes	High	High	No	Approved
No	Low	Low	No	Rejected

Pure nodes

Is there any feature, other than Credit Score, which can be grouped to get pure nodes?



Employed	Credit Score	Income	Dependents	Loan
Yes	Low	Low	No	Rejected
Yes	Low	High	No	Rejected
Yes	Low	Low	Yes	Rejected
No	Low	Low	No	Rejected
Yes	Low	High	Yes	Approved
Yes	Low	Low	Yes	Rejected
No	Low	Low	No	Rejected
No	Low	High	Yes	Rejected
Yes	High	Low	Yes	Approved
No	Low	Low	No	Rejected
Yes	High	High	Yes	Approved
Yes	Low	Low	No	Rejected
No	High	High	No	Approved
Yes	High	High	No	Approved
No	Low	Low	No	Rejected

Measures of Purity of a node

- Entropy
- Gini Index
- Classification error

Measures of Purity of a node

- Entropy
- Gini Index
- Classification error

Entropy

- The entropy of a variable is calculated as

$$E = -\sum_{i=1}^{c} p_c \log_2 p_c$$ where p_c: probability of occurrence of the class

- The entropy of two variables is calculated as from the contingency table of the two variables

$$E(T,X) = \sum_{x \in X} P(c)E(c)$$

It is the sum if the of the product of the probability of occurrence of the class and its entropy.

Measures of Purity of a node

- Entropy
- Gini Index
- Classification error

Gini index

- The gini index of a variable is calculated as

$$Gini = 1 - \sum_{c=1}^{n} p_c^2$$

where p_c: probability of occurrence of the class

- For samples belonging to one class, the gini index is 0 and for equally distributed samples, the gini index is also 0

Gini index

Obtain the gini index of the given data

Obesity		Total
Not-Obese	Obese	
20	15	35

Entropy(Obesity) = 1 - [P(Not-obese)^2 +P(Obese)^2]

Entropy(Obesity) = 1- [(20/35)^2 + (15/35)^2]

Entropy(Obesity) = 0.306

Information gain using gini index

- It is similar to that of the information gain using entropy
- Information gain is the reduction in gini index

$$\text{Information Gain }(T, X) = \text{Gini index}(T) - \text{Gini index}(T|X)$$

Measures of Purity of a node

- Entropy
- Gini Index
- Classification error

Classification Error

- The classification error of a variable is calculated as

$$Error = 1 - \max p_c^2$$

where p_c: probability of occurrence of the class

- For samples belonging to one class, the classification error is 0 and for equally distributed samples, the classification error is 0.5

Summary

Measure	Properties
Entropy	● Used for C4.5 decision trees ● Computationally complex due to log in the equation
Gini Index	● Used for CART ● Calculated with less computation

Note: By default, in python 'DecisionTreeClassifier()' considers the 'Gini' measure.

Construction of Decision Tree

Construction of decision tree

- A decision tree is built from top to bottom. That is we begin with the root node
- While constructing a decision tree we try to achieve pure nodes
- A node is considered to be pure when all the data points belong to the same class
- This purity of nodes is determined using the entropy value

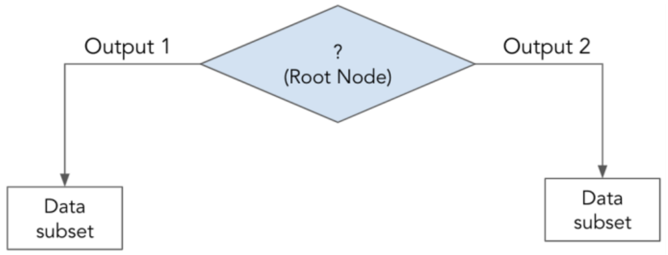
Construction of a decision tree

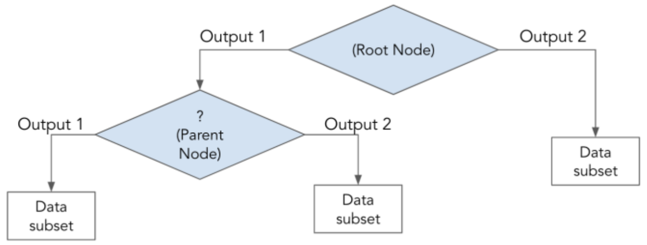
- Now we create the subset of the data
- Consider the remaining variables for the next iteration
- The child node which has Credit Score = High is pure, so the process terminates for that node
- It is the leaf node

Credit Score	Employed	Income	Dependents	Loan
Low	Yes	Low	No	Approved
	Yes	Low	No	Approved
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	No	Low	Yes	Rejected
	No	High	No	Approved
High	Yes	Low	Yes	Approved
	No	Low	No	Approved
	Yes	High	Yes	Approved
	Yes	Low	No	Approved
	No	High	No	Approved
	Yes	High	No	Approved
	No	Low	No	Approved

Construction of a decision tree - Procedure

- It is a recursive procedure
- To select the root node, from k features, select the feature with the highest information gain
- Split the data on this feature
- At the next node, from (k-1) features, select the feature with the highest information gain
- Split the data on this feature
- Continue the process till you exhaust all features



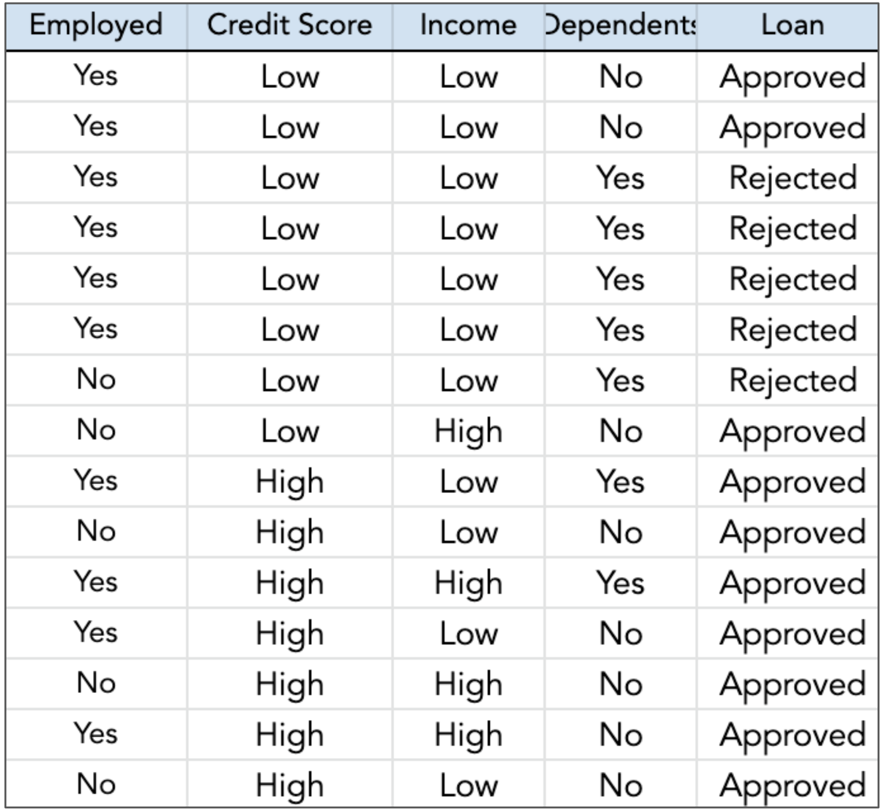


Construction of a decision tree

- Consider the adjacent data. We have 4 categorical features

- We shall first find the root node

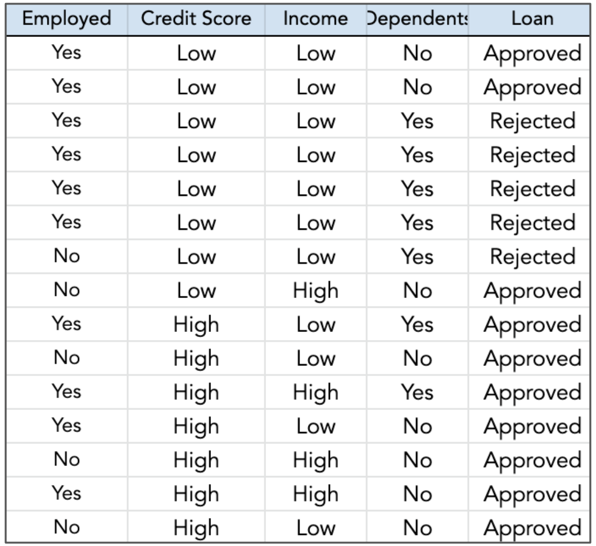
- To do so, calculate the information gain on each feature



Employed	Credit Score	Income	Dependents	Loan
Yes	Low	Low	No	Approved
Yes	Low	Low	No	Approved
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
No	Low	Low	Yes	Rejected
No	Low	High	No	Approved
Yes	High	Low	Yes	Approved
No	High	Low	No	Approved
Yes	High	High	Yes	Approved
Yes	High	Low	No	Approved
No	High	High	No	Approved
Yes	High	High	No	Approved
No	High	Low	No	Approved

Construction of a decision tree

Consider the categorical features and calculate the information gain.



Employed	Credit Score	Income	Dependent	Loan
Yes	Low	Low	No	Approved
Yes	Low	Low	No	Approved
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
Yes	Low	Low	Yes	Rejected
No	Low	Low	Yes	Rejected
No	Low	High	No	Approved
Yes	High	Low	Yes	Approved
No	High	Low	No	Approved
Yes	High	High	Yes	Approved
Yes	High	Low	No	Approved
No	High	High	No	Approved
Yes	High	High	No	Approved
No	High	Low	No	Approved

Variable	Employed	Credit Score	Income	Dependents
Information gain	0.030	0.331	0.185	0.282

Refer 7-12 for information gain calculation

Construction of a decision tree

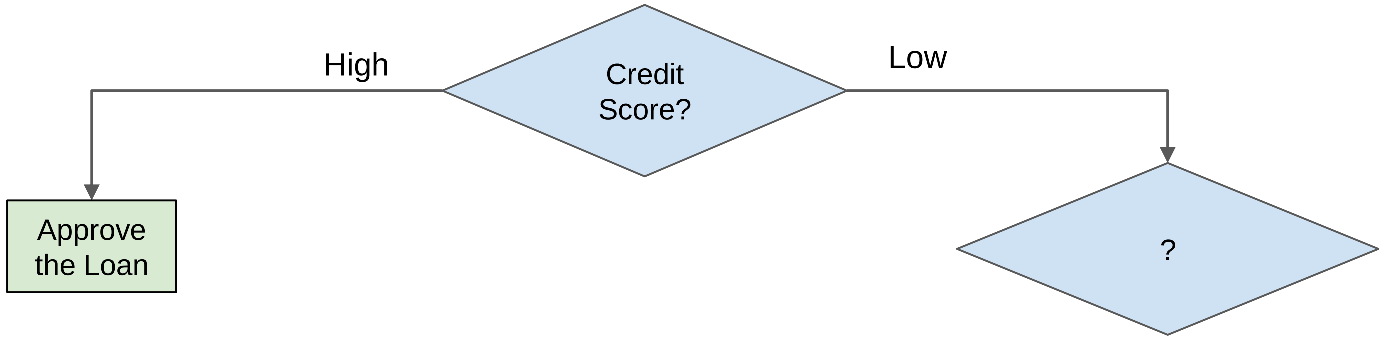
Thus, we have information gain values as

Variable	Employed	Credit Score	Income	Dependents
Information gain	0.030	0.331	0.185	0.282

Hence, we conclude Credit Score has the highest information gain.

Construction of a decision tree

The resultant tree at this stage is:

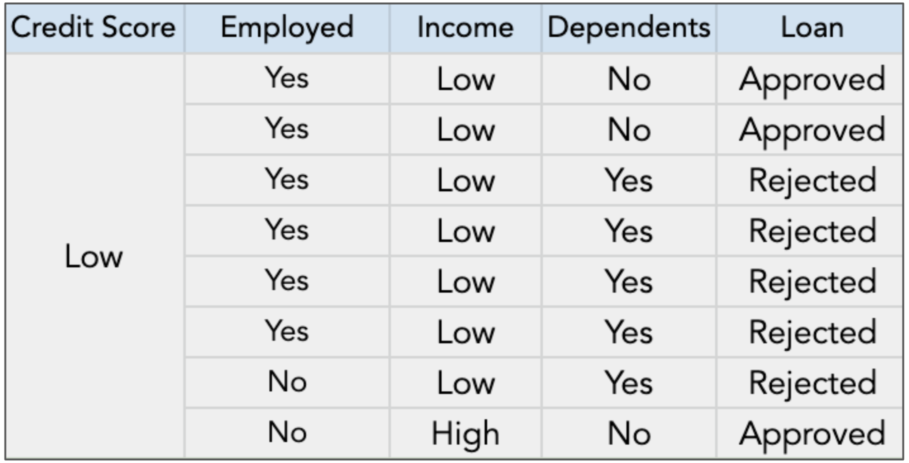


Construction of a decision tree

- The decision tree will now grow on the child node with Credit Score = Low

- Note that we now use only the subset of the data. So the entropy of the target variable needs to be computed again

- Compute the information gain for the remaining variables - Employed, Income and Dependents



Credit Score	Employed	Income	Dependents	Loan
Low	Yes	Low	No	Approved
	Yes	Low	No	Approved
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	No	Low	Yes	Rejected
	No	High	No	Approved

Construction of a decision tree

Thus, we have information gain values as

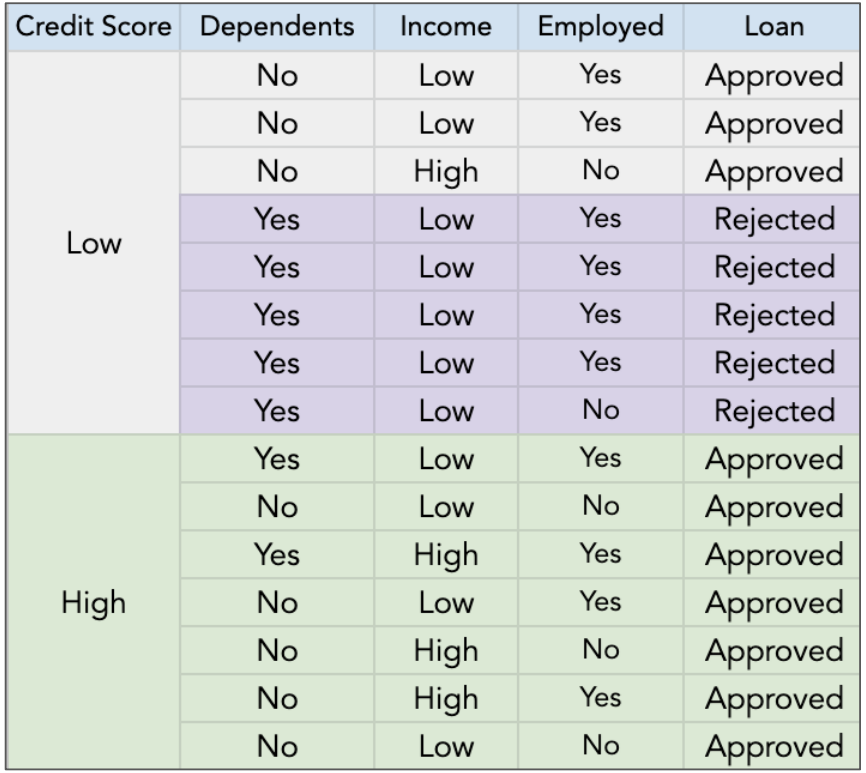
Variable	Employed	Income	Dependents
Information gain	0.159	0.610	0.954

Hence, we conclude Dependents has the highest information gain.

Construction of a decision tree

- Now we create the subset of the data

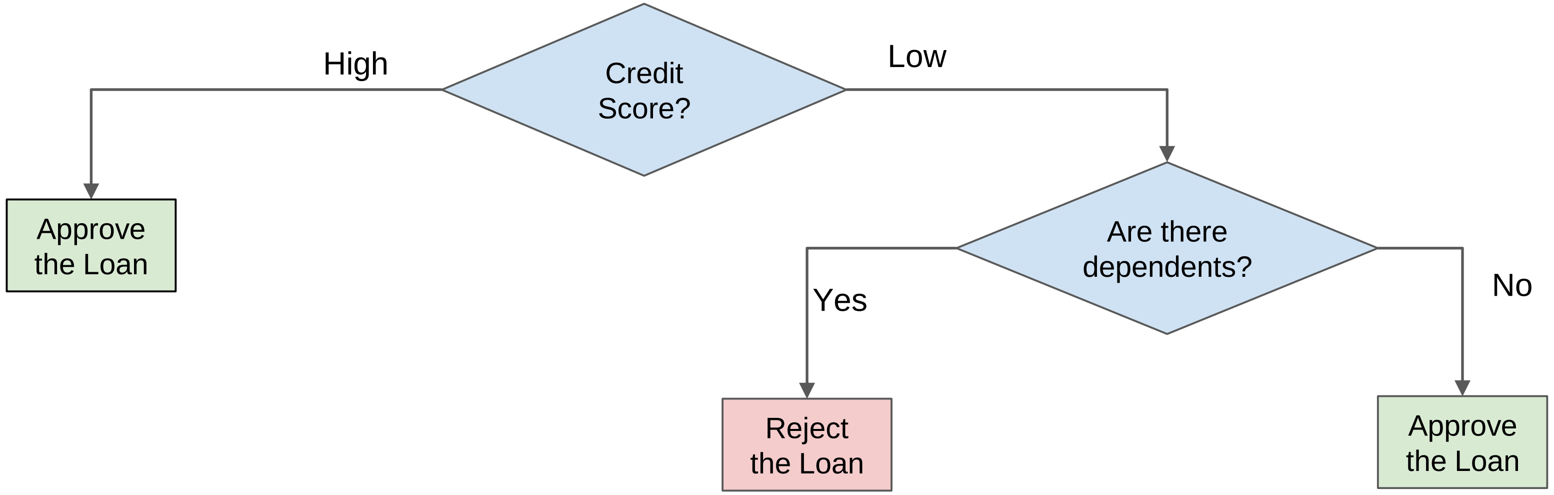
- Consider the remaining variables for the next iteration



Credit Score	Dependents	Income	Employed	Loan
Low	No	Low	Yes	Approved
	No	Low	Yes	Approved
	No	High	No	Approved
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	No	Rejected
High	Yes	Low	Yes	Approved
	No	Low	No	Approved
	Yes	High	Yes	Approved
	No	Low	Yes	Approved
	No	High	No	Approved
	No	High	Yes	Approved
	No	Low	No	Approved

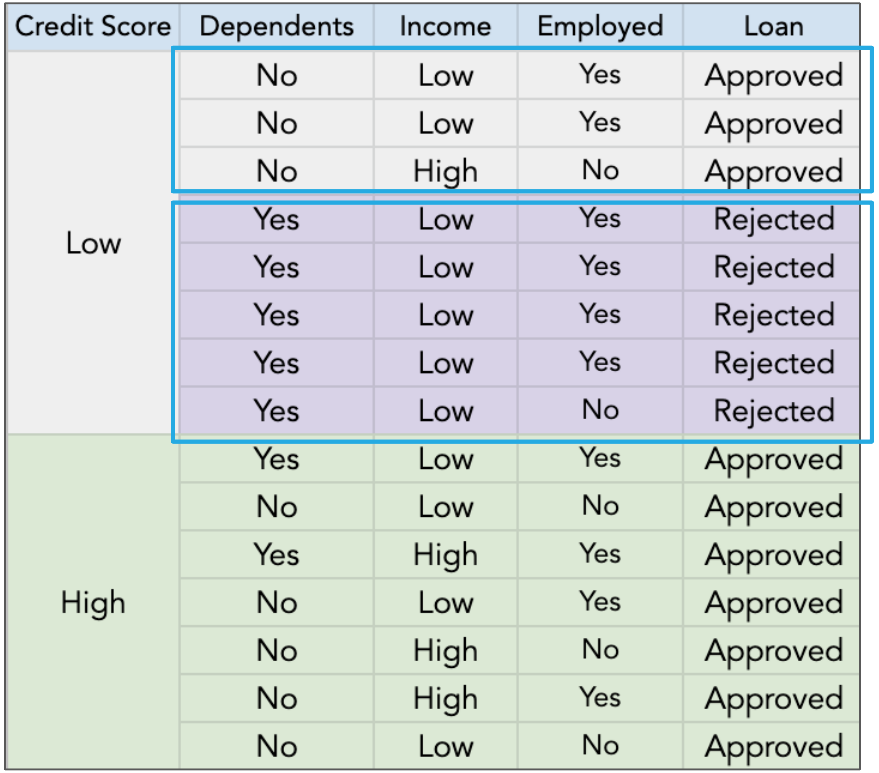
Construction of a decision tree

The resultant decision tree:



Construction of a decision tree

- The child nodes for Dependents are pure, so the process terminates here
- They are the leaf nodes



Credit Score	Dependents	Income	Employed	Loan
Low	No	Low	Yes	Approved
	No	Low	Yes	Approved
	No	High	No	Approved
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	Yes	Rejected
	Yes	Low	No	Rejected
High	Yes	Low	Yes	Approved
	No	Low	No	Approved
	Yes	High	Yes	Approved
	No	Low	Yes	Approved
	No	High	No	Approved
	No	High	Yes	Approved
	No	Low	No	Approved

Choosing between attributes with same information gain

- The first predictor found from left to right in a data set is considered
- Some decision tree algorithm implementations might consider each of the variables with same information gain at a time and check which model performs better
- This rule applies to all parent nodes

WANT TO KNOW MORE?

How is the entropy calculated for numeric features?

1. Sort the data in ascending order and compute the midpoints of the successive values. These midpoints act as the thresholds

2. For each of these threshold values, enumerate through all possible values to compute the information gain

3. Select the value which has the highest information gain

(Demonstration in the next slide)

WANT
TO KNOW
MORE?

Entropy for numeric feature

Data

Age	Loan
45	Rejected
54	Approved
56	Rejected
58	Rejected
21	Approved
31	Rejected
45	Approved

Sorted Data

Age	Loan
21	Approved
31	Rejected
45	Approved
45	Rejected
54	Approved
56	Rejected
58	Rejected

Midpoints

Age	Midpoints
21	-
31	26
45	38
54	49.5
56	55
58	57

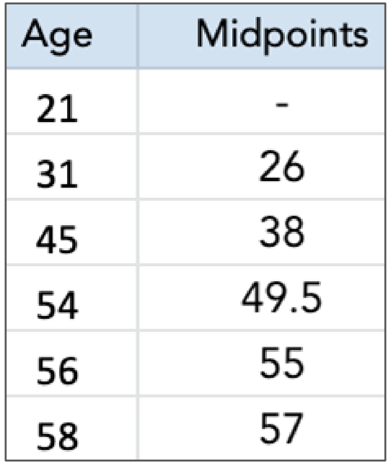
For repeated values, we consider it only once (in this case 45).

Continued to next page...

WANT
TO KNOW
MORE?

Information gain for numeric feature

Midpoints

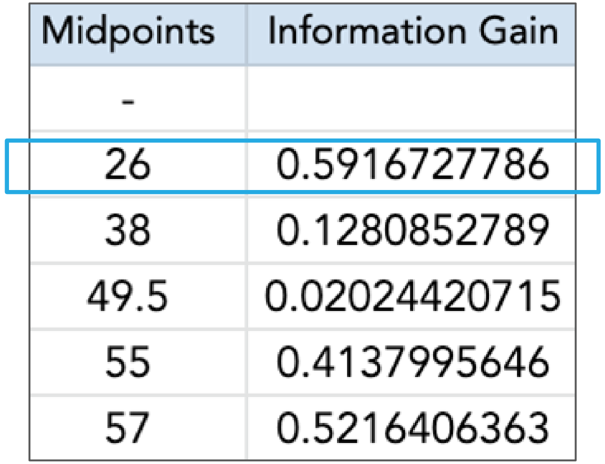


Age	Midpoints
21	-
31	26
45	38
54	49.5
56	55
58	57

For midpoint, m, the data is divided into two parts - data less than m and data more than m

These two part form the two branches in the tree

Information gain for all midpoints



Midpoints	Information Gain
-	
26	0.5916727786
38	0.1280852789
49.5	0.02024420715
55	0.4137995646
57	0.5216406363

We consider the threshold with maximum information gain. In this case, we consider Age < 26

Model Evaluation

Model Evaluation

- Training Error:
  - Number of misclassification on the training set
  - Also known as resubstitution or apparent error
- Generalization error:
  - Number of misclassification on the test set

Model Performance Measures

Performance metrics

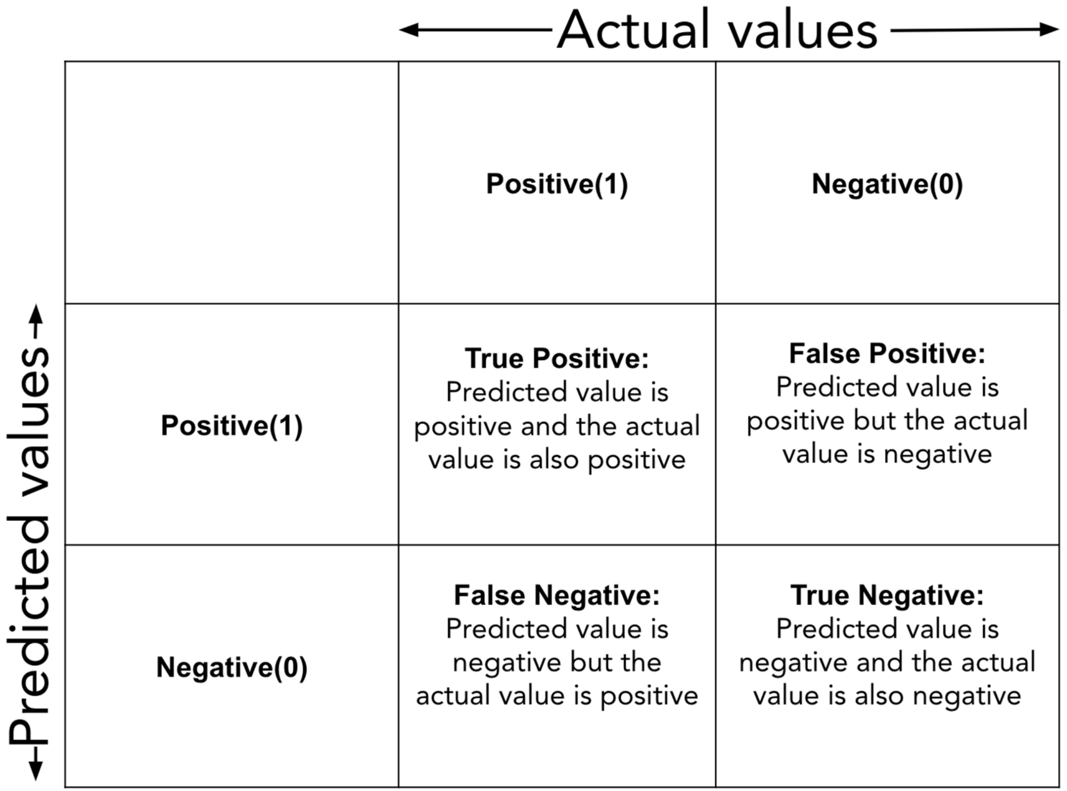
The following metrics can be used to evaluate the performance of classification models:

- Confusion matrix
- Cross entropy
- Receiver Operating Characteristic (ROC)

Confusion matrix

- Performance measure for classification problem

- It is a table used to compare predicted and actual values of the target variable



		Actual values	
		Positive(1)	**Negative(0)**
Predicted values	**Positive(1)**	**True Positive:** Predicted value is positive and the actual value is also positive	**False Positive:** Predicted value is positive but the actual value is negative
	Negative(0)	**False Negative:** Predicted value is negative but the actual value is positive	**True Negative:** Predicted value is negative and the actual value is also negative

Cross Entropy

- Cross entropy is the loss function commonly used in classification problems

- As the prediction goes closer to actual value the cross entropy decreases

$$H(y) = -\sum_i y_{act(i)} \ln\left(y_{pred(i)}\right)$$

i = class (0 or 1)

H(y) = cross entropy

$y_{act(i)}$ = actual probability for class i

$y_{pred(i)}$ = predicted probability for class i

ROC

- The True Positive Rate and False Positive Rate values change with different threshold values

- ROC curve is the plot of TPR against the FPR values obtained at all possible threshold values

Decision tree as variable selection method

- Recall all the stepwise selection methods learnt in linear regression module
- Variables are chosen to be in the model based on their significance
- Likewise, in decision tree we use one of the impurity measures to select the variable that should be included first in the decision tree

Overfitting in a Decision Tree

Overfitting in a decision tree

- Decision trees are prone to overfitting
- Overfitting occurs when the decision tree uses all of the data samples in the decision tree, resulting in a perfect fit
- An overfitted tree
    - may have leaf nodes that contain only one sample, ie. singleton node
    - are generally complicated and long decision chains
- An overfitted tree has low training error and a high generalization error, hence can not be generalised for new data

Handle overfitting

- An approach to handle overfitting is pruning
- Pruning is a technique that removes the branches of a tree that provide little power to classify instances, thus reduces the size of the tree
- Pruning reduces the complexity of the tree
- Pruning can be achieved in two ways:
  - Pre-Pruning: The decision tree stops growing before the tree completely grown
  - Post-Pruning: The decision tree is allowed to grow completely and then prune

Hyperparameters

Pre-pruning can be done by specifying the following hyperparameters:

- max_depth:
  - It is the maximum length of the decision allowed to grow
  - Once the max_depth value is reached the tree will not grow further

- min_samples_split:
  - The minimum samples required to split an internal node

Hyperparameters

- max_leaf_nodes:
  - The maximum number of leaf nodes the decision tree can have in best-fit manner
  - The best nodes are defined as relative reduction in impurity
  - By default the value is None, implies that no restriction to number of leaf nodes

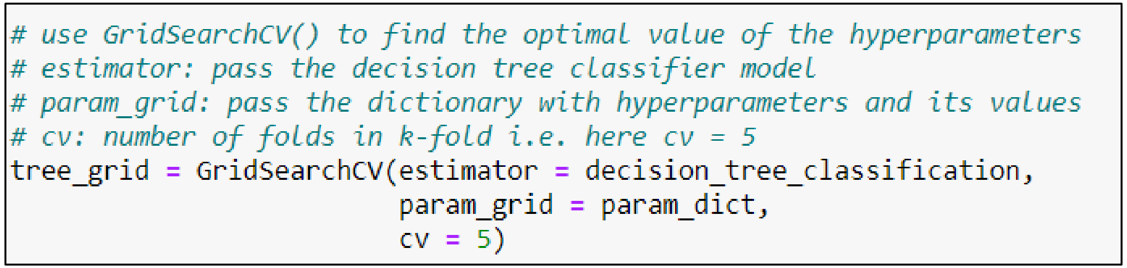
- max_feature_size:
  - The maximum number of features to be considered to while splitting a node

Hyperparameters

- min_samples_leaf:
    - The minimum samples required to be at the leaf node
    - A node will split further only if its child nodes will have the min_sample_leaf
    - May give the effect of smoothing

Hyperparameter tuning

- The hyperparameters can be tuned using GridSearch method
- It considers all the combinations of the hyperparameters and returns the optimal hyperparameter values



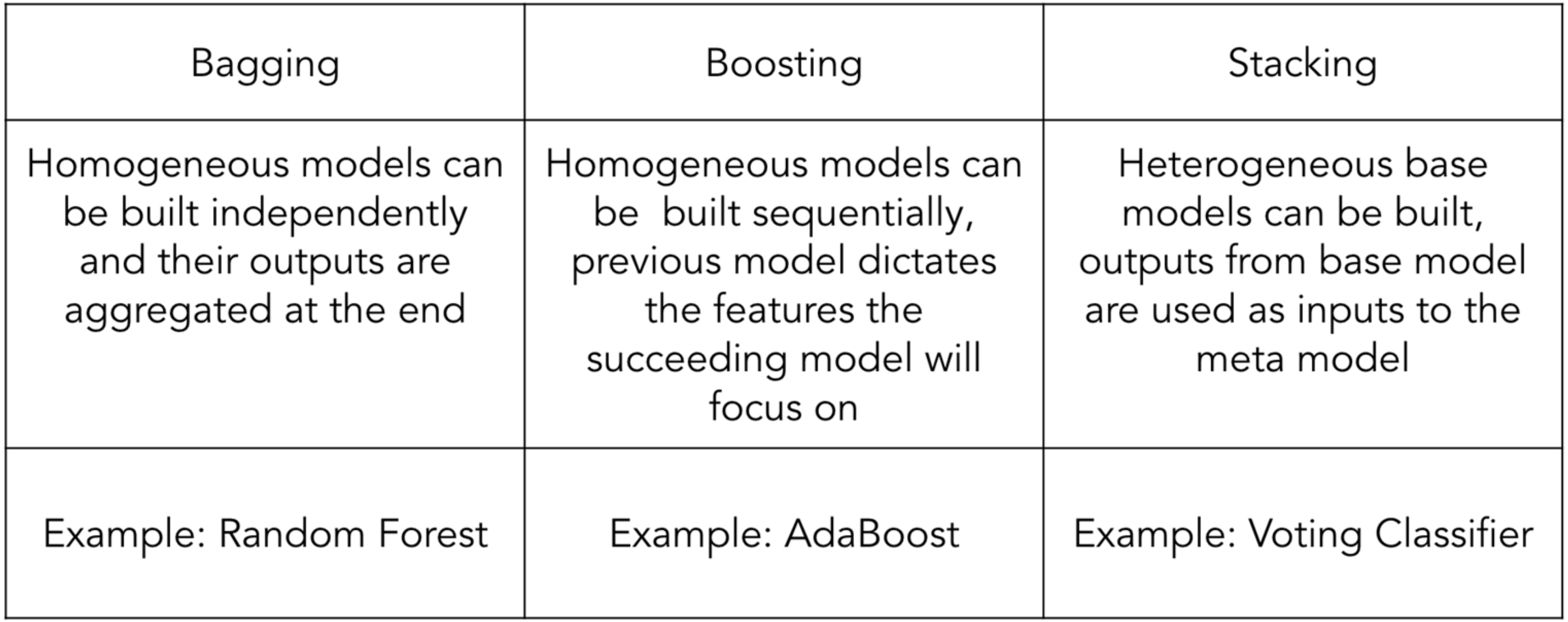
```
# use GridSearchCV() to find the optimal value of the hyperparameters
# estimator: pass the decision tree classifier model
# param_grid: pass the dictionary with hyperparameters and its values
# cv: number of folds in k-fold i.e. here cv = 5
tree_grid = GridSearchCV(estimator = decision_tree_classification,
                         param_grid = param_dict,
                         cv = 5)
```

Ensemble Learning

Ensemble learning

- Ensemble learning algorithms combine multiple models into one predictive model
- Decisions from several weak learners are combined to increase the model performance

Ensemble learning methods



Bagging	Boosting	Stacking
Homogeneous models can be built independently and their outputs are aggregated at the end	Homogeneous models can be built sequentially, previous model dictates the features the succeeding model will focus on	Heterogeneous base models can be built, outputs from base model are used as inputs to the meta model
Example: Random Forest	Example: AdaBoost	Example: Voting Classifier

Random Forest Classifier

$$\underbrace{\text{Random}}_{\substack{\text{The subsets of} \\ \text{variables are} \\ \text{selected at} \\ \text{random to build} \\ \text{a decision tree}}}\ \underbrace{\text{Forest}}_{\substack{\text{Forest of} \\ \text{“Decision” trees}}}\ \text{Classifier}$$

Random Forest

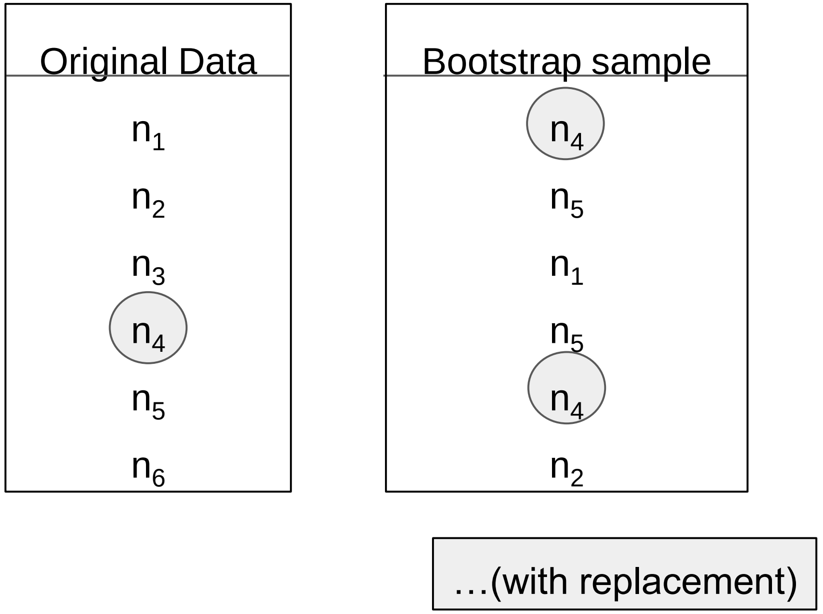
- Random Forest consists of several independent decision trees that operate as an ensemble

- It is an ensemble learning algorithm based on bagging

- Train decision tree models on bootstrap samples where variables are selected at random. The aggregate output from these tree is considered as the final output

Bootstrap sample

- Random sampling with replacement
- For a data with i observations, a random sample with replacement of size i, is a bootstrap sample
- The observations n_3 and n_6 are not included in the bootstrap sample, they are the out-of-bag (OOB) samples



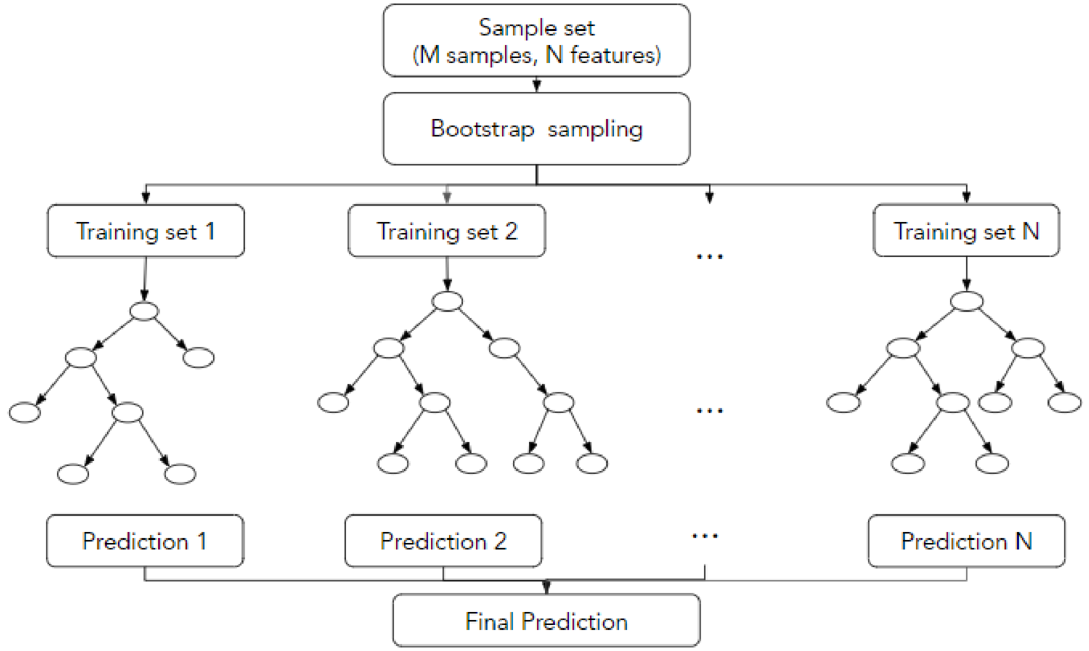
did you know?

Out-of-bag sample

Almost 36.8% of the training data has the potential to be the out-of-bag sample.

How?
Since, we consider the bootstrap sample, i.e. a random sample with replacement. The probability of not selecting a sample is $(1 - 1/N)^N$. For large N, the probability $(1 - 1/N)^N$ is approximately $1/e = 0.368$.

Steps for prediction using random forest:



For classification, the final prediction considers the mode of the predicted labels and for regression, it considers the average of the predicted values.

Random forest hyperparameters

Hyperparameter	Description
n_estimators	number of decision trees built for the random forest
max_depth	longest path between root node and leaf node
min_samples_split	minimum number of observations at which the node stop splitting even if it is not pure
max_leaf_nodes	specifies the maximum number of terminal nodes a decision tree in the random forest can have
min_samples_leaf	if a node has observations equal to min_samples_leaf it cannot split further
max_samples	determines the number of bootstrapped samples
max_features	maximum number of features considered at a node for splitting

Feature importance in random forest

- A technique that assigns a score to independent features based on its importance in predicting the target variable

- These scores indicates the relative importance of the features

- Two techniques that are used to find the feature importance in random forest:
  - Gini importance
  - Mean decrease in accuracy

Gini importance

- Also known as mean decrease impurity
- It is the average total decrease in the node impurity weighted by the probability of reaching it
- The average is taken over all the trees in the random forest
- The sklearn library in python considers this measure
- ‘rf_model.feature_importances_’ returns the feature importance for each variable

Mean decrease in accuracy

- Measure the decrease in the accuracy on the out-of-bag data
- The purpose is to measure the decrease in the accuracy on OOB data when you randomly permute the values for that feature
- If the decrease is low, then the feature is not important, and vice-versa

Thank You