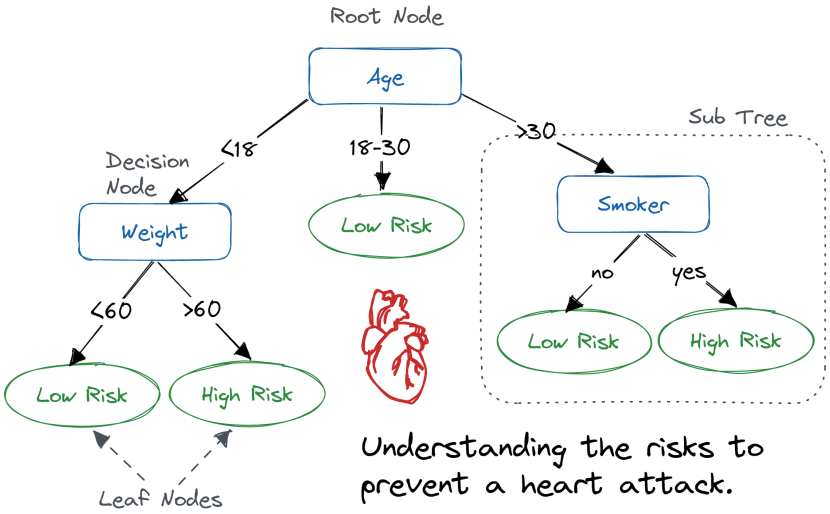
Classification and Regression Trees (CART)

What is a decision tree?

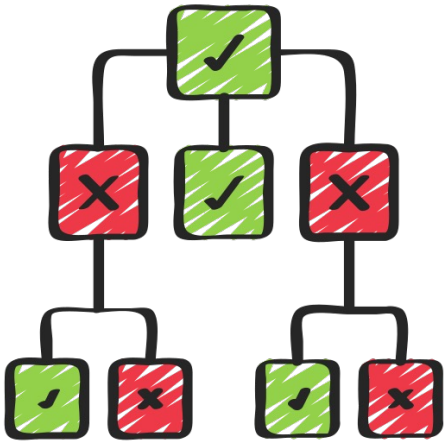
Breiman et al. (1984)



Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

In the decision tree, nodes are **split** into sub-nodes **based on a threshold** value of an attribute. The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.

Key Components of Decision Trees



Root Node: The decision tree's starting node, which stands for the complete dataset.

Branch Nodes: Internal nodes that represent decision points, where the data is split based on a specific attribute.

Leaf Nodes: Final categorization or prediction-representing terminal nodes.

Decision Rules: Rules that govern the splitting of data at each branch node.

Attribute Selection: The process of choosing the most informative attribute for each split.

Splitting Criteria: Metrics like information gain, entropy, or the Gini Index are used to calculate the optimal split.

Procedure

Step 1 Find the best attribute and place it on the root node of the tree.
Step 2 Split the training set of the dataset into subsets.
While making the subset make sure that each subset of training dataset should have the same value for an attribute.
Step 3 Find leaf nodes in all branches by repeating 1 and 2 on each subset.

Measures of Impurity

The **impurity function** measures the extent of purity for a region containing data points from possibly different classes.

Gini Index

- $I = 1 - \sum_{k=1}^{m} p_k^2$

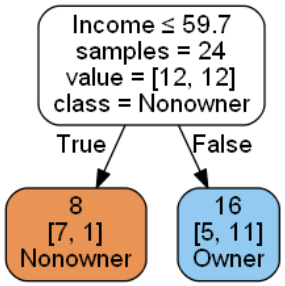
Entropy

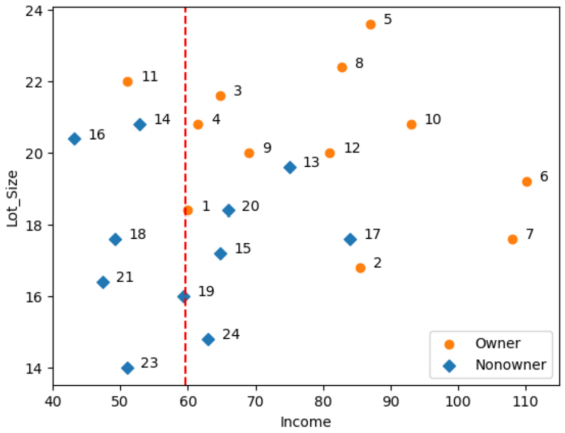
- $H = -\sum_{k=1}^{m} p_k log_2(p_k)$

The impurity function is zero when all observations belong to the same class

p_k is the proportion of observations that belong to class k

Riding Mowers





$$I_{left} = 1 - \left(\frac{7}{8}\right)^2 - \left(\frac{1}{8}\right)^2 = 0.219$$

$$H_{left} = -\frac{7}{8} log_2 \frac{7}{8} - \frac{1}{8} log_2 \frac{1}{8} = 0.544$$

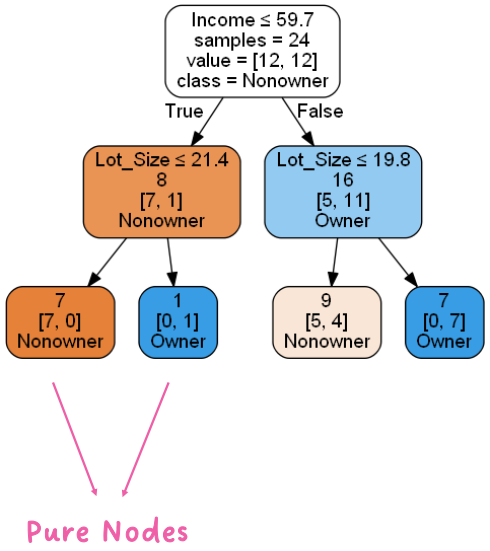
$$I_{right} = 1 - \left(\frac{11}{16}\right)^2 - \left(\frac{5}{16}\right)^2 = 0.430$$

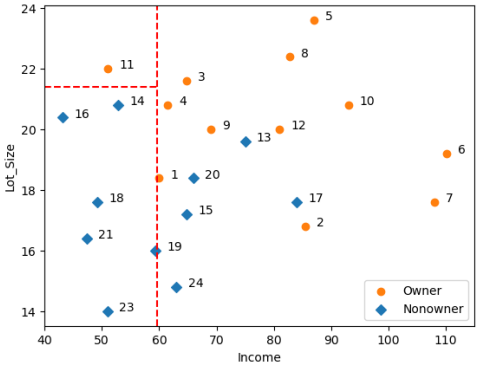
$$H_{right} = -\frac{11}{16} log_2 \frac{11}{16} - \frac{5}{16} log_2 \frac{5}{16} = 0.896$$

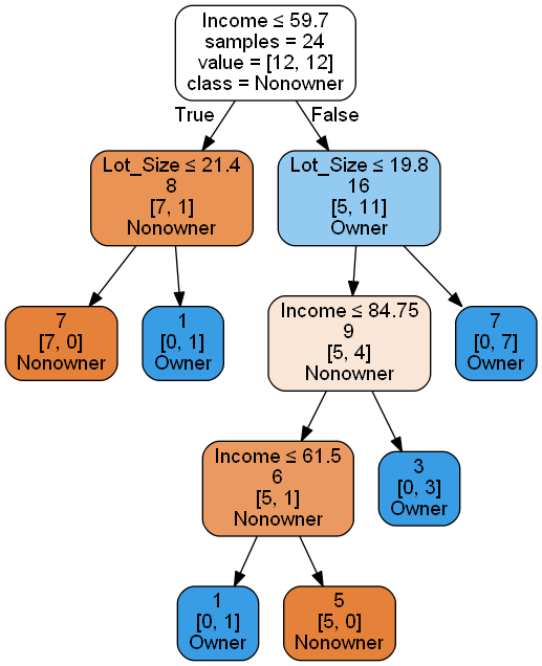
$$gini = \frac{8}{24} \times 0.219 + \frac{16}{24} \times 0.430 = 0.359$$

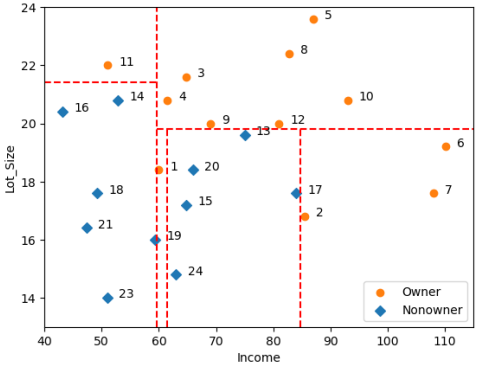
$$entropy = \frac{8}{24} \times 0.544 + \frac{16}{24} \times 0.896 = 0.779$$

Riding Mowers







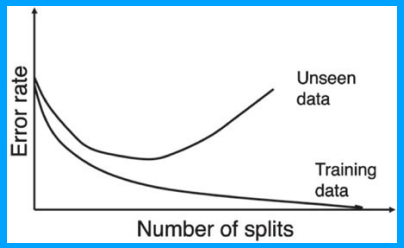


Evaluating the Performance of a Classification Tree

- Tree structure can be quite **unstable**, shifting substantially depending on the sample chosen.
- A fully-fit tree will invariably lead to **overfitting**.

Use cross-validation

Stop tree growth

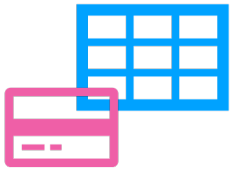


Acceptance of Personal Loan

Universal Bank is a relatively young bank that is growing rapidly in terms of overall customer acquisition. The majority of these customers are liability customers with varying sizes of relationship with the bank. The customer base of asset customers is quite small, and the bank is interested in growing this base rapidly to bring in more loan business. In particular, it wants to explore ways of converting its liability (deposit) customers to personal loan customers.

A campaign the bank ran for liability customers showed a healthy conversion rate of over 9% successes. This has encouraged the retail marketing department to devise smarter campaigns with better target marketing. **The goal of our analysis is to model the previous campaign's customer behavior to analyze what combination of factors make a customer more likely to accept a personal loan.** This will serve as the basis for the design of a new campaign.

Acceptance of Personal Loan



```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
import matplotlib.pylab as plt
from dmba import plotDecisionTree, classificationSummary, regressionSummary
```

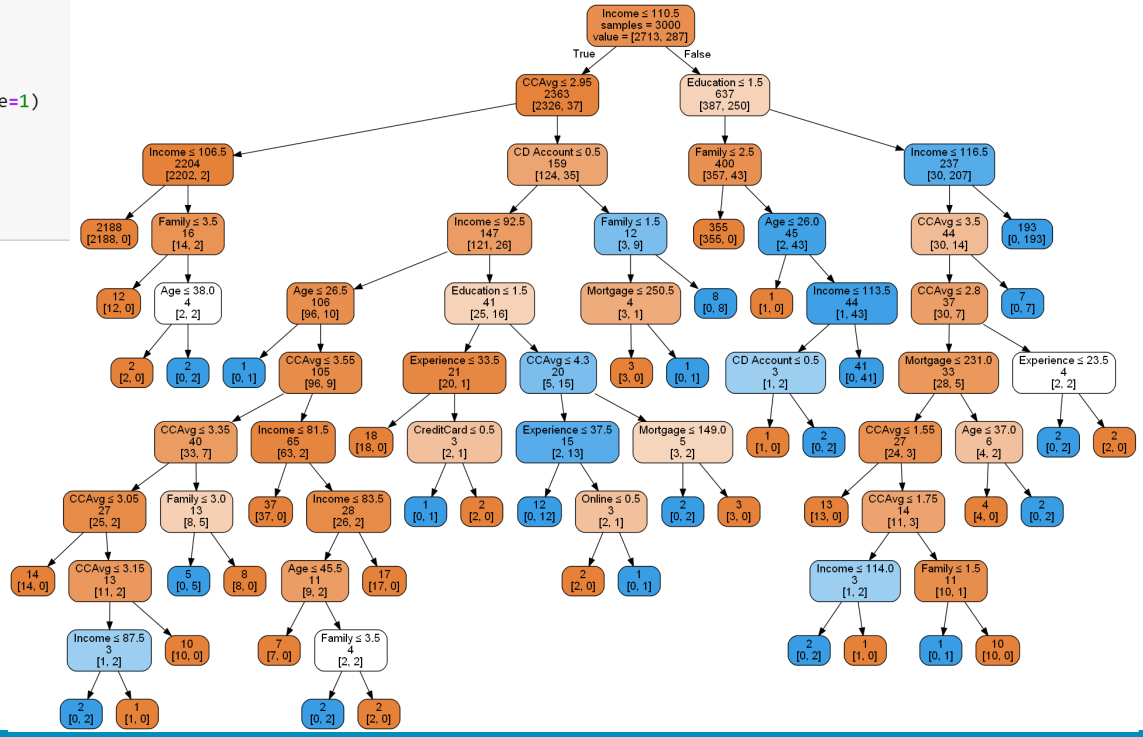
```
bank_df = pd.read_csv('UniversalBank.csv')
bank_df.head()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

Acceptance of Personal Loan

```
bank_df.drop(columns=['ID', 'ZIP Code'], inplace=True)

X = bank_df.drop(columns=['Personal Loan'])
y = bank_df['Personal Loan']
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

fullClassTree = DecisionTreeClassifier(random_state=1)
fullClassTree.fit(train_X, train_y)

plotDecisionTree(fullClassTree, feature_names=train_X.columns)
```

```
Confusion Matrix (Accuracy 1.0000)

       Prediction
Actual    0    1
     0 2713    0
     1    0  287
Confusion Matrix (Accuracy 0.9790)

       Prediction
Actual    0    1
     0 1790   17
     1   25  168
```



Acceptance of Personal Loan

Tree Instability

```
treeClassifier = DecisionTreeClassifier(random_state=1)

scores = cross_val_score(treeClassifier, train_X, train_y, cv=5)
print('Accuracy scores of each fold: ', [f'{acc:.3f}' for acc in scores])
```

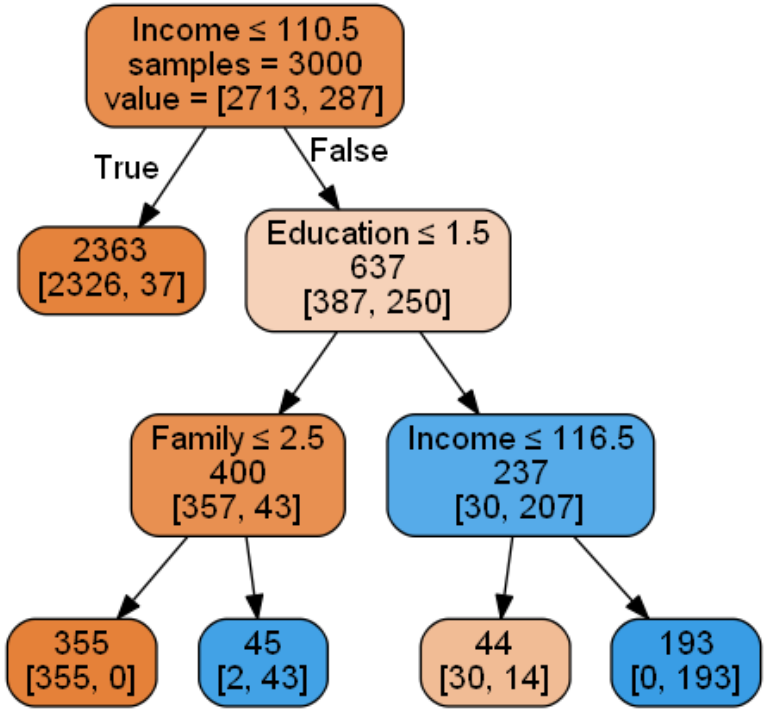
```
Accuracy scores of each fold:  ['0.988', '0.973', '0.993', '0.982', '0.993']
```

Overfitting

- Tree depth
- Minimum number of records in a terminal node
- Minimum reduction in impurity

```
smallClassTree = DecisionTreeClassifier(max_depth=30, min_samples_split=20,min_impurity_decrease=0.01, random_state=1)
smallClassTree.fit(train_X, train_y)

plotDecisionTree(smallClassTree, feature_names=train_X.columns)
```

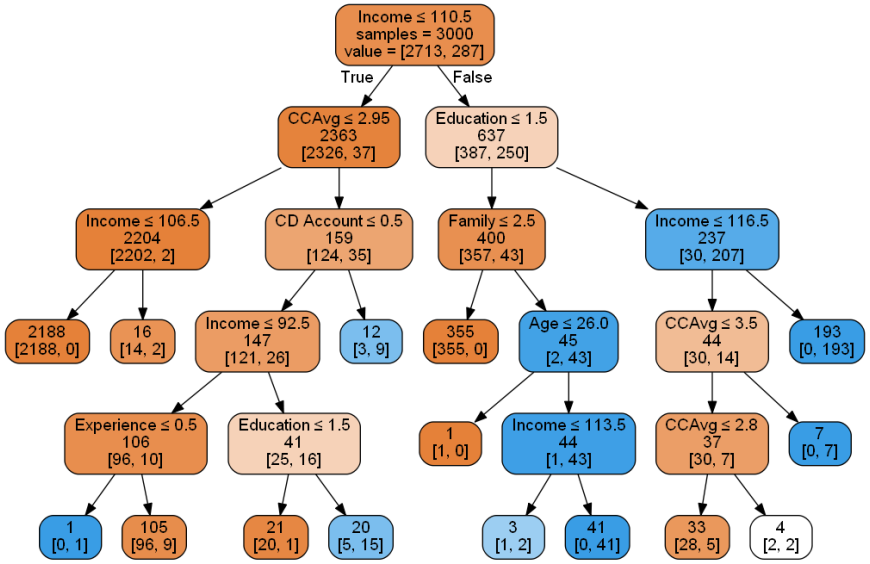
Acceptance of Personal Loan




```
Confusion Matrix (Accuracy 0.9823)

       Prediction
Actual    0    1
     0 2711    2
     1   51  236
Confusion Matrix (Accuracy 0.9770)

       Prediction
Actual    0    1
     0 1804    3
     1   43  150
```

Acceptance of Personal Loan

Fine tune method parameters



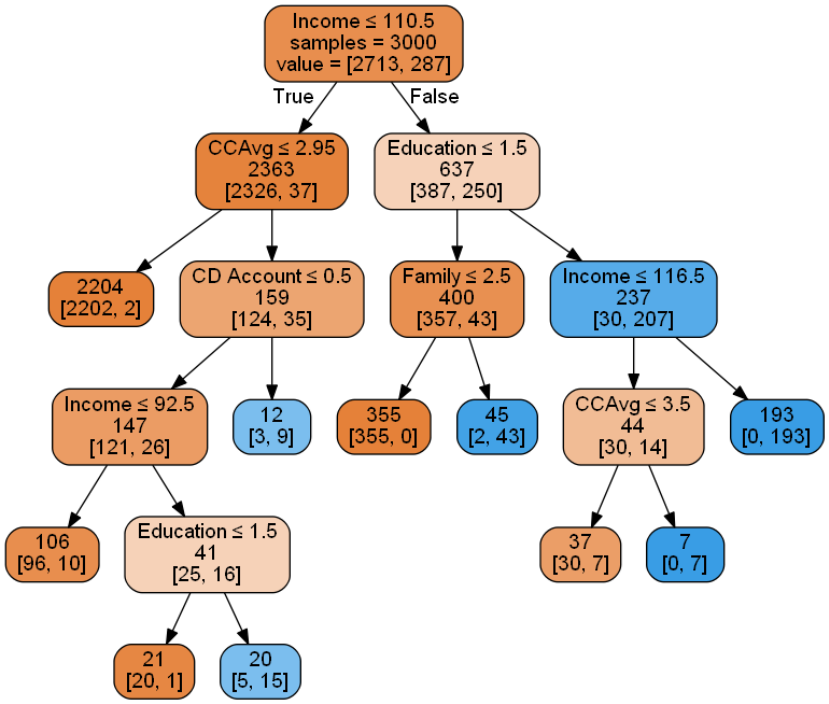

```
# Start with an initial guess for parameters
param_grid = {
    'max_depth': [10, 20, 30, 40],
    'min_samples_split': [20, 40, 60, 80, 100],
    'min_impurity_decrease': [0, 0.0005, 0.001, 0.005, 0.01],
}
gridSearch = GridSearchCV(DecisionTreeClassifier(random_state= 1), param_grid, cv=5, n_jobs=-1)
gridSearch.fit(train_X, train_y)
print('Initial score: ', gridSearch.best_score_)
print('Initial parameters: ', gridSearch.best_params_)
```

```
Initial score:  0.9876666666666667
Initial parameters:  {'max_depth': 10, 'min_impurity_decrease': 0.0005, 'min_samples_split': 20}
```

```
# Adapt grid based on result from initial grid search
param_grid = {
    'max_depth': list(range(2, 16)),  # 14 values
    'min_samples_split': list(range(10, 22)), # 11 values
    'min_impurity_decrease': [0.0001, 0.0005, 0.0009], # 3 values
}
gridSearch = GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid, cv=5,n_jobs=-1)
gridSearch.fit(train_X, train_y)
print('Improved score: ', gridSearch.best_score_)
print('Improved parameters: ', gridSearch.best_params_)

bestClassTree = gridSearch.best_estimator_
```

```
Improved score:  0.9883333333333333
Improved parameters:  {'max_depth': 5, 'min_impurity_decrease': 0.0001, 'min_samples_split': 20}
```

Acceptance of Personal Loan



Pruning

$$Cost\ of\ Complexity\ (T) = error\ (T) + \alpha\, L(T)$$

Number of terminal nodes

```
bestClassTree = DecisionTreeClassifier(max_depth=5, min_samples_split=20,min_impurity_decrease=0.0001,
                                       ccp_alpha = 0.001, random_state=1)
bestClassTree.fit(train_X, train_y)

plotDecisionTree(bestClassTree, feature_names=train_X.columns)
```