

# Decision Trees

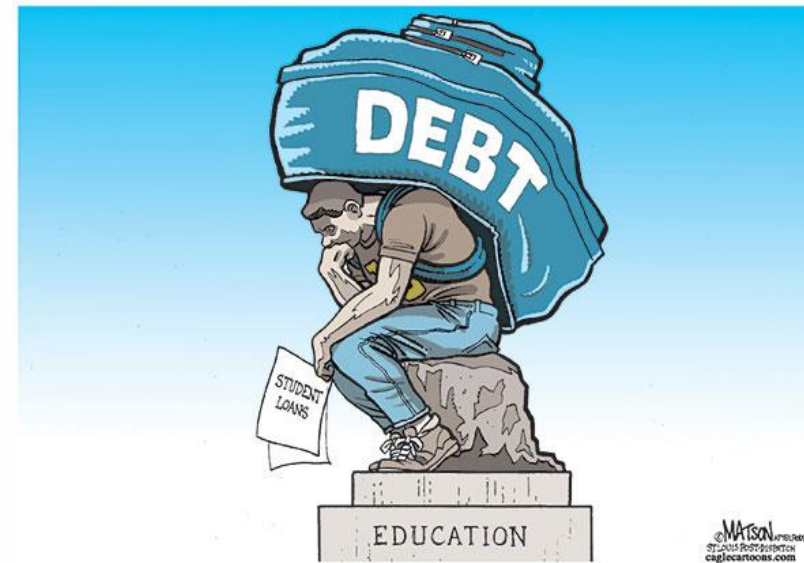
# Decision Trees

- Decision Trees create a set of binary splits on the predictor variables
- These splits are used to classify new observations into one of the two groups
- There are two categories of decision trees:
  - Classification Tree for Categorical Response Variable
  - Regression Tree for Numerical Response Variable

# Classification Trees

# Example 1: Loan Defaulter

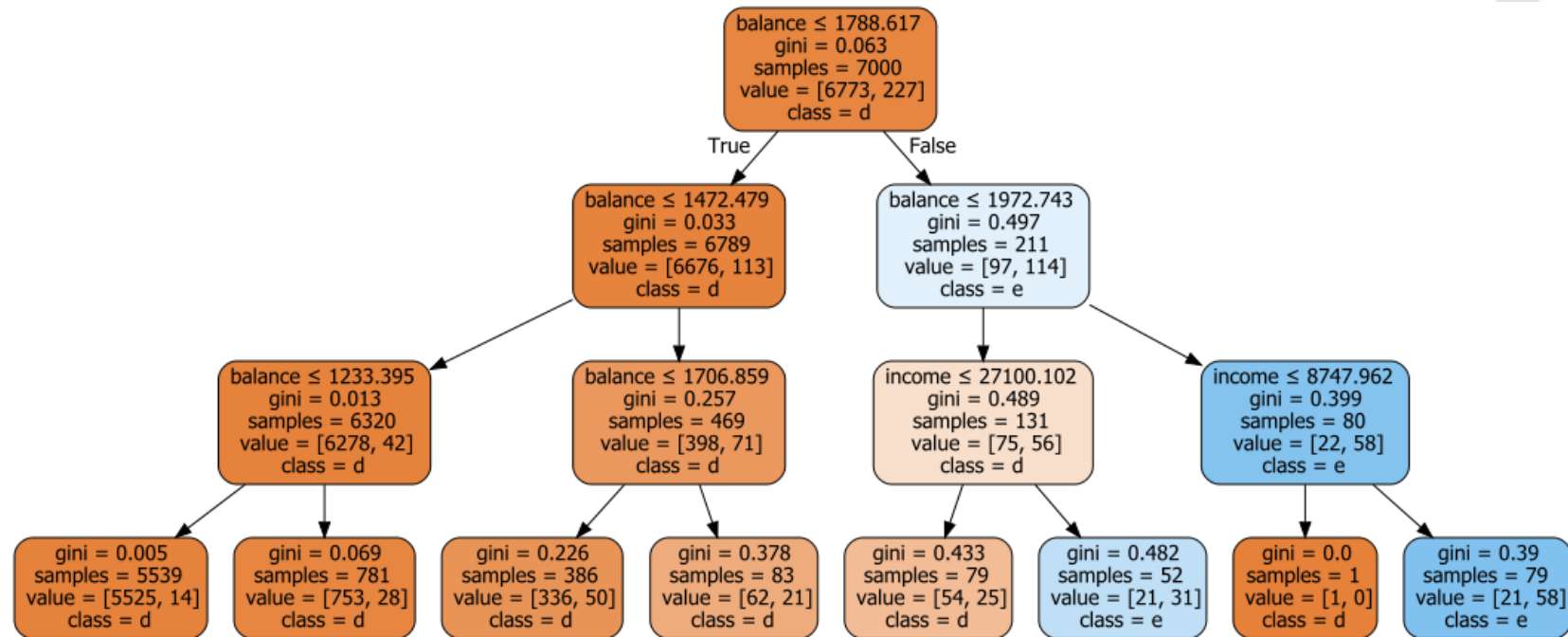
- Consider the data on loan defaulters.
- For analysis, we have taken non-defaulter + defaulters
- Data has been recorded in the file Default.csv



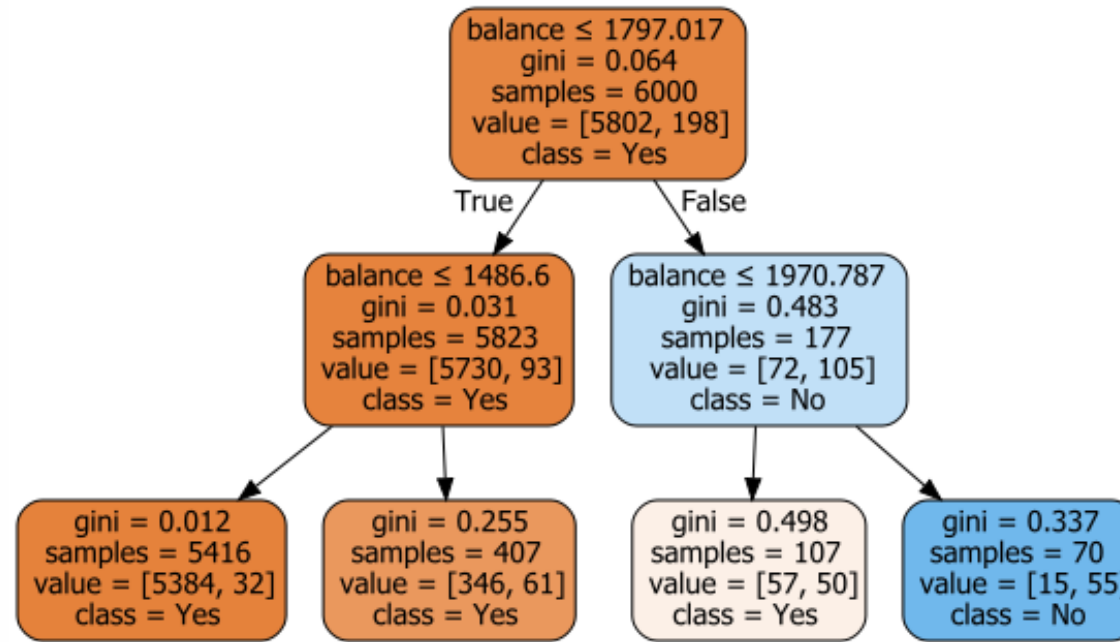
# Data

Index	default	student	balance	income
0	No	No	729.526	44361.6
1	No	Yes	817.18	12106.1
2	No	No	1073.55	31767.1
3	No	No	529.251	35704.5
4	No	No	785.656	38463.5
5	No	Yes	919.589	7491.56
6	No	No	825.513	24905.2
7	No	Yes	808.668	17600.5
8	No	No	1161.06	37468.5
9	No	No	0	29275.3
10	No	Yes	0	21871.1
11	No	Yes	1220.58	13268.6
12	No	No	237.045	28251.7
13	No	No	606.742	44994.6

# Classification Tree



# What did we gain from classification?



- For the whole training data, we had  $n(\text{Yes})=198$  and  $n(\text{No})=5802$  with proportions as 0.033 and 0.967 respectively
- By splitting on balance cut-off as 1797.017, we got two partitions with proportions as [  $P(\text{Yes})=0.016$ ,  $P(\text{No})=0.98$  ] and [  $P(\text{Yes})=0.41$ ,  $P(\text{No})=0.59$  ] respectively
- Hence we gained a **purity** or **homogeneity** in one case and lost it in other case
- The splits get proceeded for increasing purity

# Types of Decision Tree Algorithms

- There are many specific decision-tree algorithms. Notable ones include:
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 (successor of ID3)
  - CART (Classification And Regression Tree)
  - CHAID (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.
  - MARS: extends decision trees to handle numerical data better.
  - Conditional Inference Trees: Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.
- We will be covering CART



# Homogeneity Measures

- Two homogeneity measures are considered for optimization in classification tree algorithms:
  - Gini's Impurity Index
  - Entropy

# Gini's Impurity Index

- Gini impurity can be computed by summing the probability of each item being chosen times the probability of a mistake in categorizing that item.
- It reaches its minimum (zero) when all cases in the node fall into a single target category.

$$I_G(f) = \sum_{i=1}^m f_i(1 - f_i) = \sum_{i=1}^m (f_i - f_i^2) = \sum_{i=1}^m f_i - \sum_{i=1}^m f_i^2 = 1 - \sum_{i=1}^m f_i^2 = \sum_{i \neq k} f_i f_k$$

# Entropy / Log Loss

- Entropy can be computed by summing the term of product of prevalence probability and its log
- It reaches its minimum (zero) when all cases in the node fall into a single target category.

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^n p_{i,k} \log(p_{i,k})$$

Where

$p_{i,k}$ : Probability of Prevalence of class k for the  $i^{\text{th}}$  node

log function is to the base 2

## How does CART calculate the threshold for splitting ?

- CART is a popular among all the tree algorithms
- Used in libraries like rpart ( R ) and scikit-learn ( Python )
- The algorithm goes on splitting the node with a feature  $k$  and a threshold  $t_k$  for which the following cost function is minimized

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

Where

$m_{left}$ : number of observations in the left sub-setted node

$G_{left}$ : Impurity of the left sub-setted node

$m_{right}$ : number of observations in the right sub-setted node

$G_{right}$ : Impurity of the right sub-setted node

# Classical Decision Trees Algorithm

- Response : Categorical outcome variable
- Predictors : Categorical and/or Continuous Variables.
- Steps are as follows:
  1. Predictor variable gets chosen in such a way that it best splits the data into two groups with maximized purity.
    - a) If the predictor is continuous, then cut-point is chosen for maximizing the purity
    - b) If the predictor is categorical ( Not for in sklearn ), then the categories are combined together to obtain two groups with maximized purity
  2. The data is separated into two groups and the process for each subgroup is continued
  3. Steps 1 and 2 are repeated until a subgroup is obtained containing fewer number of observations than a minimum number specified or the algorithm may terminate if further splitting doesn't increase purity beyond a specific threshold

# Classical Decision Trees Algorithm

- The subgroups in the lowest branch of the tree are called terminal nodes or leaf nodes.
- Each leaf node is classified as one single category of the outcome
- For classifying any observation in the validation / test data set, that observation is traversed through the branches of tree and leaf node at which the traversal stops is predicted as the outcome of the observation.

# Package tree from scikit-learn

- We need to instantiate the DecisionTreeClassifier first
- Then, call fit function on training data
- Finally, we predict on test data

```
clf = tree.DecisionTreeClassifier(max_depth=2)
clf2 = clf.fit(X_train, y_train)
```

```
y_pred = clf2.predict(X_test)
```

# DecisionTreeClassifier( )

Syntax :

```
sklearn.tree.DecisionTreeClassifier(criterion='gini', max_depth=None, ...)
```

Where

criterion : split criterion; “gini” or “entropy”

max\_depth : The maximum depth of the tree



# Program and Output

```
# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4,
                                                    random_state=42)
```

```
clf = tree.DecisionTreeClassifier(max_depth=2)
clf2 = clf.fit(X_train, y_train)
```

```
y_pred = clf2.predict(X_test)
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
In [111]: y_pred = clf2.predict(X_test)
...:
...: print(confusion_matrix(y_test, y_pred))
...: print(classification_report(y_test, y_pred))
```

[[3850	15]				
[ 100	35]]				
		precision	recall	f1-score	support
	0	0.97	1.00	0.99	3865
	1	0.70	0.26	0.38	135
avg / total		0.97	0.97	0.96	4000

# Regularization Hyper-parameters

- If the tree is made fully to grow unconstrained, then it specializes itself only to the training data. In other words, it overfits.
- Hence, we need to put some constraints to its growth to minimize the risk of overfitting.
- The following can be some of the regularization parameters for controlling the growth:
  - Maximum depth of the tree - **max\_depth**
  - Minimum number of observations a node must have before a split is applied on it - **min\_samples\_split**
  - Minimum number of observations a leaf node must have – **min\_samples\_leaf**

Questions?