PROFESSIONAL CERTIFICATE IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Let's give everyone a couple of minutes to join…

Module 14
Decision Trees

Office Hours with Viviana Márquez
December 12, 2024

Berkeley Engineering | Berkeley Haas

Happy Holidays!

Holiday break:
22 December 2024 - 05 January 2025

Berkeley Engineering | BerkeleyHaas

AGENDA

- Content review Module 14: Decision Trees
- Code examples from the industry
- Questions

AGENDA

- Content review Module 14: Decision Trees
- Code examples from the industry
- Questions

👾

20 Questions
Only yes/no answers

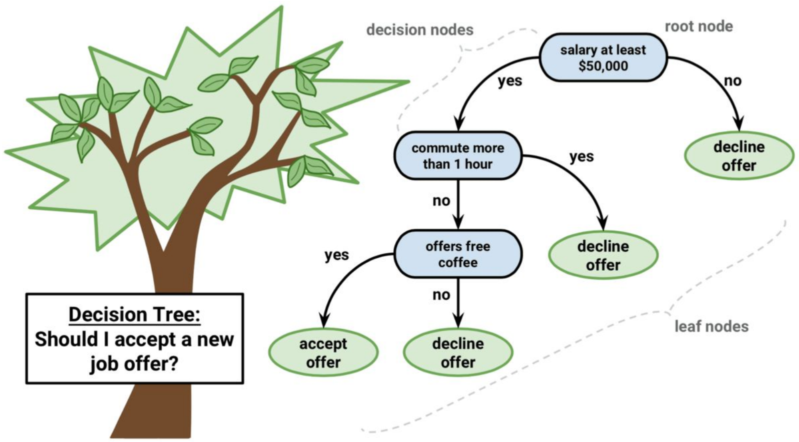
Decision Tree

- **Non-parametric SUPERVISED** machine learning model

- It can be used for both regression and classification problems

Parametric	Non-parametric
Makes assumption about form of the function of data	No assumption
Simple function	Complex
Faster	Slower & computationally more expensive
Need less data	Need more data
Underfit	Overfit
Model stays the same when number of rows increase	Model changes when number of rows increase

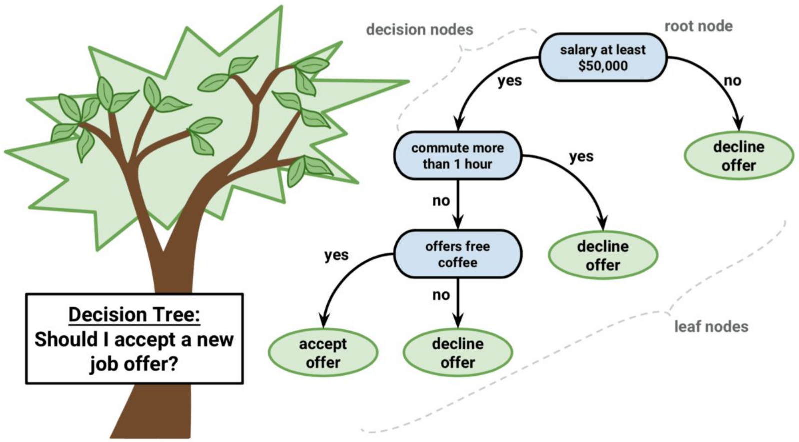
Decision Tree



- In this technique, we split the population into two or more sets based on the most significant splitter/differentiator in input variables

- Components:
  - **Root Node**: Entire population
  - **Splitting**: The process of dividing a node into two or more sub-nodes
  - **Branch**: Link between nodes
  - **Decision node**: When a sub-note splits into further sub-nodes, it's called the decision node
  - **Leaf/Terminal node**: Nodes that do not split
  - **Pruning:** Removal of sub-nodes of a decision note

Decision Tree



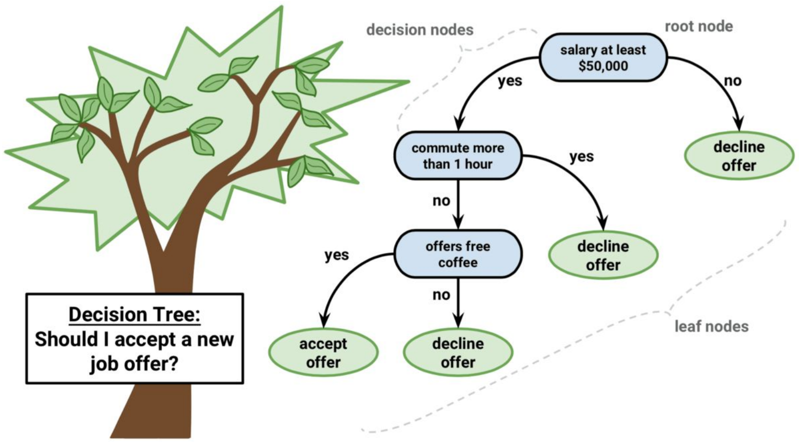
👍 **ADVANTAGES:**

- Interpretable

👎 **DISADVANTAGES:**

- Prone to overfitting

Decision Tree

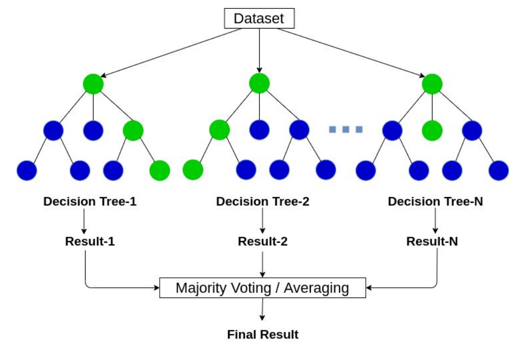


👍 ADVANTAGES:

- Interpretable

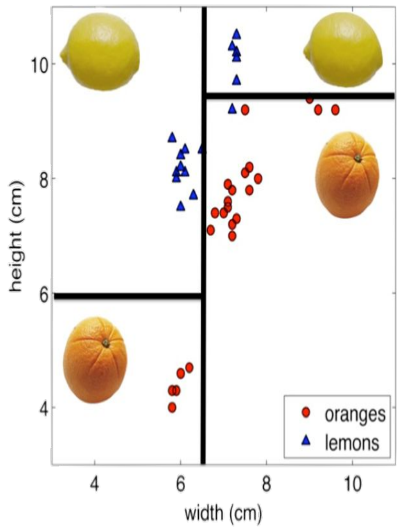
👎 DISADVANTAGES:

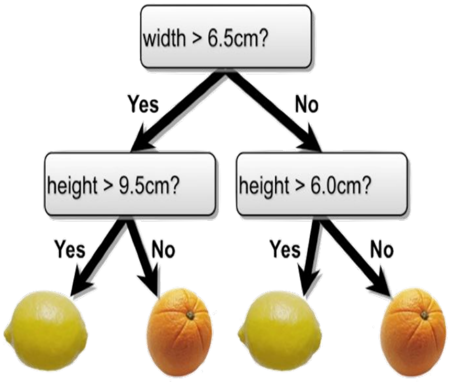
- Prone to overfitting



Random Forest
Ensemble learning method that reduces risk of overfitting

Splits in Decision Tree





- Decision Trees start with a single node, then divide the dataset on the feature that results in the largest **information gain** (IG)

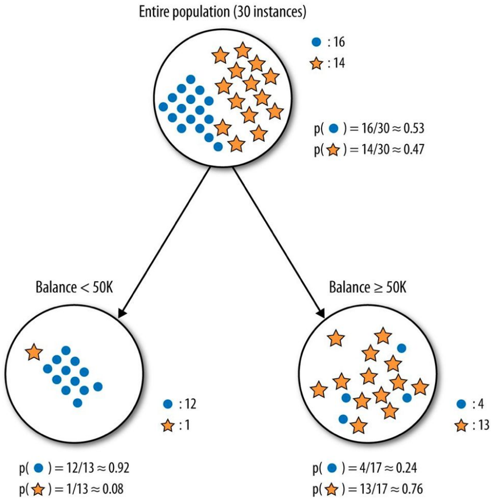
- This process is repeated, creating an increasingly accurate prediction as we traverse down the tree. This is recursive partitioning

- The recursion continues until a **stopping criteria** is met, for example, no further information gain, or a certain tree depth is reached

- Every flow chart tree corresponds to a partition of the feature space

Decision Tree Split Criteria

- **Impurity Measures**
  Quantifies how mixed the classes in a subset of data are. The goal when splitting nodes in a decision tree is to reduce the impurity of the child nodes as much as possible compared to the parent node



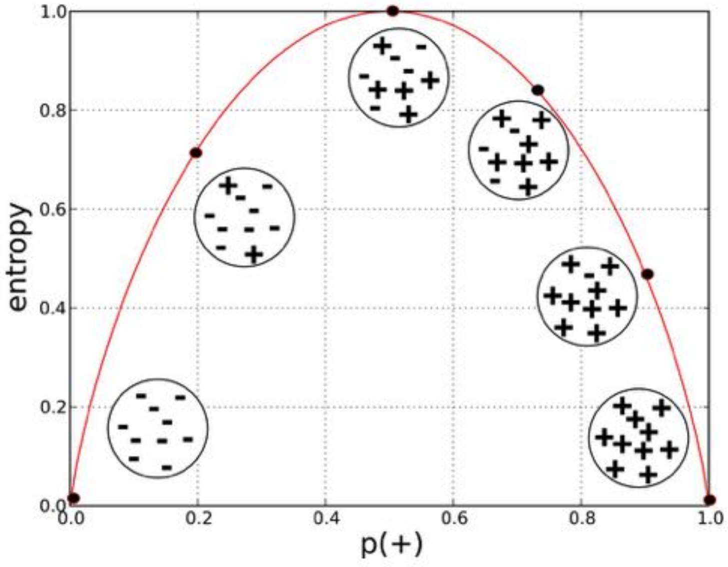
Impurity measures

- **Entropy**
  - An impurity measure
  - Ranges from 0 to 1. We want this number to be as low as possible
  - **Higher entropy** means the distribution is uniform-like (flat histogram) and thus values sampled from it are ‘less predictable’ (all possible values are equally probable)
  - **Lower entropy** means the distribution has more defined peaks and valleys and thus values sampled from it are ‘more predictable’ (values around the peaks are more probable)

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Entropy

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$



So if we had a total of 100 data points in our dataset with 30 belonging to the positive class and 70 belonging to the negative

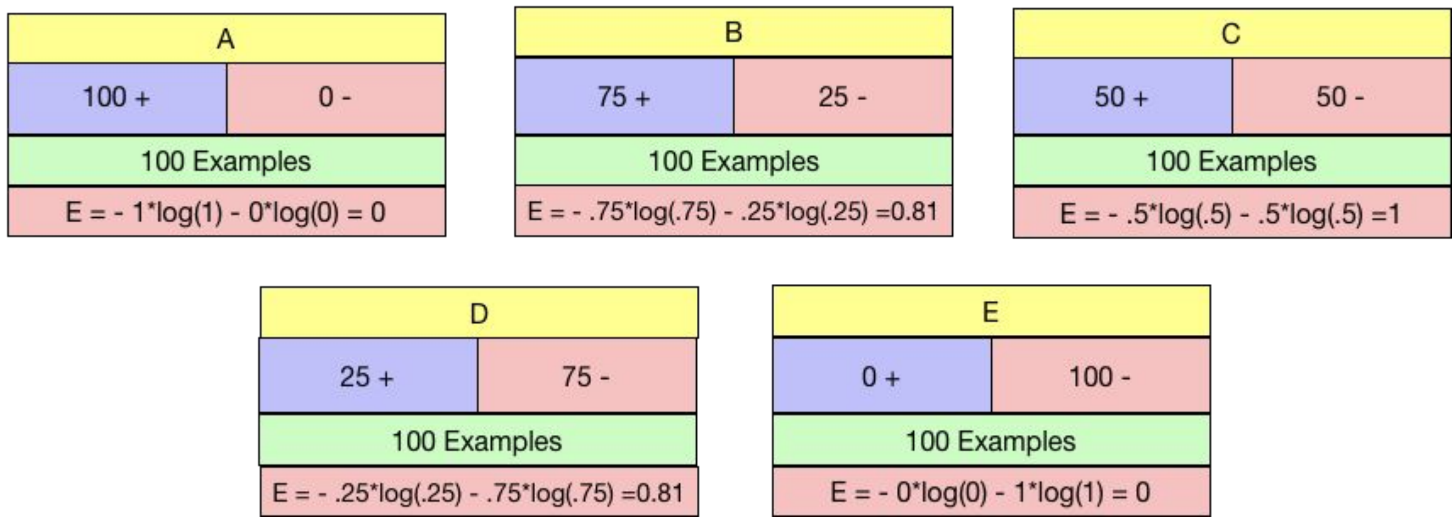
$$-\frac{3}{10} \times \log_2\left(\frac{3}{10}\right) - \frac{7}{10} \times \log_2\left(\frac{7}{10}\right) \approx 0.88$$

Information Gain: The information gain is based on the decrease in entropy after a data-set is split on an attribute.

Information Gain = How much Entropy we removed

If the sample is completely homogeneous the entropy is zero and if the sample is equally divided then it has entropy of one

Entropy


A
100 +
0 -
100 Examples
E = - 1*log(1) - 0*log(0) = 0
B
75 +
25 -
100 Examples
E = - .75*log(.75) - .25*log(.25) =0.81
C
50 +
50 -
100 Examples
E = - .5*log(.5) - .5*log(.5) =1
D
25 +
75 -
100 Examples
E = - .25*log(.25) - .75*log(.75) =0.81
E
0 +
100 -
100 Examples
E = - 0*log(0) - 1*log(1) = 0

- Example

Decision trees

Entropy

weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

11 yes
14 no

$$H(R) = -\frac{11}{25}\log_2\left(\frac{11}{25}\right) - \frac{14}{25}\log_2\left(\frac{14}{25}\right) = 0.989.$$

Decision trees

Entropy

weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

11 yes
14 no

$$H(R) = -\frac{11}{25}\log_2\left(\frac{11}{25}\right) - \frac{14}{25}\log_2\left(\frac{14}{25}\right) = 0.989.$$

If we split on the ‘day’ predictor, we get

$$\text{weekday} \rightarrow 13 \text{ no}, 7 \text{ yes}; \qquad \text{weekend} \rightarrow 1 \text{ no}, 4 \text{ yes}.$$

13/20 = 0.65; 7/20 = 0.35
1/5 = 0.8; 4/5 = 0.2

The information gain is

Day

$$0.989 + \frac{20}{25}\left(0.65 \cdot \log_2(0.65) + 0.35 \cdot \log_2(0.35)\right) + \frac{5}{25}\left(0.8 \cdot \log_2(0.8) + 0.2 \cdot \log_2(0.2)\right) = 0.097.$$

Decision trees

Entropy

weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

11 yes
14 no

$$H(R) = -\frac{11}{25}\log_2\left(\frac{11}{25}\right) - \frac{14}{25}\log_2\left(\frac{14}{25}\right) = 0.989.$$

If we split on the 'day' predictor, we get

$$\text{weekday} \to 13 \text{ no}, 7 \text{ yes};$$
$$\text{weekend} \to 1 \text{ no}, 4 \text{ yes}.$$

13/20 = 0.65; 7/20 = 0.35
1/5 = 0.8; 4/5 = 0.2

The information gain is

Proportion of training data in the child node

Day

$$0.989 + \boxed{\frac{20}{25}}(0.65 \cdot \log_2(0.65) + 0.35 \cdot \log_2(0.35)) + \frac{5}{25}(0.8 \cdot \log_2(0.8) + 0.2 \cdot \log_2(0.2)) = 0.097.$$

Decision trees

Entropy

weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

11 yes
14 no

$$H(R) = -\frac{11}{25}\log_2\left(\frac{11}{25}\right) - \frac{14}{25}\log_2\left(\frac{14}{25}\right) = 0.989.$$

If we split on the 'day' predictor, we get

$$\text{weekday} \to 13 \text{ no}, 7 \text{ yes};$$
$$\text{weekend} \to 1 \text{ no}, 4 \text{ yes}.$$

13/20 = 0.65; 7/20 = 0.35
1/5 = 0.8; 4/5 = 0.2

The information gain is

Proportion of training data in the child node

Day

$$0.989 + \boxed{\frac{20}{25}}(0.65 \cdot \log_2(0.65) + 0.35 \cdot \log_2(0.35)) + \frac{5}{25}(0.8 \cdot \log_2(0.8) + 0.2 \cdot \log_2(0.2)) = 0.097.$$

Weather

$$0.989 + \frac{7}{25}(0.14 \cdot \log_2(0.14) + 0.86 \cdot \log_2(0.86)) + \frac{18}{25}(0.277 \cdot \log_2(0.277) + 0.723 \cdot \log_2(0.723)) = 0.21.$$

Time

$$0.989 + \frac{12}{25}(0.42 \cdot \log_2(0.42) + 0.58 \cdot \log_2(0.58)) + \frac{13}{25}(0.53 \cdot \log_2(0.53) + 0.47 \cdot \log_2(0.47)) = 0.056.$$

Decision trees

Entropy

weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

11 yes
14 no

$$H(R) = -\frac{11}{25}\log_2\left(\frac{11}{25}\right) - \frac{14}{25}\log_2\left(\frac{14}{25}\right) = 0.989.$$

If we split on the 'day' predictor, we get

weekday $\rightarrow$ 13 no, 7 yes;
weekend $\rightarrow$ 1 no, 4 yes.

13/20 = 0.65; 7/20 = 0.35
1/5 = 0.8; 4/5 = 0.2

The information gain is

Proportion of training data in the child node

Day

$$0.989 + \boxed{\frac{20}{25}}(0.65 \cdot \log_2(0.65) + 0.35 \cdot \log_2(0.35)) + \frac{5}{25}(0.8 \cdot \log_2(0.8) + 0.2 \cdot \log_2(0.2)) = 0.097.$$

Weather

$$0.989 + \frac{7}{25}(0.14 \cdot \log_2(0.14) + 0.86 \cdot \log_2(0.86)) + \frac{18}{25}(0.277 \cdot \log_2(0.277) + 0.723 \cdot \log_2(0.723)) = 0.21.$$

Time

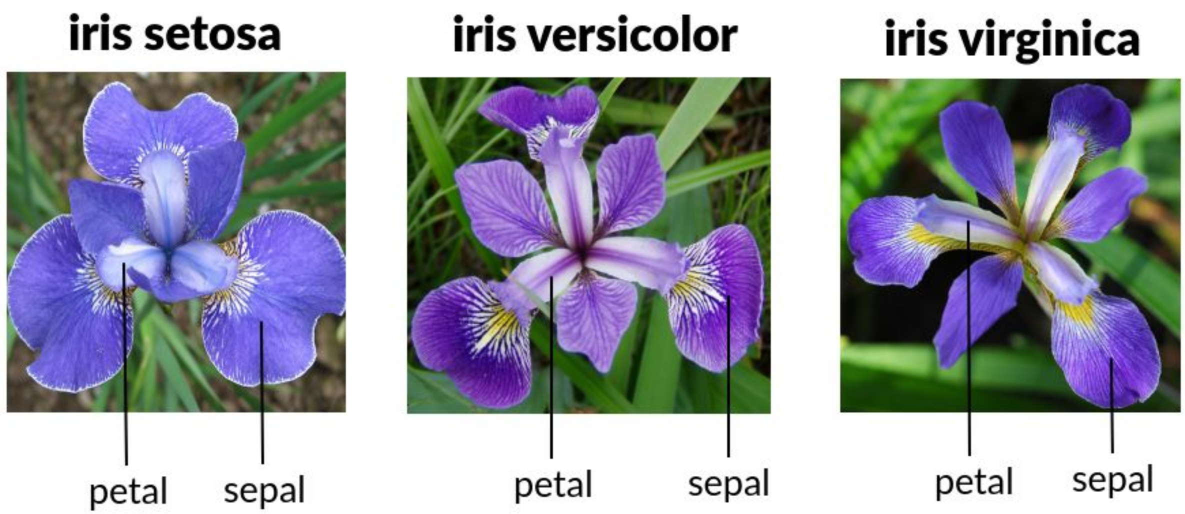
$$0.989 + \frac{12}{25}(0.42 \cdot \log_2(0.42) + 0.58 \cdot \log_2(0.58)) + \frac{13}{25}(0.53 \cdot \log_2(0.53) + 0.47 \cdot \log_2(0.47)) = 0.056.$$

AGENDA

- ✅ Required activities for Module 14
- ✅ Content review Module 14: Decision Trees
- Code examples from the industry
- Questions

Code

- https://colab.research.google.com/drive/10BXb1sHbM-qxGyNQddzVHiHi_PYZKpIC
- https://colab.research.google.com/drive/1sPaW1R6Y9joUjbPMKc3-Y28xAs2cEIWY#scrollTo=9rrRZHI1b3FB



AGENDA

- ✅ Required activities for Module 14
- ✅ Content review Module 14: Decision Trees
- ✅ Code examples from the industry
- Questions

Berkeley Engineering | BerkeleyHaas

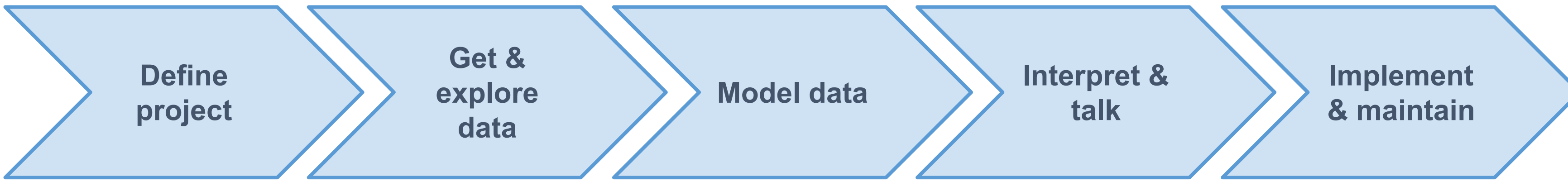
QUESTIONS?

AGENDA

- ✅ Required activities for Module 14
- ✅ Content review Module 14: Decision Trees
- ✅ Code examples from the industry
- ✅ Questions

APPENDIX

The Machine Learning pipeline



Define project

- Specify business problem
- Acquire domain knowledge

Get and explore data

- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

Model data

- Determine ML task
- Build candidate models
- Select model based on performance metrics

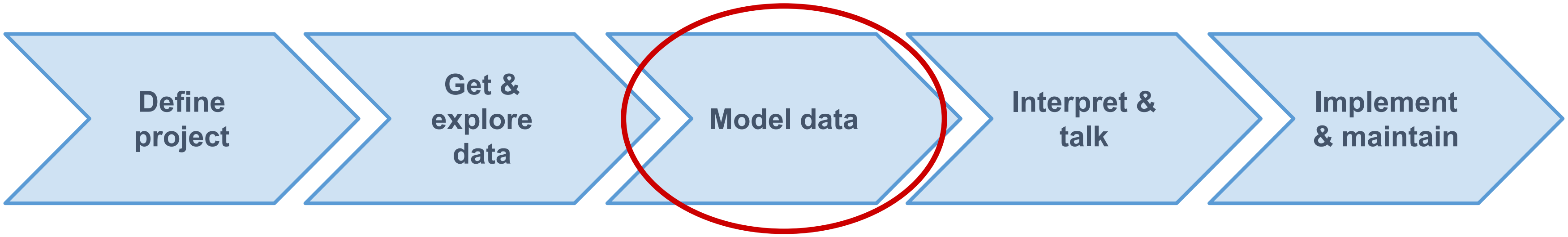
Interpret & talk

- Interpret model
- Communicate model insights

Implement & maintain

- Set up function to predict on new data
- Document process
- Monitor and maintain model

The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

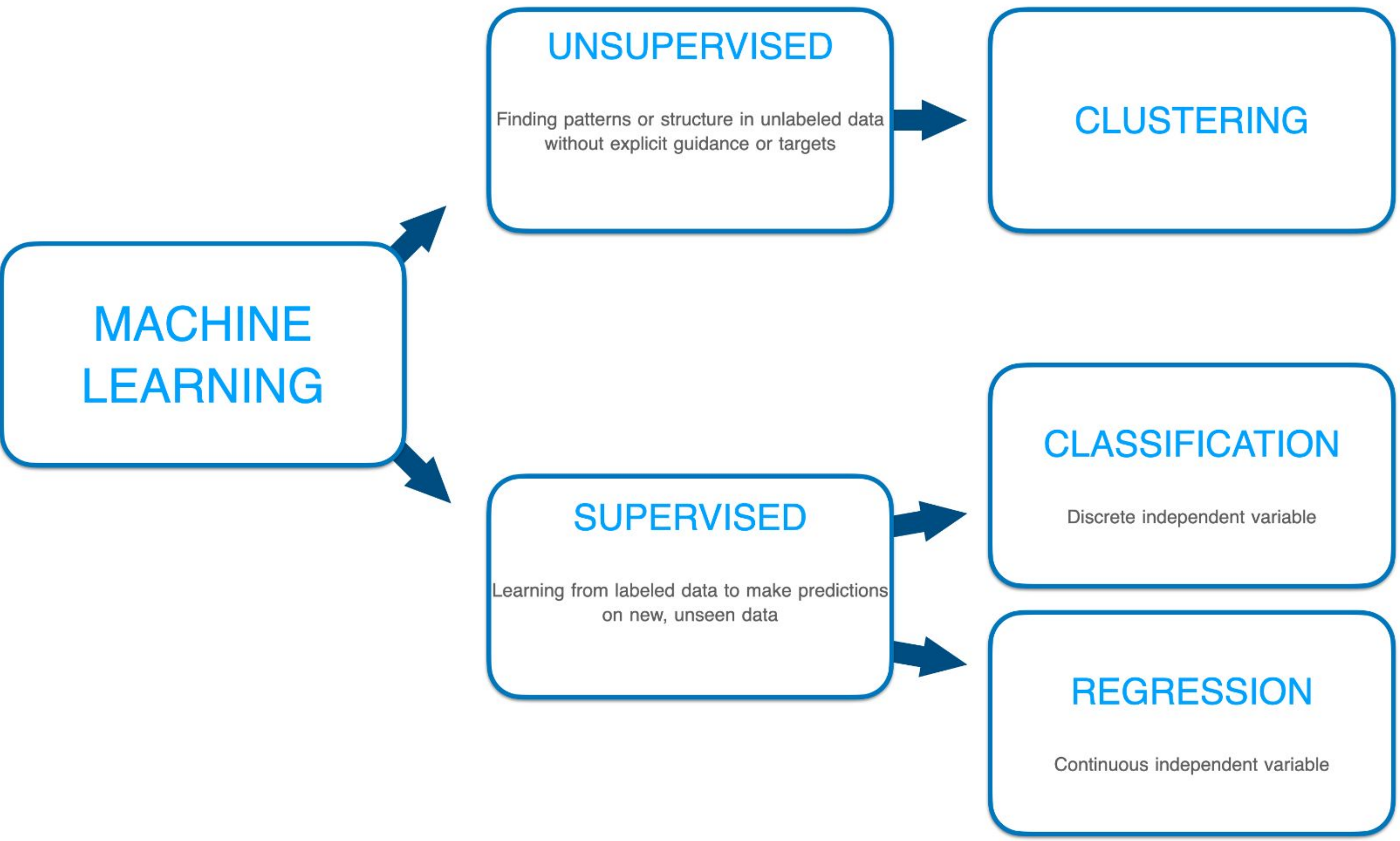
Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

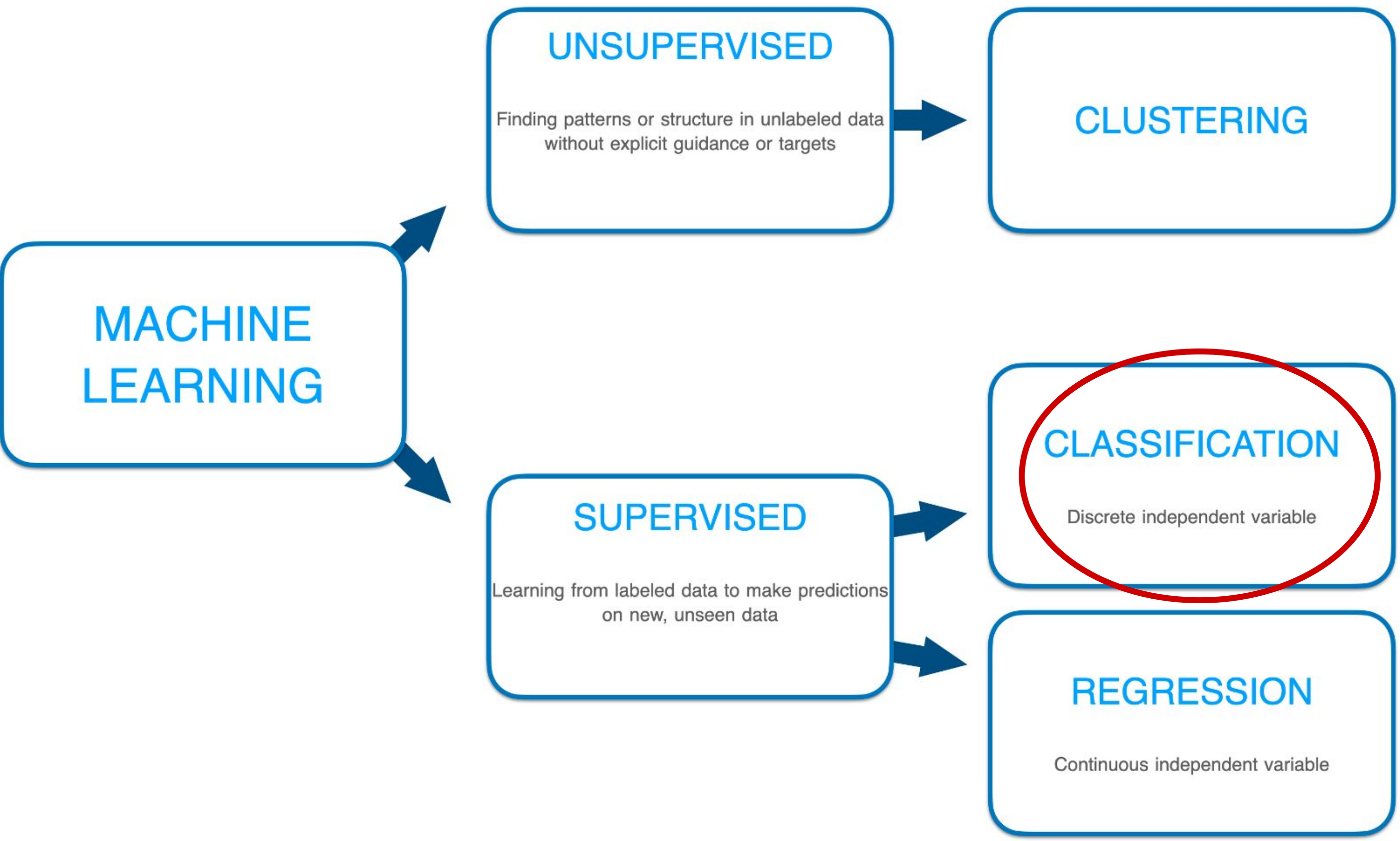
Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

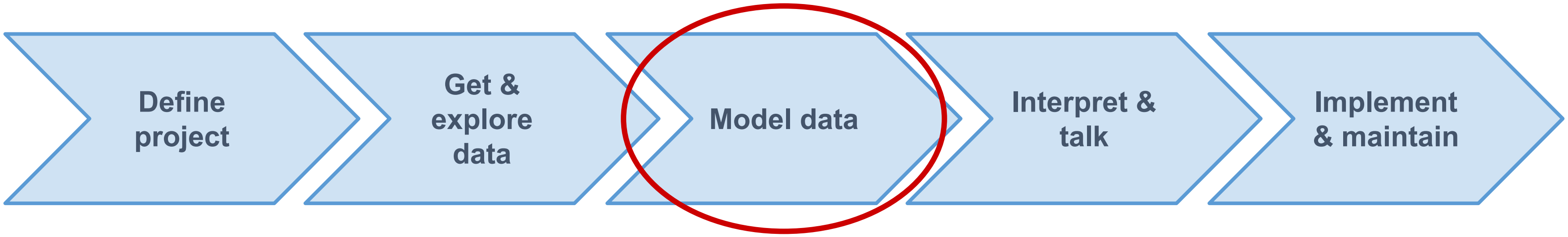
Do we have labels? Is my target variable discrete?



Do we have labels? Is my target variable discrete?



The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

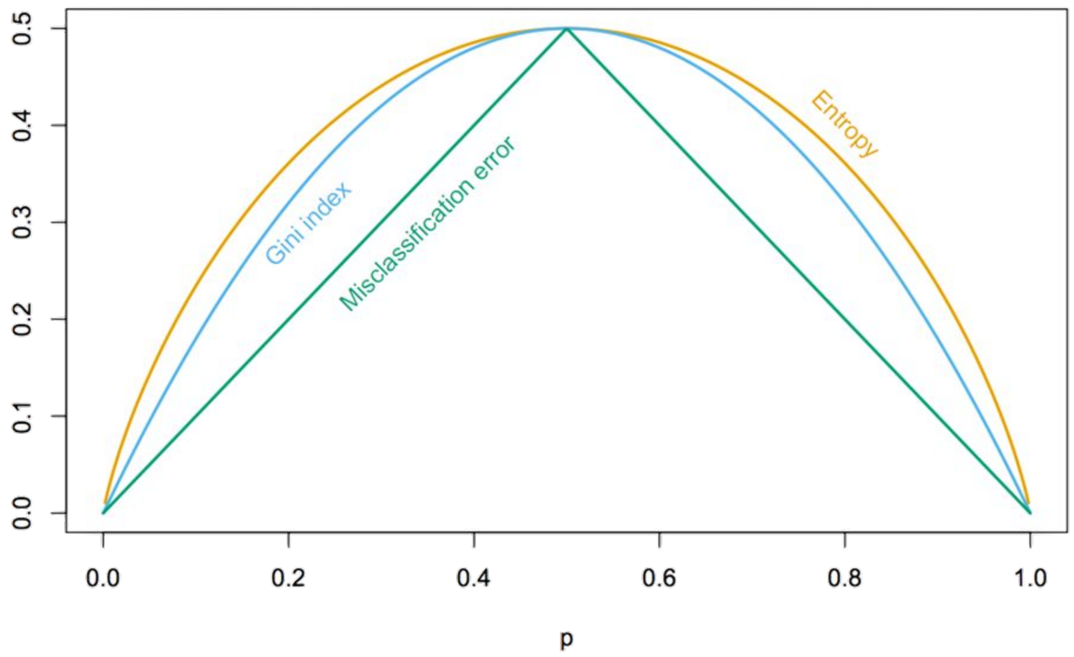
Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

Decision trees

Comparison of Criteria



Decision trees

Gini or Entropy

The choice between Gini Impurity and Entropy typically does not impact the performance of a decision tree significantly, as they often lead to similar splits. However, there are some differences between the two measures that might inform your decision on which one to use:

- Calculation Time: Gini Impurity is slightly faster to compute as it doesn't involve logarithmic functions, which might be a consideration if you're dealing with a very large dataset and computational efficiency is a concern.

- Tendency to Favor Larger Partitions: Entropy has a tendency to produce slightly more balanced trees because it is more sensitive to changes when the class probabilities are close to 0 or 1. Gini Impurity, on the other hand, tends to find the smallest class in the split.

- Bias: Gini Impurity tends to be less biased with respect to the number of classes. Entropy can be biased towards multiclass splits because with more classes, the chance of misclassifying increases, and entropy would consider this as a higher disorder (or higher entropy).

In practice, the standard approach is usually to try both and see which one gives better validation performance for your specific use case. But again, it's worth noting that in most cases, they tend to produce very similar results.

Impurity measures

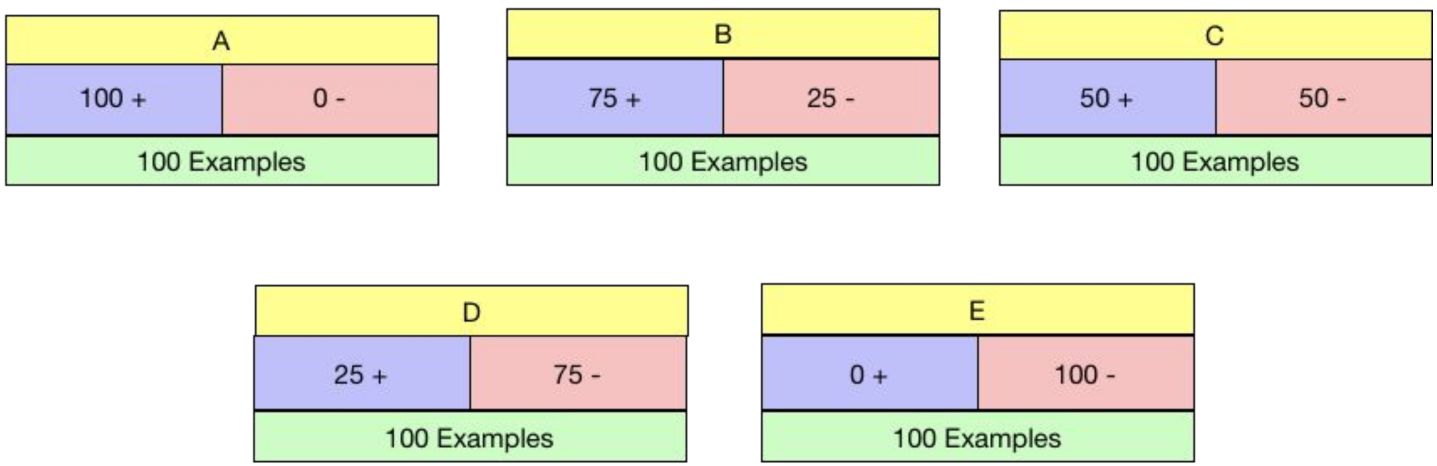
- **Gini Index**
  - An impurity measure
  - It helps us decide which features are the best to split on at each node of the tree
  - Ranges from 0 to 1. We want this number to be as low as possible

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

Gini Index

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

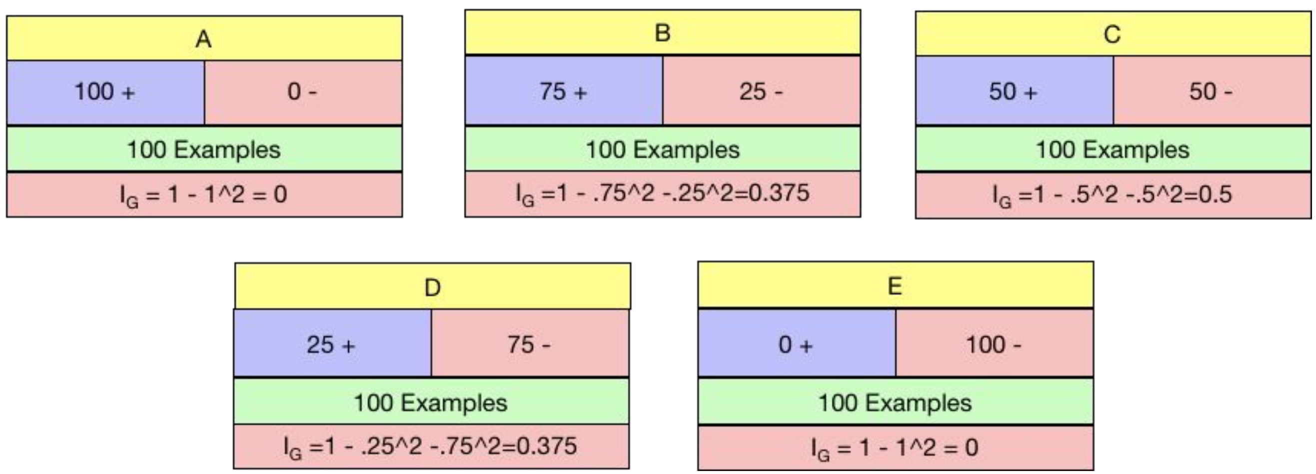
Five examples of candidate notes, which is the ideal situation to be in?



Gini Index

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

Five examples of candidate notes, which is the ideal situation to be in?



Decision trees

Gini Impurity (Example)

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
41	0	2	112	268	0	0	172	1	0.0	2	0	2	1
66	1	0	160	228	0	0	138	0	2.3	2	0	1	1
57	1	2	128	229	0	0	150	0	0.4	1	1	3	0
63	0	2	135	252	0	0	172	0	0.0	2	0	2	1

https://towardsdatascience.com/the-simple-math-behind-3-decision-tree-splitting-criterions-85d4de2a75fe

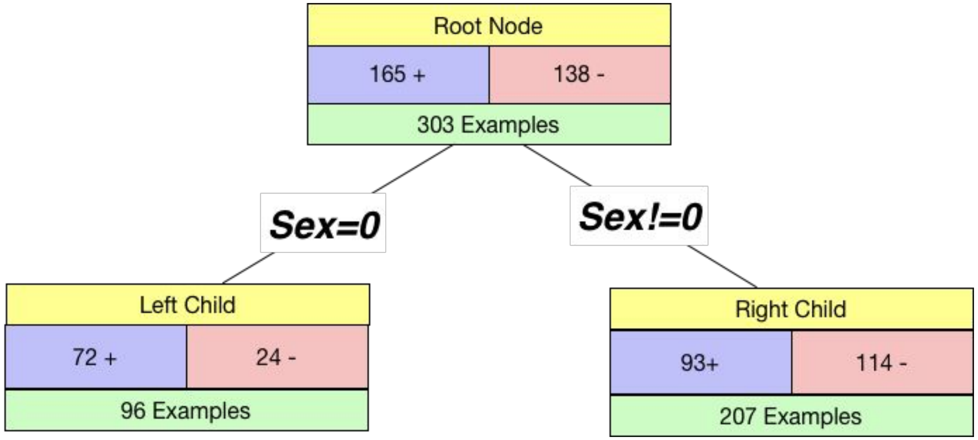
Decision trees

Gini Impurity (Example)

```
I_Left = 1 - (72/96)**2 - (24/96)**2
I_Right = 1 - (93/207)**2 - (114/207)**2

print("Left Node Impurity:",I_Left)
print("Right Node Impurity:",I_Right)
-----------------------------------------------------------------
Left Node Impurity: 0.375
Right Node Impurity: 0.4948540222642302
```

Categorical variable split (e.g. Sex)




```
gender_split_impurity = 96/(96+207)*I_Left + 207/(96+207)*I_Right
print(gender_split_impurity)
-----------------------------------------------------------------
0.45688047065576126
```

https://towardsdatascience.com/the-simple-math-behind-3-decision-tree-splitting-criterions-85d4de2a75fe

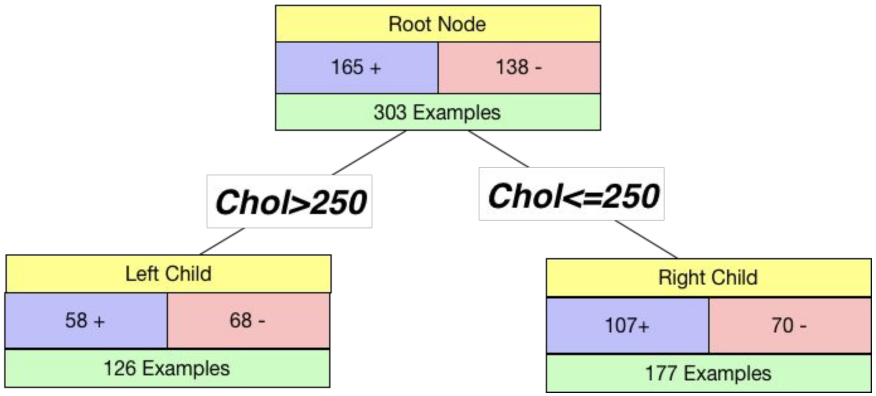
Decision trees

Gini Impurity (Example)

```
I_Left = 1 - (58/126)**2 - (68/126)**2
I_Right = 1 - (107/177)**2 - (70/177)**2

print("Left Node Impurity:",I_Left)
print("Right Node Impurity:",I_Right)
----------------------------------------------------------------
Left Node Impurity: 0.49685059208868737
Right Node Impurity: 0.47815123368125373
```

Continuous variable split (e.g. Chol)




```
chol_split_impurity = 126/(126+177)*I_Left + 177/(126+177)*I_Right
print(chol_split_impurity)
-------------------------------------------------------------
0.48592720450414695
```

https://towardsdatascience.com/the-simple-math-behind-3-decision-tree-splitting-criterions-85d4de2a75fe

Constructing a Decision Tree

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Constructing a Decision Tree

Gini = 0.49

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Constructing a Decision Tree

Gini = 0.49

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Constructing a Decision Tree

Gini = 0.49

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Sun
ny

Rai
ny

Gini = 0.0

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
2	3	Sunny	no	yes	yes
5	6	Sunny	yes	no	yes

Gini = 0.21

	Day	Weather	Just Ate	Late at Work	Will I go Running?
1	2	Rainy	yes	yes	no
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
6	7	Rainy	no	yes	no

Constructing a Decision Tree

Gini = 0.49

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Yes

Gini = 0.19

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
5	6	Sunny	yes	no	yes

No

Gini = 0.29

	Day	Weather	Just Ate	Late at Work	Will I go Running?
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
6	7	Rainy	no	yes	no

Constructing a Decision Tree

Gini = 0.49

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes
6	7	Rainy	no	yes	no

Yes

No

Gini = 0.19

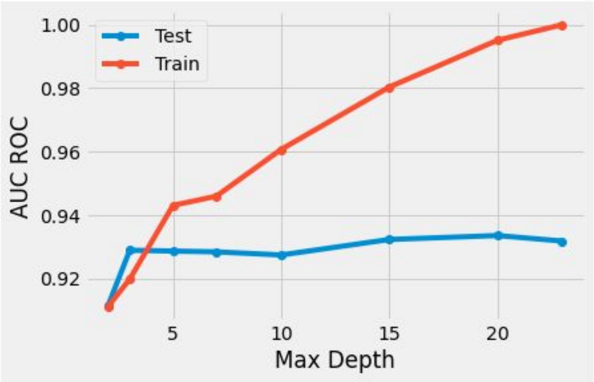
	Day	Weather	Just Ate	Late at Work	Will I go Running?
1	2	Rainy	yes	yes	no
2	3	Sunny	no	yes	yes
6	7	Rainy	no	yes	no

Gini = 0.21

	Day	Weather	Just Ate	Late at Work	Will I go Running?
0	1	Sunny	yes	no	yes
3	4	Rainy	no	no	no
4	5	Rainy	no	no	yes
5	6	Sunny	yes	no	yes

Pre-Pruning

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including **max_depth, min_samples_leaf, min_samples_split** can be tuned to early stop the growth of the tree and prevent the model from overfitting.



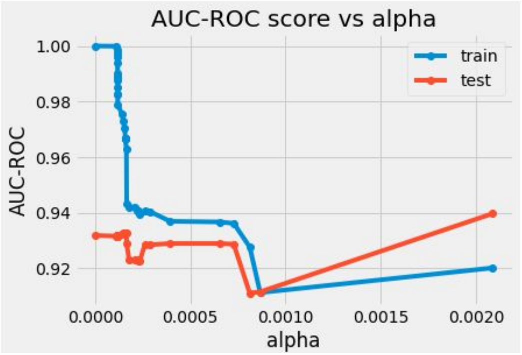
(Image by Author), AUC-ROC score vs max depth

As observed from the plot, with an increase in max_depth training AUC-ROC score continuously increases, but the test AUC score remains constants after a value of max depth. The best-fit decision tree is at a max depth value of 5. Increase the max depth value further can cause an overfitting

Post-Pruning

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting.

Cost complexity pruning (ccp) is one type of post-pruning technique. In case of cost complexity pruning, the ccp_alpha can be tuned to get the best fit model.



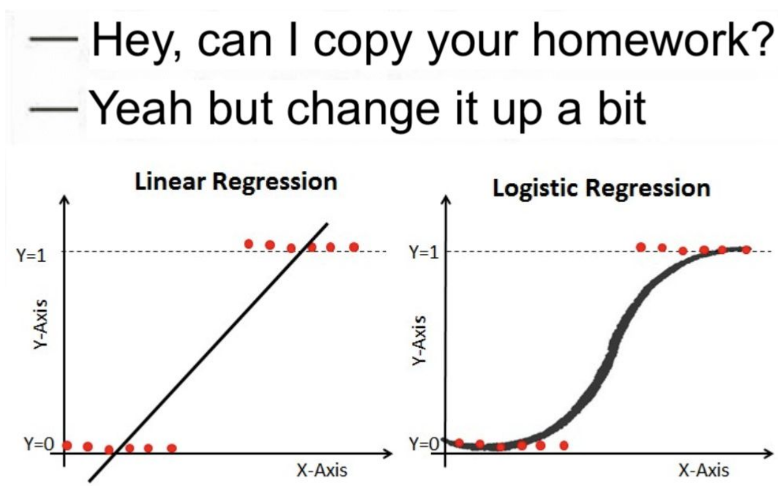
- Train decision tree classifiers with different values of ccp_alphas and compute train and test performance scores.
- Plot train and test scores for each value of ccp_alphas values.

From the above plot, ccp_alpha=0.000179 can be considered as the best parameter as

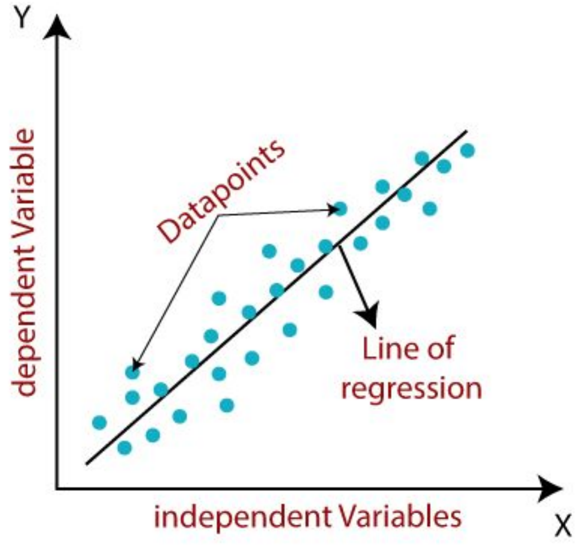
— Hey, can I copy your homework?

— Yeah but change it up a bit


— Hey, can I copy your homework?
— Yeah but change it up a bit


— Hey, can I copy your homework?
— Yeah but change it up a bit
Linear Regression
Y=1
Y-Axis
Y=0
X-Axis
Logistic Regression
Y=1
Y-Axis
Y=0
X-Axis

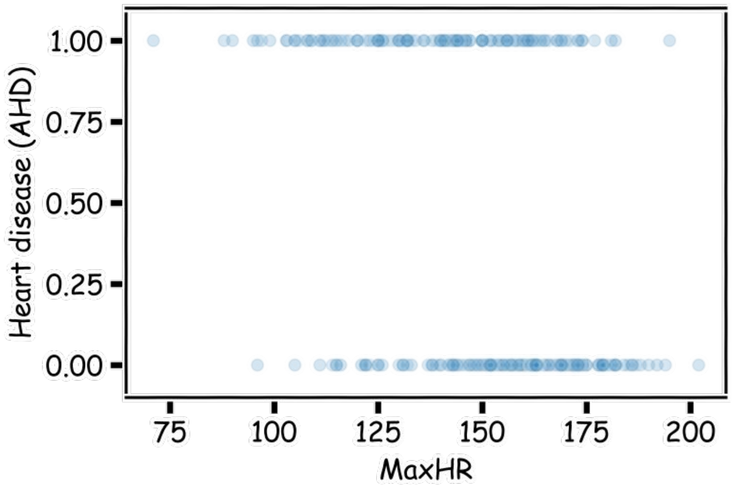
Linear Regression – Recap



- **Supervised regression** machine learning algorithm
- It assumes a linear relationship between the inputs and the output
- The algorithm tries to find the best-fitting straight line (in simple linear regression) or a hyperplane (in multiple linear regression) that describes this relationship
- This "best fit" is often determined by minimizing the difference (or error) between the predicted values and the actual observed values

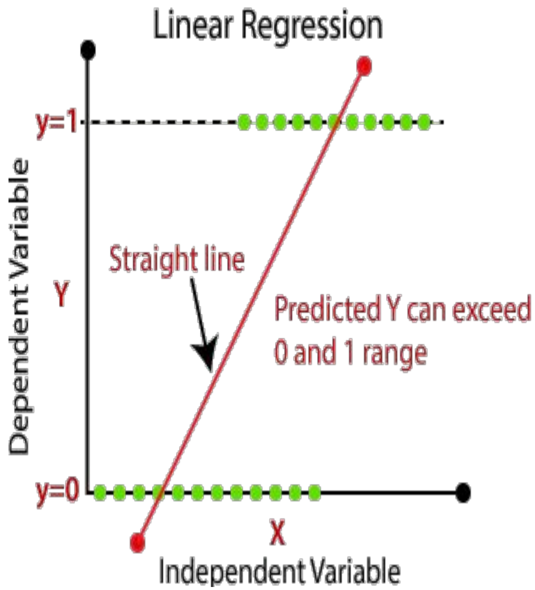
Logistic Regression

How should we use this data to predict new labels?



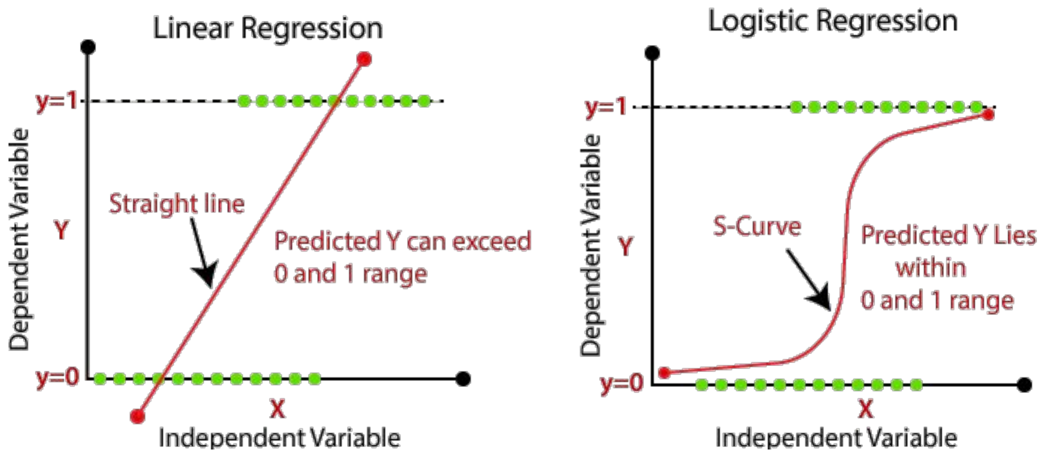
Logistic Regression

How should we use this data to predict new labels?

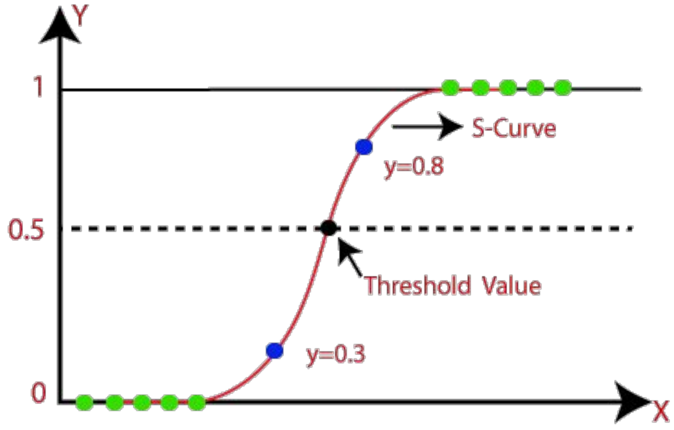


Logistic Regression

How should we use this data to predict new labels?



Logistic Regression



- **Supervised** **classification** machine learning algorithm
- Used to predict the probability of a binary outcome (1 / 0, Yes / No, True / False)
- Despite the name "regression," it's used for binary classification tasks
- It works by estimating the probability that a given instance belongs to a particular category, using a logistic function (or sigmoid) to squeeze the output between 0 and 1

Sigmoid Function

What are Odds?

In a binary event (like flipping a coin), the odds in favor of an event is defined as:

Odds(A) = Number of ways A can happen / Number of ways A cannot happen

= p / 1 - p

Sigmoid Function

What are Odds?

In a binary event (like flipping a coin), the odds in favor of an event is defined as:

Odds(A) = Number of ways A can happen / Number of ways A cannot happen
= p / 1 - p

In the case of flipping a fair coin, the odds are , a.k.a. the event is equally likely to happen (because the probability of heads is 0.5 and tails is 0.5, so 0.5/0.5 = 1).

Sigmoid Function

Log Odds or Logit

The log odds, also known as the logit, is simply the logarithm of the odds. If p is the probability of our event (like heads):

a.k.a. **Log Odds**

or **Logit**

$$\boxed{\log\left(\frac{P}{1-P}\right)}$$

Sigmoid Function

From Log Odds to Probability

In logistic regression, we model the log odds of the dependent variable using a linear combination of predictors. The equation can be stated as:

a.k.a. **Log Odds** or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

$$\frac{P}{1-P} = e^{\beta_0 + \beta_1 X}$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

$$\frac{P}{1-P} = e^{\beta_0 + \beta_1 X}$$

$$P = e^{\beta_0 + \beta_1 X} - Pe^{\beta_0 + \beta_1 X}$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

$$\frac{P}{1-P} = e^{\beta_0 + \beta_1 X}$$

$$P = e^{\beta_0 + \beta_1 X} - Pe^{\beta_0 + \beta_1 X}$$

$$P(1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

$$\frac{P}{1-P} = e^{\beta_0 + \beta_1 X}$$

$$P = e^{\beta_0 + \beta_1 X} - Pe^{\beta_0 + \beta_1 X}$$

$$P(1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$$

$$\boxed{P = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}}$$

Math gymnastics

a.k.a. **Log Odds**

or **Logit**

Intercept

$$\boxed{\log\left(\frac{P}{1-P}\right)} = \beta_0 + \beta_1 X$$

$$\frac{P}{1-P} = e^{\beta_0 + \beta_1 X}$$

$$P = e^{\beta_0 + \beta_1 X} - P e^{\beta_0 + \beta_1 X}$$

$$P(1 + e^{\beta_0 + \beta_1 X}) = e^{\beta_0 + \beta_1 X}$$

$$\boxed{P = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}}$$

Sigmoid Function

Sigmoid Function

- A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve
- Outputs a value between 0 and 1, making it great for representing probabilities
- In logistic regression, the sigmoid function transforms our linear equation's output (which can be any real number) into a bounded range of [0, 1], perfect for probabilities

Making predictions

- **Binary Classification:** For a binary classification task, you would set a threshold (commonly 0.5). If the predicted probability is greater than the threshold, you predict class 1 (or "yes"), otherwise, you predict class 0 (or "no").

- **Multiclass Classification:** Logistic regression can be extended to handle multiple classes using techniques like "One-vs-All" or "Softmax Regression."

Evaluating Logistic Regression

As a classification model, you use classification performance metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC Curve and AUC