CS 471: Introduction to AI

Module 6 Part I: Machine Learning

Learning from Examples

An agent is learning if it improves its performance after making observations about the world.

Why would we want a machine to learn? Why not just program it the right way to begin with?

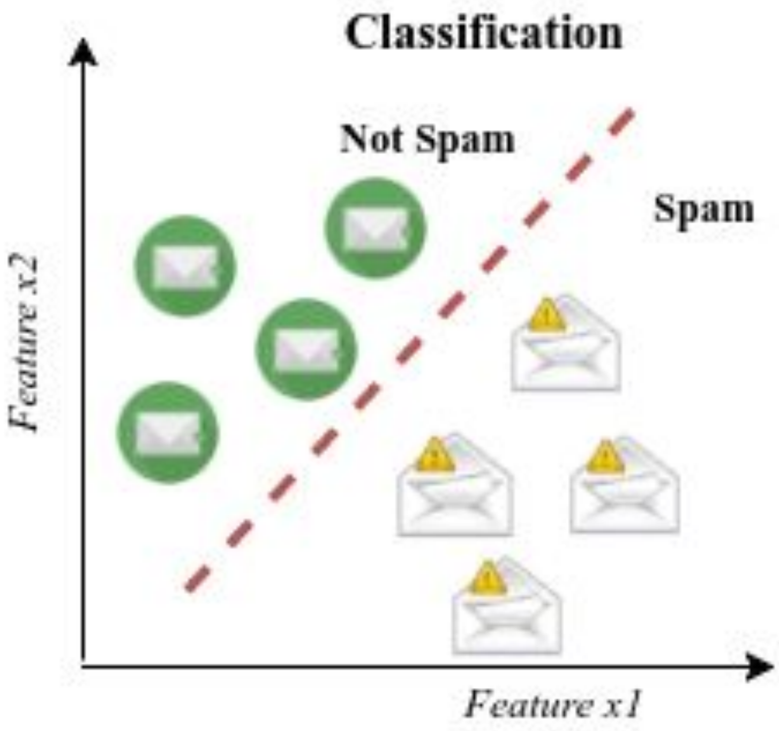
Two main reasons:

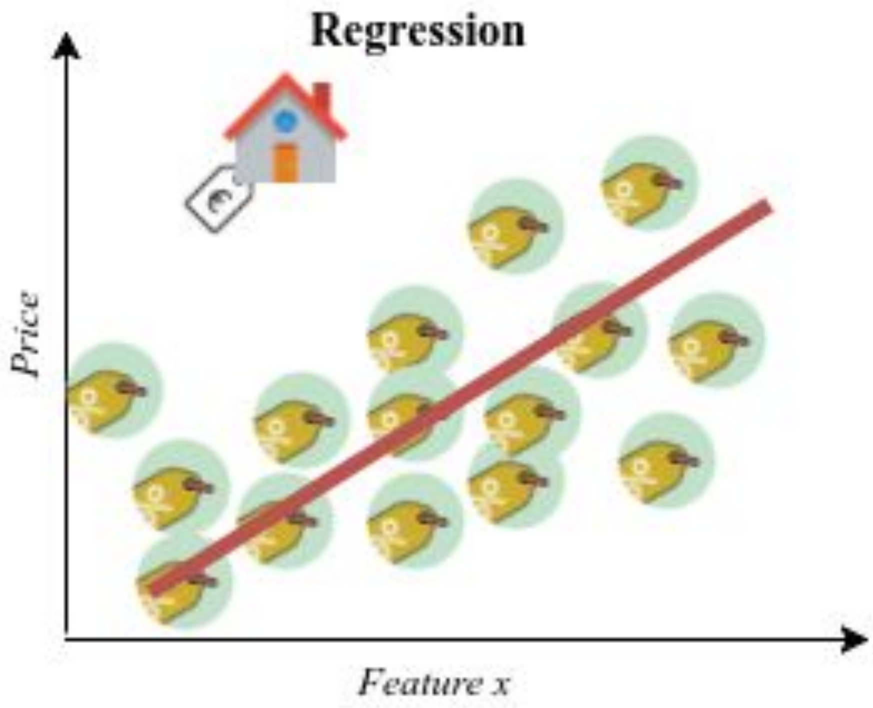
- Designers cannot anticipate all possible future situations.
- Sometimes the designers have no idea how to program a solution themselves

Forms of Learning

Classification: output is one of a finite set of values (such as sunny/cloudy/rainy or true/false)

Regression: output is a number (such as tomorrow's temperature, integer or real number)



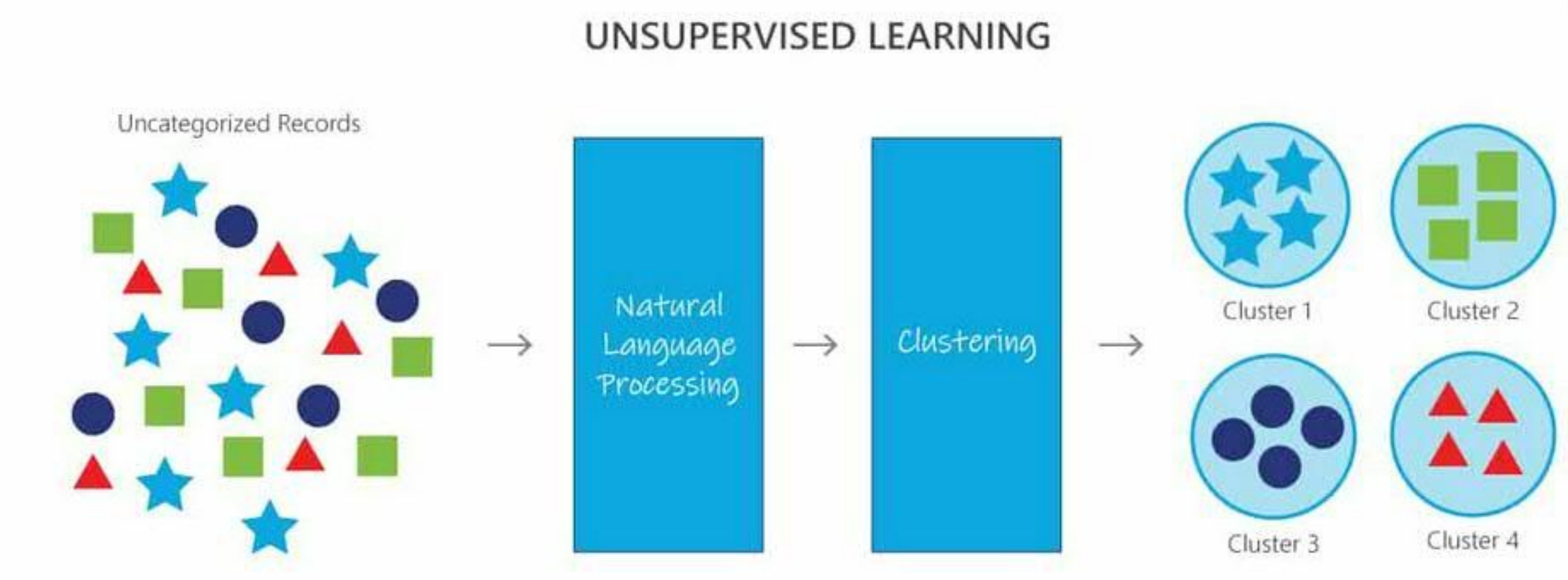


Supervised Learning

- Training data has labels
- Example: the inputs could be camera images, each one accompanied by an output saying "cat" or "dog," etc. An output like this is called a label.

Unsupervised Learning

- Training data has no labels
- Most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples.



Supervised Learning

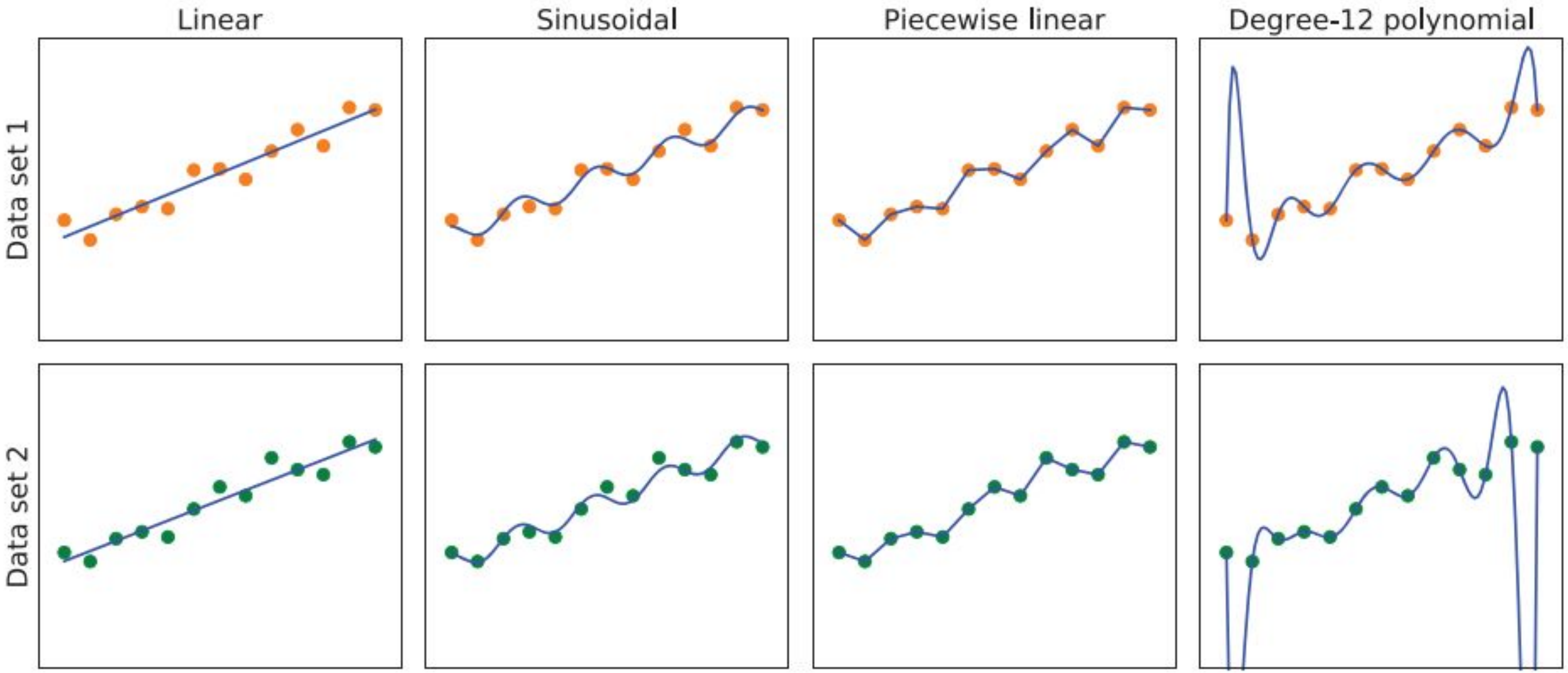
- Training set: $(x_1,y_1),(x_2,y_2),...(x_N,y_N)$

  $y = f(x)$

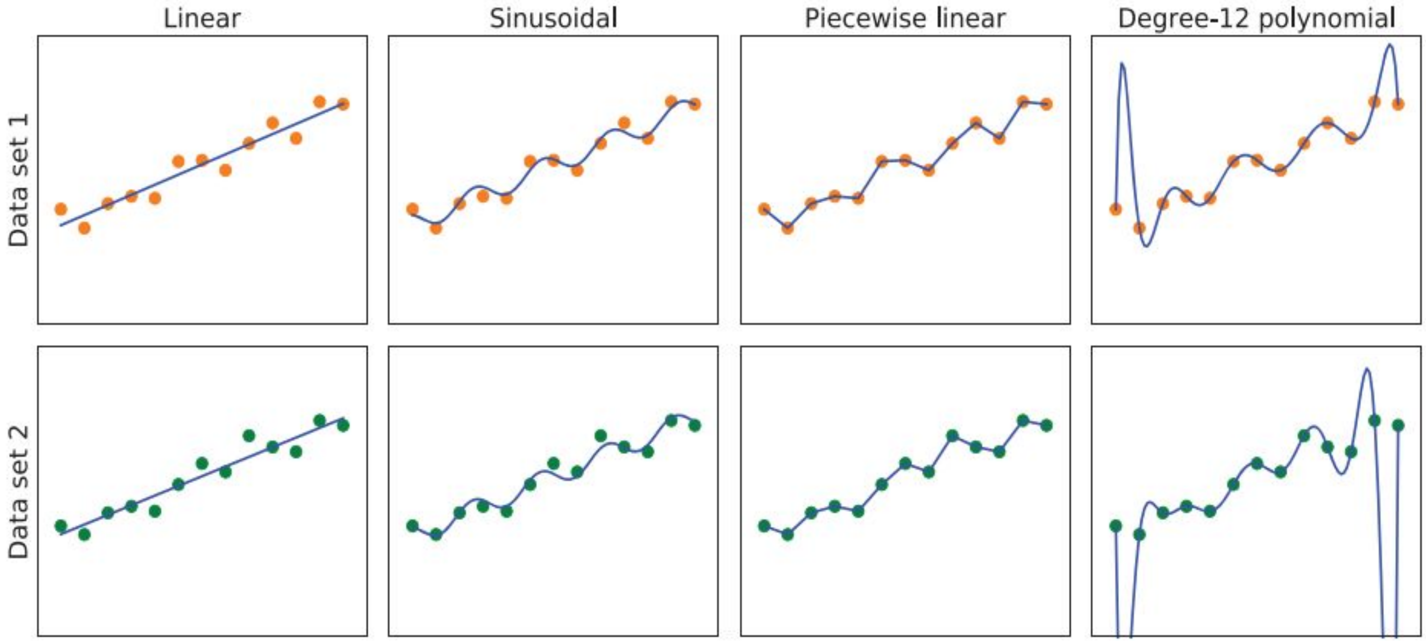
  f = true function

  h = hypothesis = model = approximation of the true function f
- We call the output y_i the ground truth: true answer we are asking our model to predict.

Choosing a good model



Finding a curve to fit data.
Top row: four plots of best-fit functions trained on data set 1.
Bottom row: the same four functions, but trained on a slightly different data set (sampled from the same f(x) function).



Column 1: Straight lines; functions of the form $h(x) = w_1x + w_0$. There is no line that would be pass through all the data points.

Column 2: Sinusoidal functions of the form $h(x) = w_1x + \sin(w_0x)$. This choice is not quite consistent, but fits both data sets very well.

Column 3: Piecewise-linear functions. These functions are always consistent.

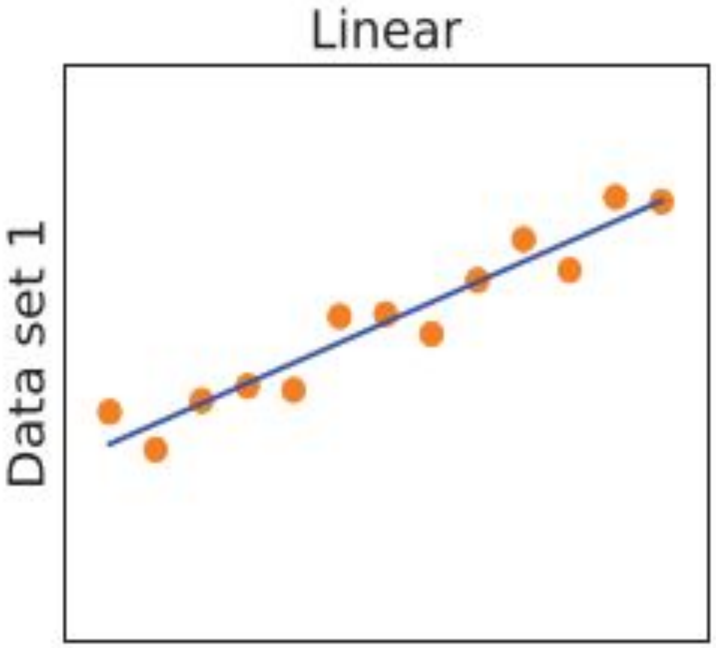
Column 4: Degree-12 polynomials. These are consistent: we can always get a degree-12 polynomial to perfectly fit 13 distinct points. This does not mean it is a good guess.

Bias

Bias: Measure of fitting training data well

Linear functions have high bias: only allows functions consisting of straight lines. Any other pattern, a linear function will not be able to represent those patterns.

High bias => underfitting

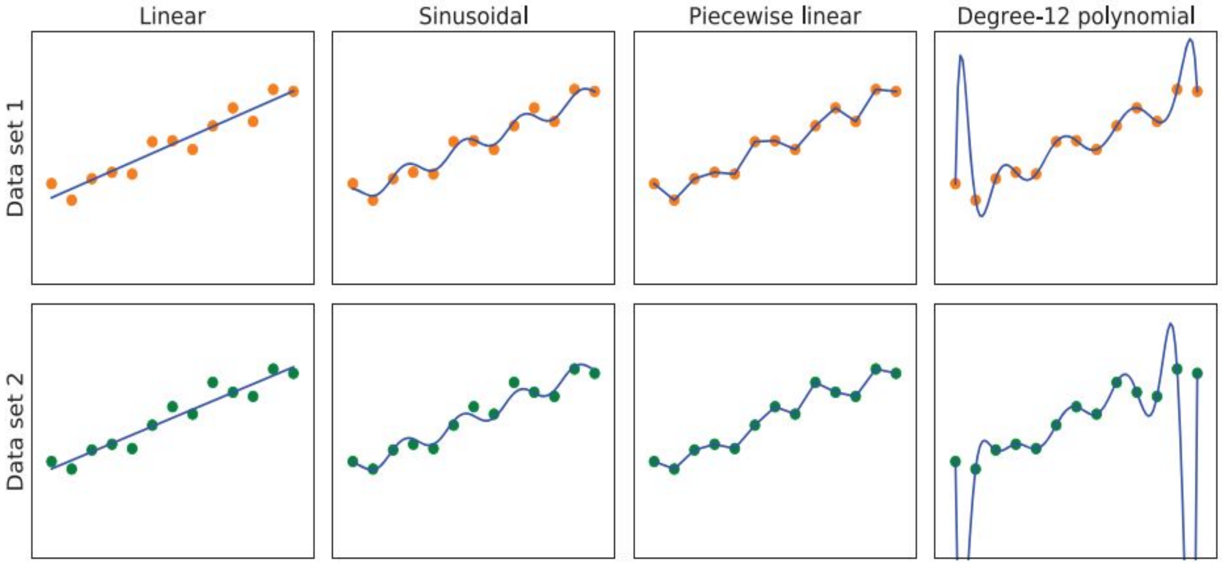


Variance

Variance: Performance on the data the model hasn't seen before

- First three columns, small difference in the data set translates into a small difference in the model. We call that low variance.
- Last column: very different functions => high variance

High variance => overfitting



Measuring Model Performance

- Often there is a bias-variance tradeoff: a choice between
  - Low-bias that fit the training data well and
  - Low-variance model that may generalize better

Which function is best?

We can't be certain. If the data is cyclic, say, the number of hits to a Website that grows from day to day, then we might favor the sinusoidal function.

Exercise

- Training error - 1%
  Testing error - 11%
  Low bias? High bias? Low variance? High variance?

- Training error - 15%
  Low bias? High bias? Low variance? High variance?

- Training error - 15%
  Testing error - 30%
  Low bias? High bias? Low variance? High variance?

Example Problem: Restaurant Waiting

Boolean classification

Output: ‘WillWait’; it is true for examples where we do wait for a table.

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry right now.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant’s price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. WaitEstimate: host’s wait estimate: 0-10, 10-30, 30-60, or >60 minutes.

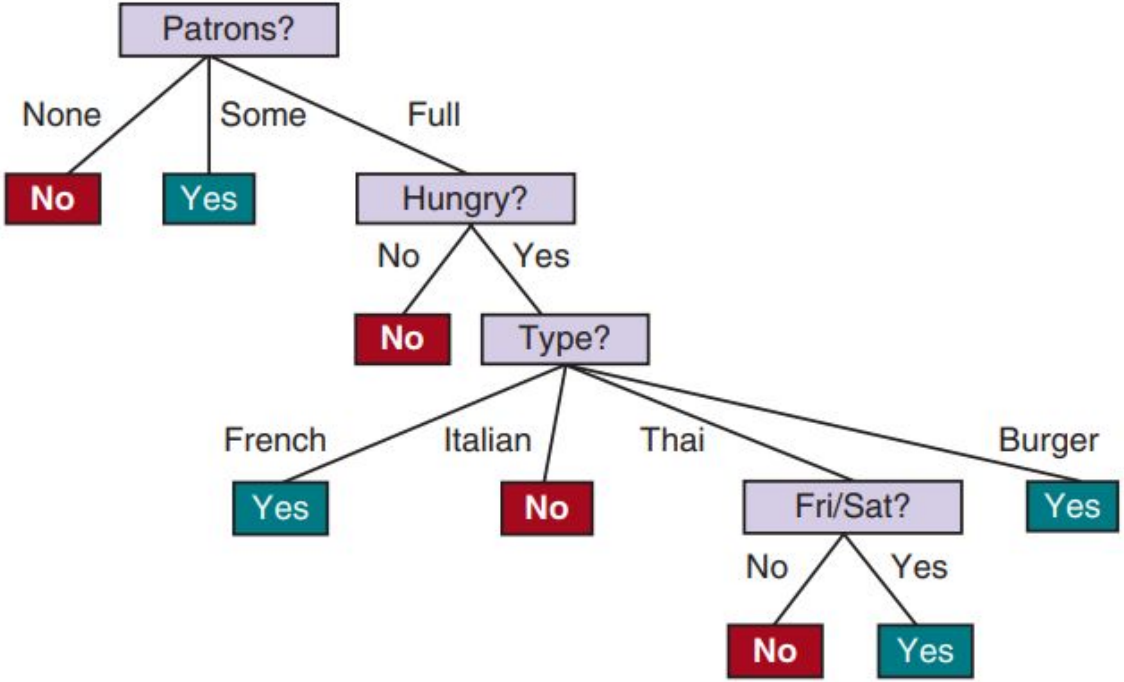
Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Decision Tree Classifier

Learning Decision Trees

Decision tree reaches its decision by performing a sequence of tests, starting at the root and following the appropriate branch until a leaf is reached.

Input and output values can be discrete or continuous.

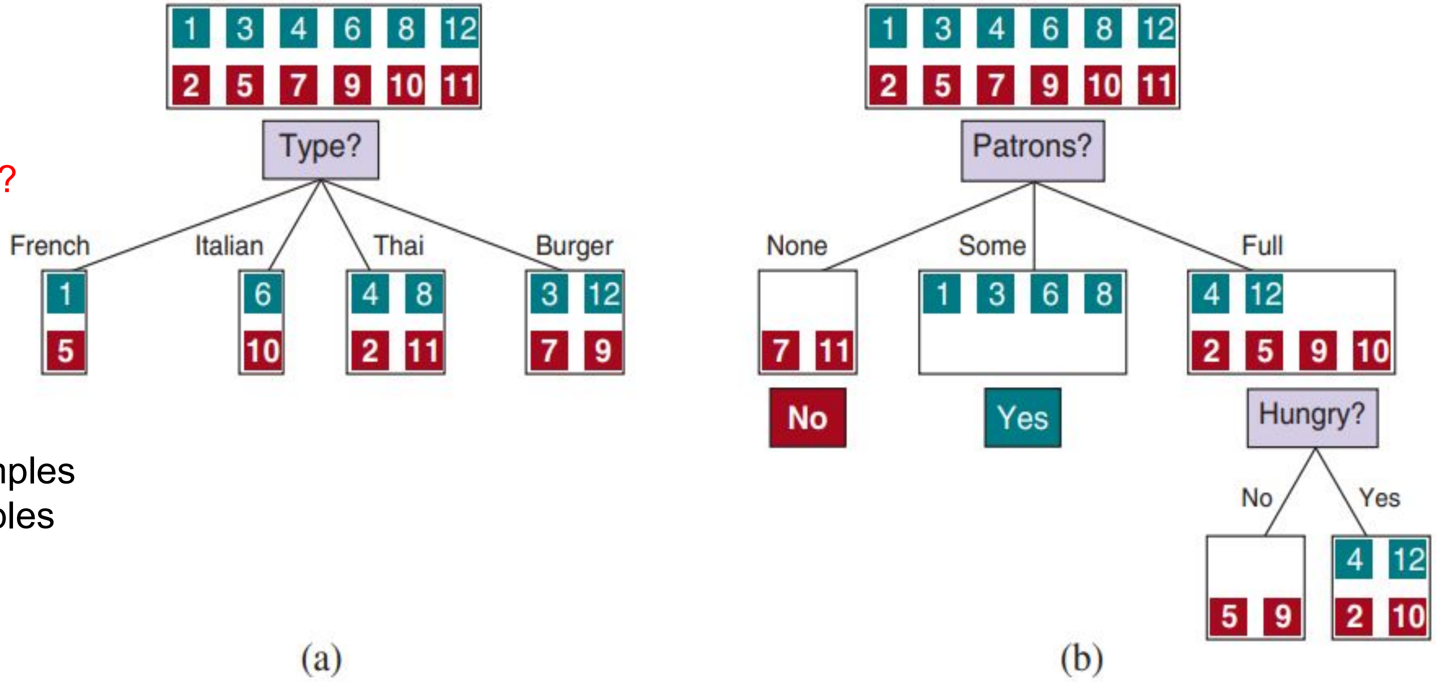


Finding an Important Attribute

Decision tree adopts a greedy divide-and-conquer strategy: always test the most important attribute first, then recursively solve the smaller subproblems.

<u>Most important attribute:</u> one that makes the most difference to the classification

Observations?
Which attribute is important?

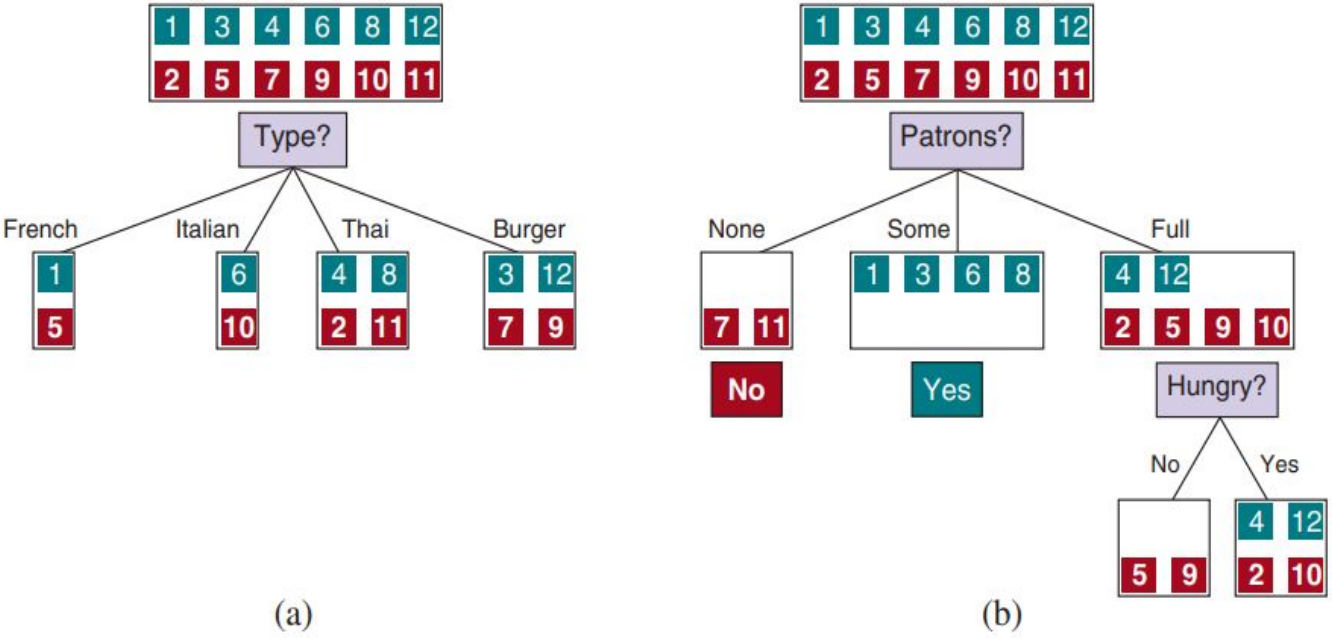


Green boxes: positive examples
Red boxes: negative examples
(a) Splitting on Type
(b) Splitting on Patrons

Finding an Important Attribute

Type is a poor attribute, because it leaves us with four possible outcomes, each of which has the same number of positive as negative examples.

Patrons is a fairly important attribute, because if the value is None or Some, then we are left with example sets for which we can answer definitively (No and Yes, respectively). If the value is Full, we are left with a mixed set of examples.



Attribute Tests

DT algorithm chooses the feature with the highest importance, measured using entropy

Entropy = measure of the uncertainty;

More information => less entropy

A coin that always comes up heads has no uncertainty => entropy = 0

Entropy

The entropy of a random variable V with values v_k having probability $P(v_k)$ is defined as:

$$\text{Entropy:} \quad H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k) .$$

Entropy of a fair coin flip:

$$H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 .$$

Entropy

Assume a coin has 99% probability of landing in heads, what is the entropy of this loaded coin?

Example: Decision Tree

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Example: Decision Tree

Step 1: Compute entropy of the entire dataset

E(source) = -[(9/14 * log2(9/14)) + (5/14 * log2(5/14))

= 0.94

Step 2: Compute entropies of outlook, temperature

Humidity, and windy

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Example: Decision Tree

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

First let's compute entropy of outlook:

Outlook	Play Tennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

Outlook	Play Tennis
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

Outlook	Play Tennis
Rainy	Yes
Rainy	Yes
Rainy	No
Rainy	Yes
Rainy	No

E(sunny) = -[(2/5 * log2(2/5)) + (3/5 * log2(3/5))] = 0.97

E(overcast) = -[(4/4 * log2(4/4))] = 0

E(rainy) = -[(3/5 * log2(3/5)) + (2/5 * log2(2/5))] = 0.97

E(outlook) = (5/14*E(sunny)) + (4/14*E(overcast)) + (5/14*E(rainy)) = 0.693

Information Gain = reduction in randomness = E(source) - E(outlook) = 0.94 - 0.693 = 0.246

Example: Decision Tree

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Similarly, compute for temperature, humidity, and windy:

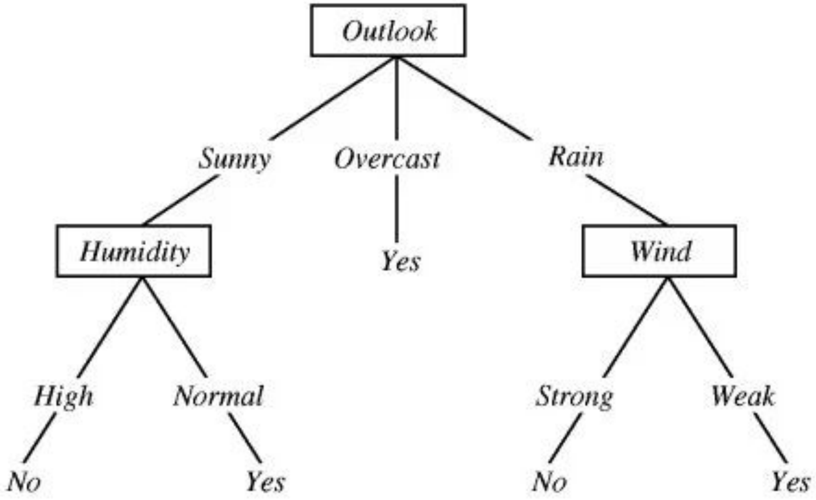
Outlook: Information Gain = 0.246

Temperature: Information Gain = 0.029

Humidity: Information Gain = 0.048

Windy: Information Gain = 0.152

Choose outlook as it has high information gain



Continuous Inputs

- For continuous attributes like Height, Weight, or Time, it may be that every example has a different attribute value.
- One way to deal with continuous values is an inequality test on the value of the feature. For example, at a given node in the tree, testing on Weight > 160 gives the most information.

Decision Trees

- In many areas of industry and commerce, decision trees are the first method tried when a classification method is to be extracted from a data set.
- Advantages with using decision trees:
    - Ease of understanding
    - Scalability to large data sets, and
    - Ability to handle discrete and continuous inputs
- Disadvantages with using decision trees:
    - If trees are very deep then getting a prediction for a new example can be expensive in run time.
    - Unstable in that adding just one new example can change the test at the root, which changes the entire tree.

Model Selection

- Simplest way is to split the examples you have into two sets: a training set to design a model, and a test set to evaluate it.
- If we are only going to create one model, then this approach is sufficient.

Often we want to compare two completely different ML models, or we might want to adjust the various “knobs” within one model. These “knobs” are called hyper-parameters.

Model Selection

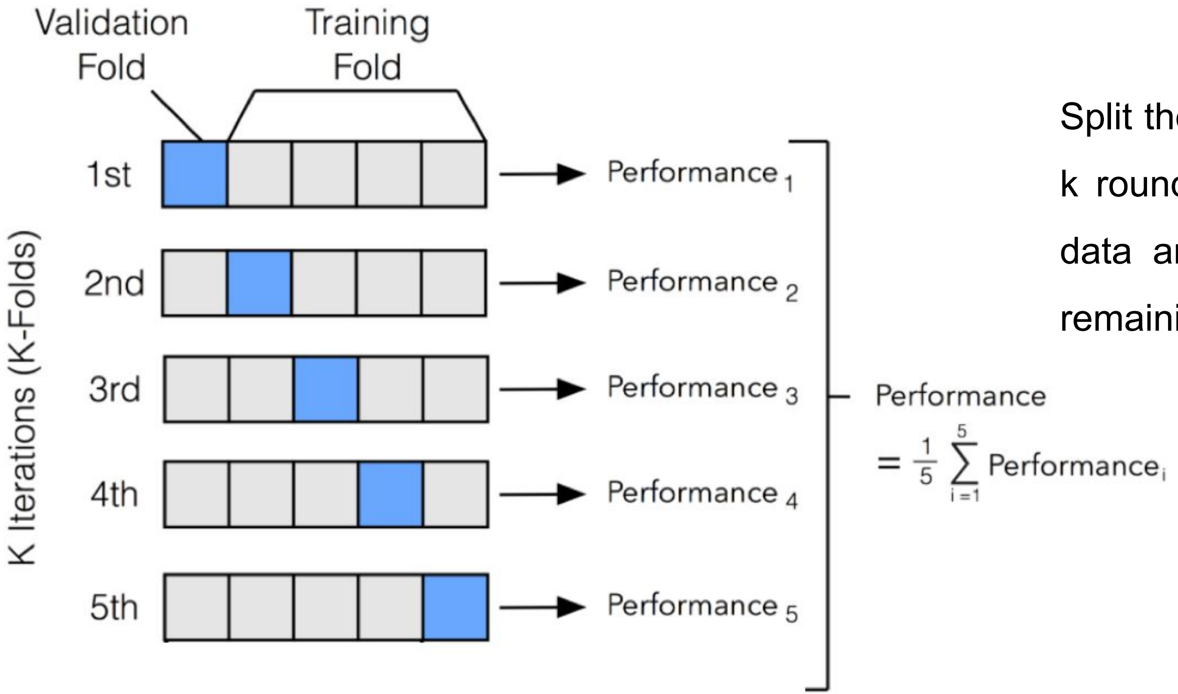
- Trying different models on the test data means we are leaking information about the test data.
- To avoid this, hold out the test set until you are completely done with training, experimenting, hyperparameter-tuning, re-training, etc.
- Three data sets:
  1. A training set to train candidate models.
  2. A validation set, also known as a development set or dev set, to evaluate the candidate models and choose the best one.
  3. A test set to do a final unbiased evaluation of the best model.

Model Selection

What if we don't have enough data to make all three of these data sets?

K-fold cross-validation

- Each example serves double duty, as training data and validation data, but not at the same time.



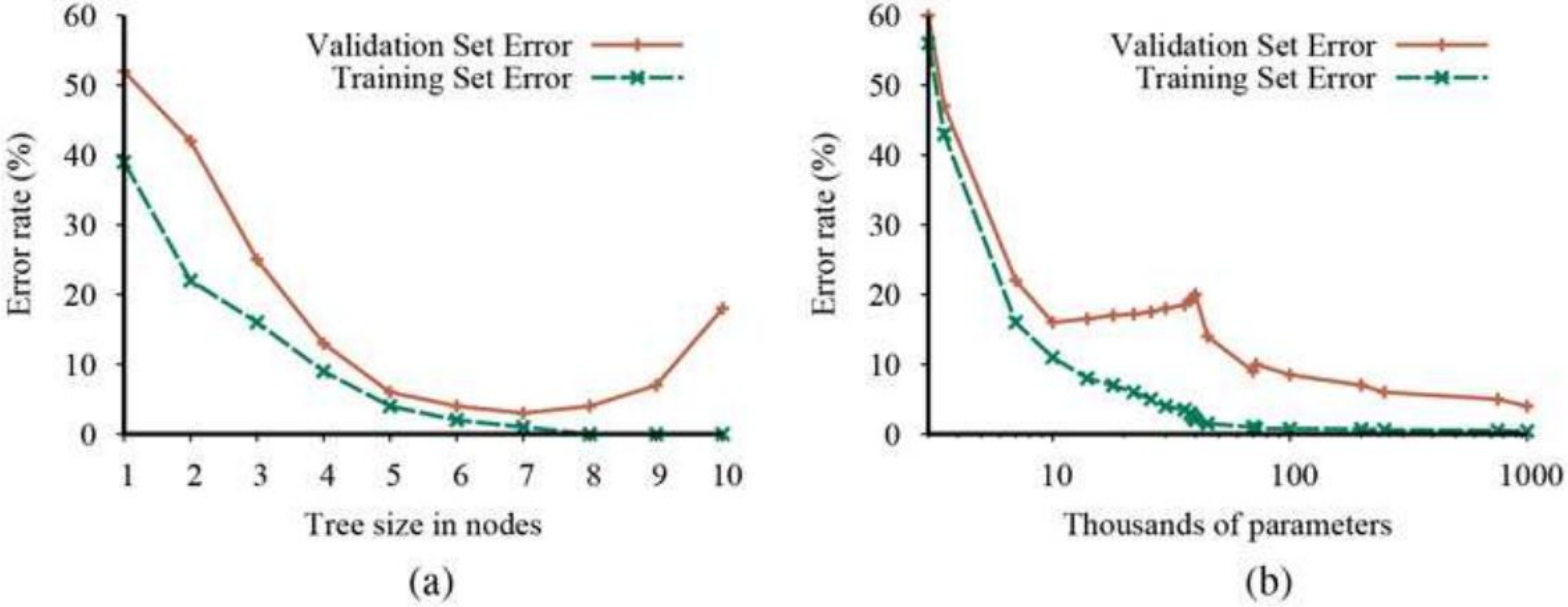
Split the data into k equal subsets. Then perform k rounds of learning; on each round 1/k of the data are held out as a validation set and the remaining examples are used as the training set.

Model Selection

Observations? Comments on the training and validation error?

Optimal number of tree size in (a)?

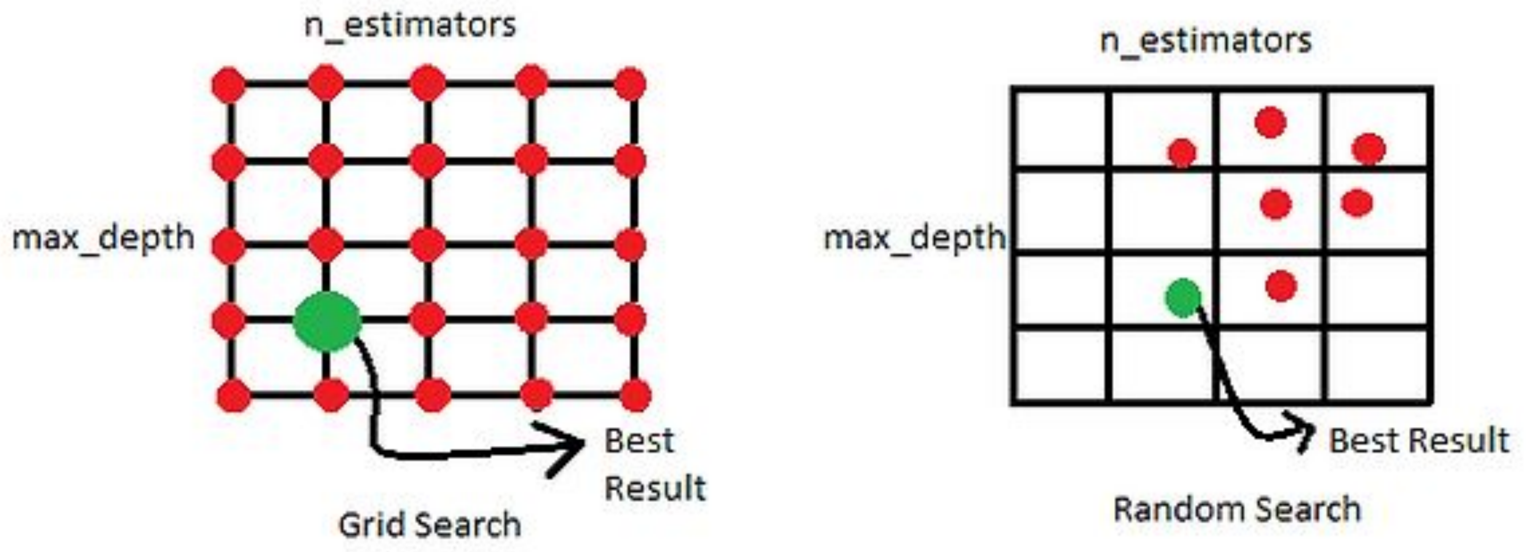
Optimal parameters in (b)?



Hyperparameter Tuning

GridSearchCV: explores all possible combinations

RandomizedSearchCV: evaluates a given number of random combinations by selecting a random value for each hyper-parameter at every iteration.



Exercise

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- You are given the dataset and asked you to choose an AI algorithm (Naive Bayes, Decision tree, KNN, SVM, logistic regression) that works well for this dataset.
- How do you go about choosing the best model?

Exercise

1. Inputs - overcast, temperature, and humidity

   outputs - play tennis or not

   Supervised learning, and it is a classification problem.
2. Split my dataset into training, testing
   a. keep my testing data aside that would help me to evaluate the model
3. Check the size of my training data
   a. If I have reasonable amount of data, then I split into training and validation
   b. If I cannot split into train and validation, then I can perform cross-validation

Exercise

4. Fit different algorithms on training set and measure the performance on the validation dataset.

Apply Naive Bayes on training set and measure performance on validation dataset

Apply Decision tree on training set and measure performance on validation dataset

Apply KNN on training set and measure performance on validation dataset

Apply SVM on training set and measure performance on validation dataset

Apply Logistic regression on training set and measure performance on validation dataset

Select the model that has high accuracy on the validation dataset

5. Train the selected model in step-4 on the training + validation dataset

6. Apply the selected model on the test dataset

Scikit-learn Introduction

Parameters vs hyperparameters

https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/

Parameter: internal to the model and can be estimated from the data

Hyperparameter: external to the model and cannot be estimated from the data

Parameters vs hyperparameters

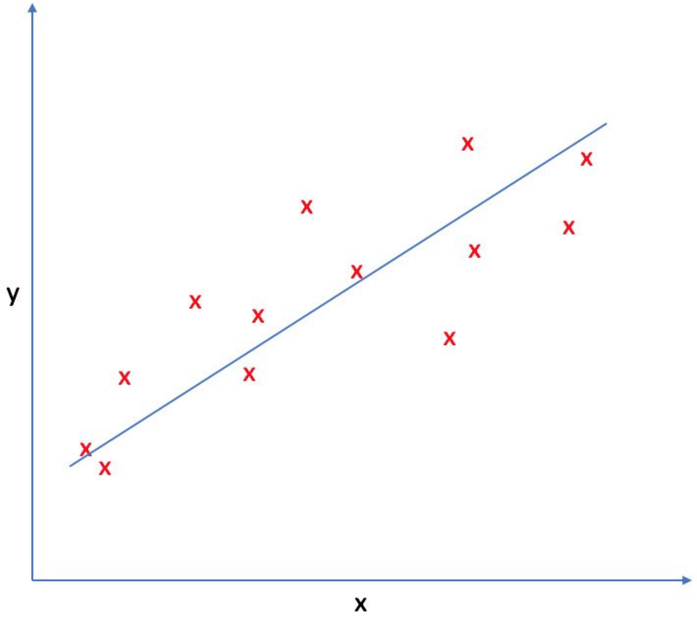
https://towardsdatascience.com/model-parameters-and-hyperparameters-in-machine-learning-what-is-the-difference-702d30970f6

Parameters: determined using training dataset. These are the fitted parameters.

Hyperparameters: adjustable parameters that must be tuned in order to obtain a model with optimal performance

Parameters vs hyperparameters: linear regression

https://www.jeremyjordan.me/linear-regression/



$$h_w(x) = w_1 x + w_0$$

Gradient descent:

$$w_i \longleftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(w)$$

Identify parameters and hyperparameters

Scikit-learn

https://scikit-learn.org/stable/modules/tree.html

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

After being fitted, the model can then be used to predict the class of samples:

```
>>> clf.predict([[2., 2.]])
array([1])
```

Identify hyperparameters in decision tree classifier

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

THANK YOU!

How do we choose a good model?

- We might have some prior knowledge about the process that generated the data.
- If not, we can perform exploratory data analysis:
  - Examining the data with statistical tests and visualizations, histograms, scatter plots, box plots to get a feel for the data.
  - Or we can just try multiple curves and evaluate which one works best.