INTRODUCTION TO MACHINE LEARNING ALGORITHMS

#DATADAWGS PRESENTATION BY JONATHAN WARING

UGA
#DATADAWGS

KAGGLE DATA SCIENCE BOWL

- We are looking for interested members who would like to participate in Kaggle's 2018 Data Science Bowl
- The contest is being sponsored by Booz Allen Hamiliton and there are cash prizes available to the top 5 placing teams (submission deadline: April 9th)
- The competition is to create an algorithm to automate nucleus detection, which could help unlock cures for disease more quickly
- Teams will create a computer model that can identify a range of nuclei across varied conditions

MACHINE LEARNING ALGORITHMS

- Representation: language for patterns/models, expressive power
- Evaluation: scoring methods for deciding what is a good fit of model to data
  - Beware of overfitting → your model may be "too good" with your training data, but may not generalize well during testing
- Search: method for enumerating patterns/models
  - Optimization techniques → most commonly used optimizer is gradient descent, which iteratively searches for the minimization of some error function (will not go into detail of how this works today)

ASSOCIATION RULE LEARNING

- Association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases
- Such information can be used as the basis for decisions about marketing activities, Web usage mining, intrusion detection, continuous production, and bioinformatics

ASSOCIATION RULE LEARNING

- The problem of association rule mining is:
  - Let I = $\{i_1, i_2, \ldots, i_n\}$ be a set of n attributes called items
  - Let D = $\{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the database
  - Each transaction D has a unique transaction ID and contains a subset of the items in I
  - A rule is defined as an implication of the form: If X then Y where X and Y are subsets of I
  - Every rule is composed by two different sets of items, also known as itemsets, X and Y, where X is called antecedent or left-hand-side (LHS) and Y consequent or right-hand-side (RHS).
- Example association rule: If A and not B, then C

SUPPORT AND CONFIDENCE

- Support is defined as the minimum percentage of transactions in the DB containing A and B.
- Confidence is defined as the minimum percentage of those transactions containing A that also contain B.
    - Ex. Suppose the DB contains 1 million transactions and that 10,000 of those transactions contain both A and B.
    - We can then say that the support of the association if A then B is:
    - S= 10,000/1,000,000 = 1%.
    - Likewise, if 50,000 of the transactions contain A and 10,000 out of those 50,000 also contain B then the association rule if A then B has a confidence 10,000/50,000 = 20%.
    - Confidence is just the conditional probability of B given A.

ASSOCIATION RULE ALGORITHMS

- Association rule algorithms typically employ some sort of method to efficiently find rules that exceed a pre-defined support or confidence level
- They do so by finding all itemsets with the given minimum support and generating rules from them!
- Let’s look at an example on the next slide

WEATHER DATA

Outlook	Temp	Humidity	Windy	Play
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

One-item sets	Two-item sets	Three-item sets	Four-item sets

Outlook = Sunny (5)	Outlook = Sunny
	Temperature = Hot (2)
		Outlook = Sunny
		Temperature = Hot
		Humidity = High (2)
			Outlook = Sunny
			Temperature = Hot
			Humidity = High
			Play = No (2)

Temperature = Cool (4)	Outlook = Sunny
	Humidity = High (3)
		Outlook = Sunny
		Humidity = High
		Windy = False (2)
			Outlook = Rainy
			Temperature = Mild
			Windy = False
			Play = Yes (2)

…	…	…	…

Total number of item sets with a minimum support of at least two instances: 12 one-item sets, 47 two-item sets, 39 three-item sets, 6 four-item sets and 0 five-item sets

GENERATING RULES FROM ITEMSET

- Once all item sets with the required minimum support have been generated, we can turn them into rules
- Example 4-item set with a support of 4 instances:

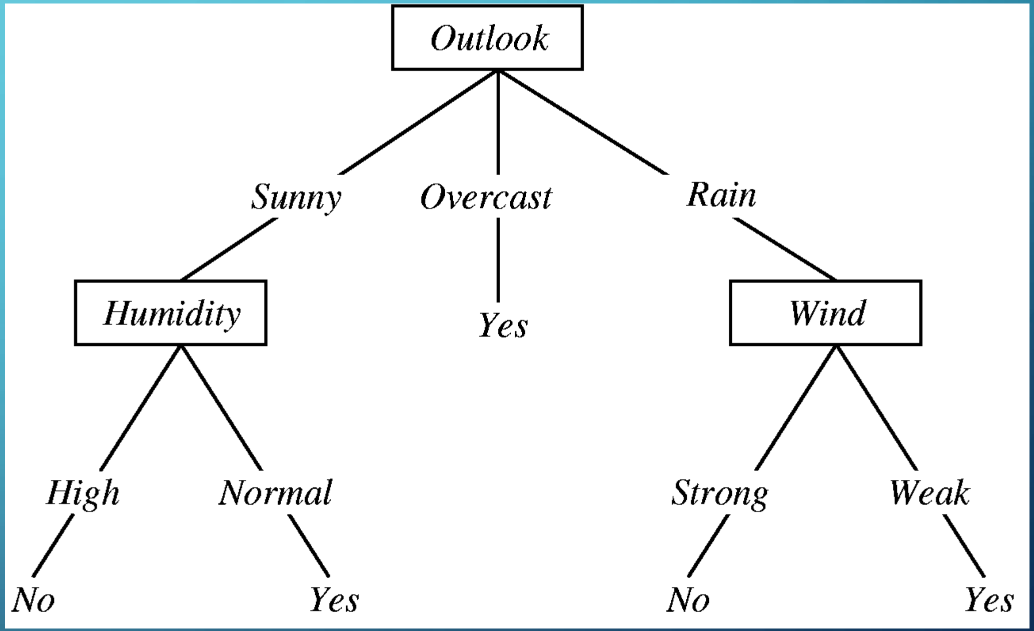
```
Humidity = Normal, Windy = False, Play = Yes (4)
```

- Seven (2^N-1) potential rules:

```
If Humidity = Normal and Windy = False then Play = Yes     4/4
If Humidity = Normal and Play = Yes then Windy = False     4/6
If Windy = False and Play = Yes then Humidity = Normal     4/6
If Humidity = Normal then Windy = False and Play = Yes     4/7
If Windy = False then Humidity = Normal and Play = Yes     4/8
If Play = Yes then Humidity = Normal and Windy = False     4/9
If True then Humidity = Normal and Windy = False
   and Play = Yes                                          4/12
```

DECISION TREE CLASSIFICATION

- A decision tree uses a tree structure to represent a number of possible decision paths and an outcome for each path
- Decision trees are very easy to understand and interpret, and also has the advantage of handling both numeric and categorical attributes (which isn't true for other algorithms will discuss)



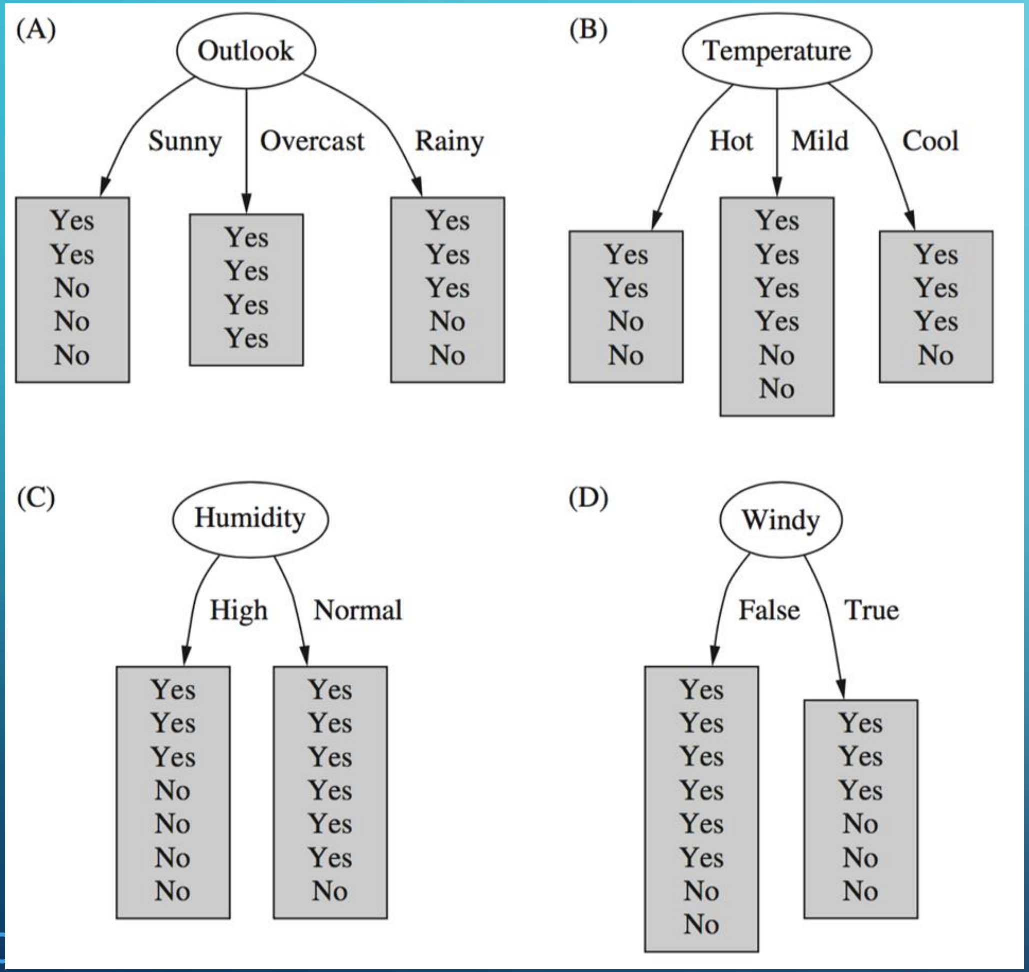
CONSTRUCTING A DECISION TREE

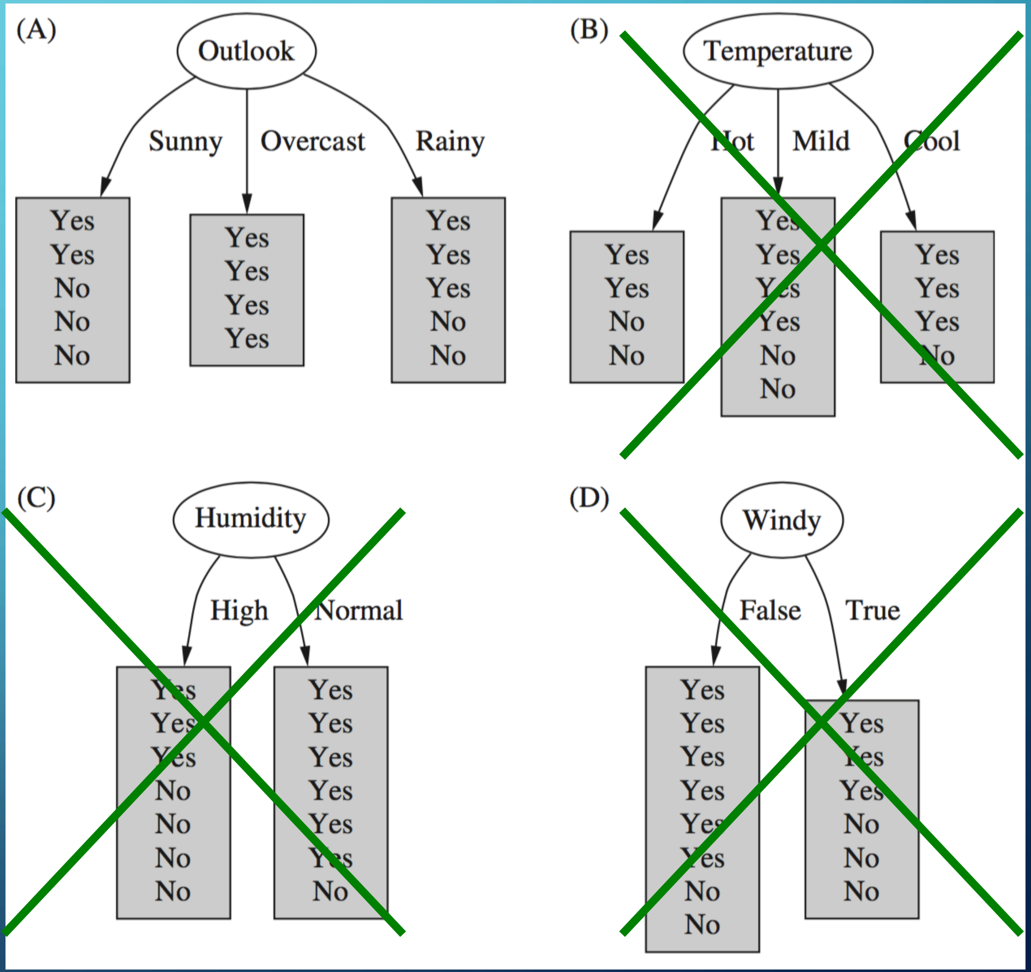
- Strategy: top down learning using recursive divide-and-conquer process
  - First: select attribute for root node

    Create branch for each possible attribute value
  - Then: split instances into subsets

    One for each branch extending from the node
  - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

WHICH ATTRIBUTE TO SELECT?





CRITERIA FOR ATTRIBUTE SELECTION

- Which is the best attribute?
  - Want to get the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Popular selection criterion: information gain
  - Information gain increases with the average purity of the subsets
- Strategy: amongst attributes available for splitting, choose attribute that gives greatest information gain
- Information gain requires measure of impurity
- Impurity measure that it uses is the entropy of the class distribution, which is a measure from information theory

COMPUTING INFORMATION

- We have a probability distribution: the class distribution in a subset of instances
- The expected information required to determine an outcome (i.e., class value), is the distribution's entropy
- Formula for computing the entropy:

$$\text{Entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

- Entropy is maximal when all classes are equally likely and minimal when one of the classes has probability 1

COMPUTING INFORMATION GAIN

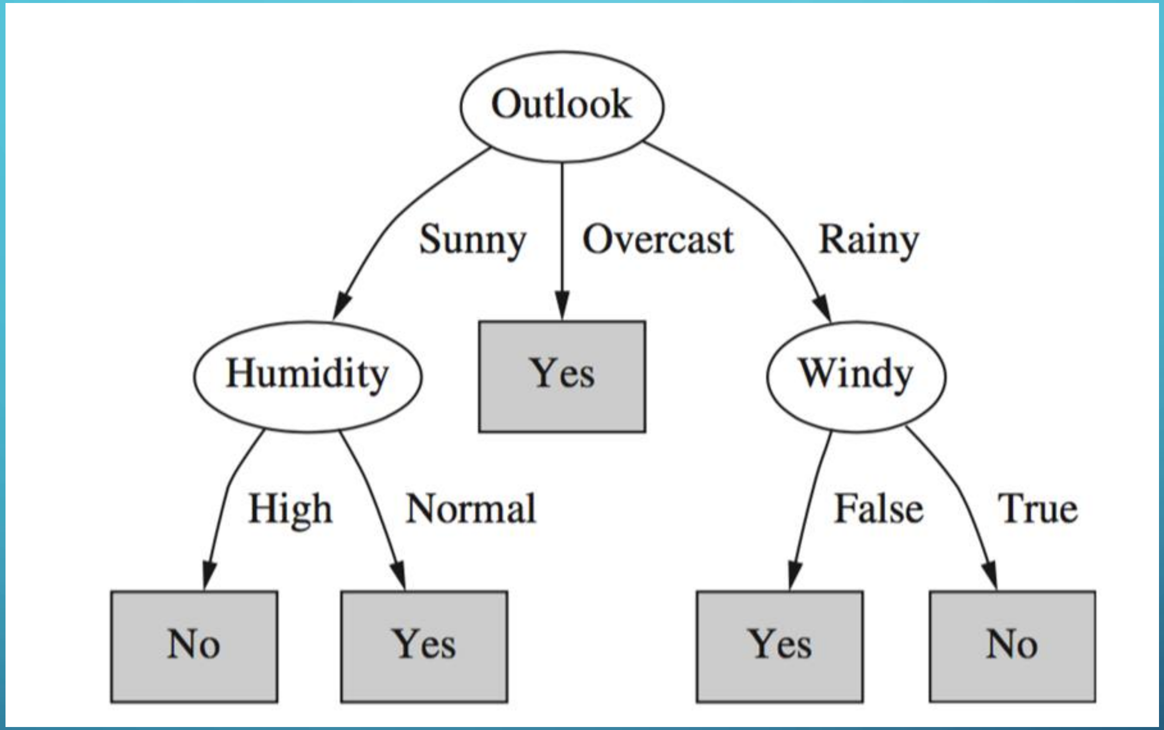
- Information gain: information before splitting – information after splitting

$$\begin{aligned} \text{Gain}(\textit{Outlook}) &= \text{Info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \text{ bits} \end{aligned}$$

- Information gain for attributes from weather data:

Gain(*Outlook*)	= 0.247 bits
Gain(*Temperature*)	= 0.029 bits
Gain(*Humidity*)	= 0.152 bits
Gain(*Windy*)	= 0.048 bits

FINAL DECISION TREE



- Note: not all leaves need to be pure; sometimes identical instances have different classes
  - Splitting stops when data cannot be split any further

RANDOM FOREST TREES

- Decision trees often can overfit the training data using the algorithm we just discussed
- Most decision tree algorithms today use a slightly different measure of node purity that attempts to combat this (it is known as the gain ratio for those wanting to look it up)
- However, random forests is another algorithm that does well at handling this problem
- It works by building multiple decision trees and lets each tree vote on how to classify inputs
- The final decision is made by the majority vote
- Again, we will not go into too much detail of this algorithm today, but it is something to know

NAÏVE BAYES CLASSIFICATION

- Naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features
- Independence assumption is almost never correct! But this scheme often works surprisingly well in practice
- Baye’s rule is stated as probability of an event, H, given evidence, E:
  - P(H | E) = P(E | H)P(H) / P(E) (we usually drop the P(E) term)
- *A priori* probability of H : P(H)
  - Probability of event *before* evidence is seen
- *A posteriori* probability of *H* : P(H | E)
  - Probability of event *after* evidence is seen
- *Likelihood* of *H* : P(E | H)

NAÏVE BAYES CLASSIFICATOIN

- Classification learning: what is the probability of the class given an instance?
  - Evidence E = instance's non-class attribute values
  - Event H = class value of instance
- Naïve assumption: evidence splits into parts (i.e., attributes) that are conditionally independent
- This means, given n attributes, we can write Bayes' rule using a product of per-attribute probabilities:

$$P(H \mid E) = P(E_1 \mid H) P(E_2 \mid H) \dots P(E_n \mid H) P(H) / P(E)$$

PROBABILITIES FOR WEATHER DATA

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Prediction of a new day

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

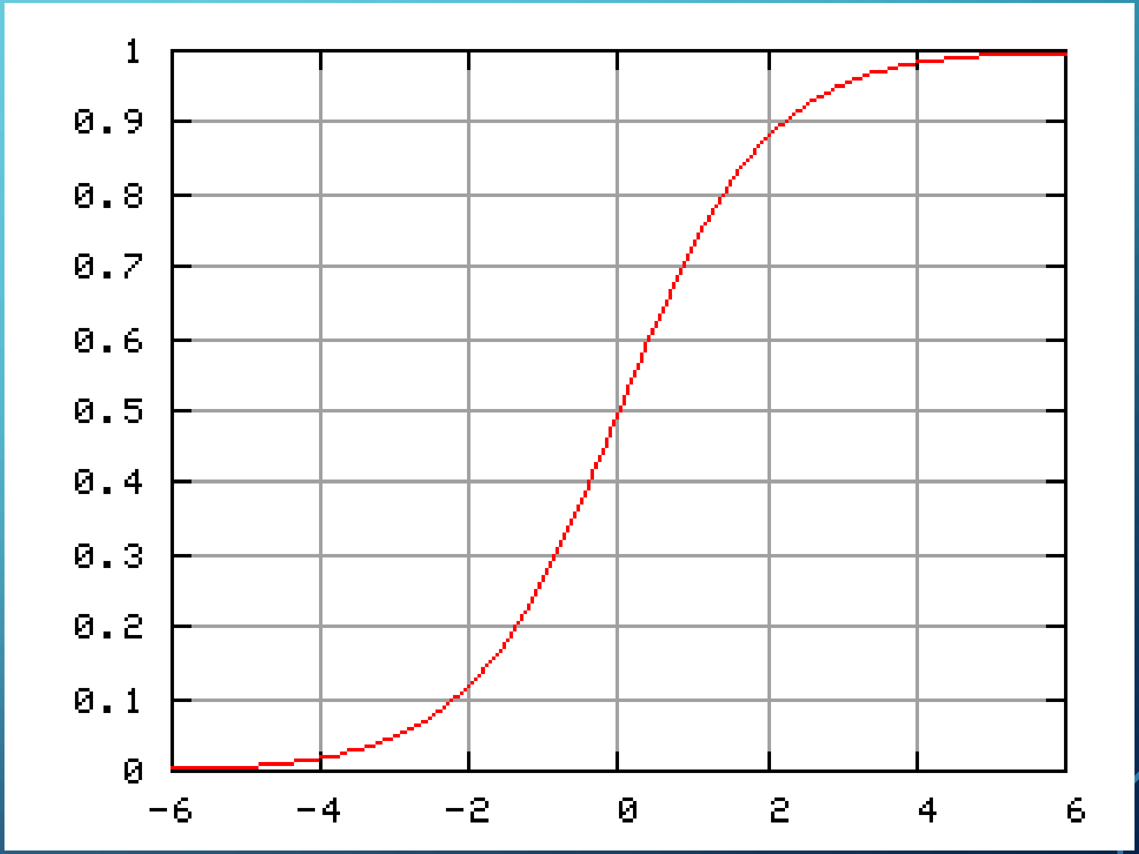
ZERO-FREQUENCY PROBLEM

- What if an attribute value does not occur with every class value? (e.g., “Humidity = high” for class “yes”)
    - Probability will be zero: $P(Humidity = High \mid yes) = 0$
    - A posteriori probability will also be zero: $P(yes \mid E) = 0$
      (Regardless of how likely the other values are!)
- Remedy: add 1 to the count for every attribute value-class combination (Laplace estimator)
- Result: probabilities will never be zero

- Note: Naïve Bayes can handle numeric attributes using Gaussian probability distribution estimates, but we will not cover that today

LOGISTIC REGRESSION FOR CLASSIFICATION

- Logistic regression is a mathematical model used in statistics to estimate (guess) the probability of an event occurring having been given some previous data
- Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0). So given some feature x it tries to find out whether some event y happens or not.
- Logistic Regression uses the logistic function to find a model that fits with the data points. The function gives an 'S' shaped curve to model the data.



ODDS

- Logistic regression uses the concept of odds ratios to calculate the probability. This is defined as the ratio of the odds of an event happening to its not happening.
  - For example, the probability of a sports team to win a certain match might be 0.75.
  - The probability for that team to lose would be 1 – 0.75 = 0.25.
  - The odds for that team winning would be 0.75/0.25 = 3. This can be said as the odds of the team winning are 3 to 1
- The odds can be defined as:
  - *Odds* = P(y = 1 | x) / 1 - P(y = 1| x)

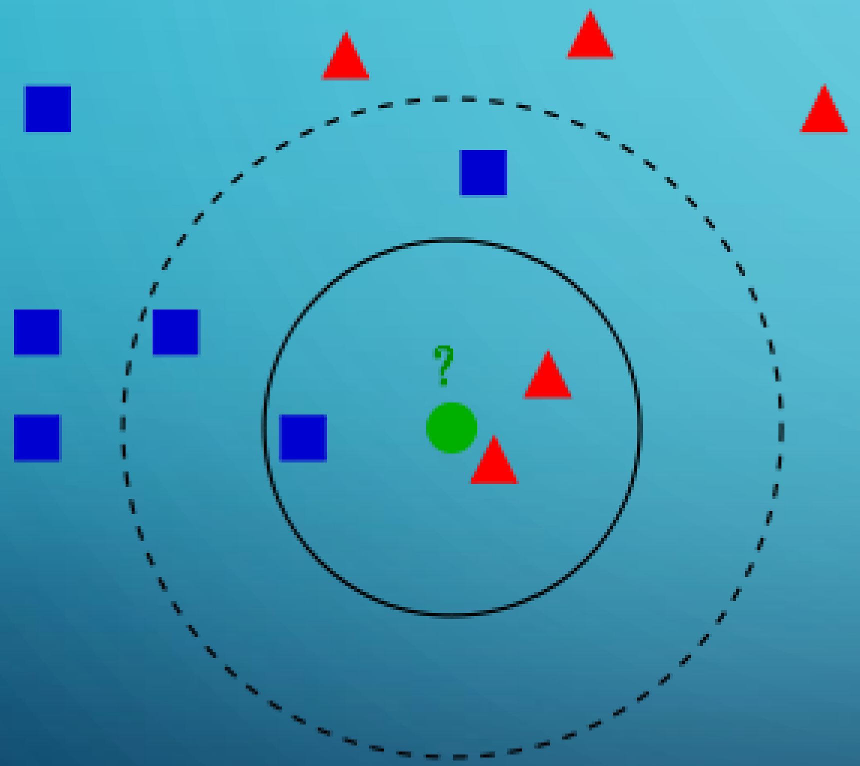
LOGIT TRANSFORMATION

- The natural logarithm of the odds ratio is then taken in order to create the logistic equation. The new equation is know as the logit:
  - $Logit(P(x)) = \ln(P(y = 1 \mid x) / 1 - P(y = 1 \mid x))$
- In Logistic regression the Logit of the probability is said to be linear with respect to x, so the logit becomes:
  - $Logit(P(x)) = a + bx$
- Using the two equations together then gives the following:
  - $P(y = 1 \mid x) / 1 - P(y = 1 \mid x) = e^{a + bx}$
- This then leads to the probability:
  - $P(Y = 1 \mid x) = e^{a + bx} / 1 + e^{a + bx} = 1 / 1 + e^{-(a + bx)}$

K-NN

- K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression
- In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:
  - In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
  - In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

K-NN EXAMPLE



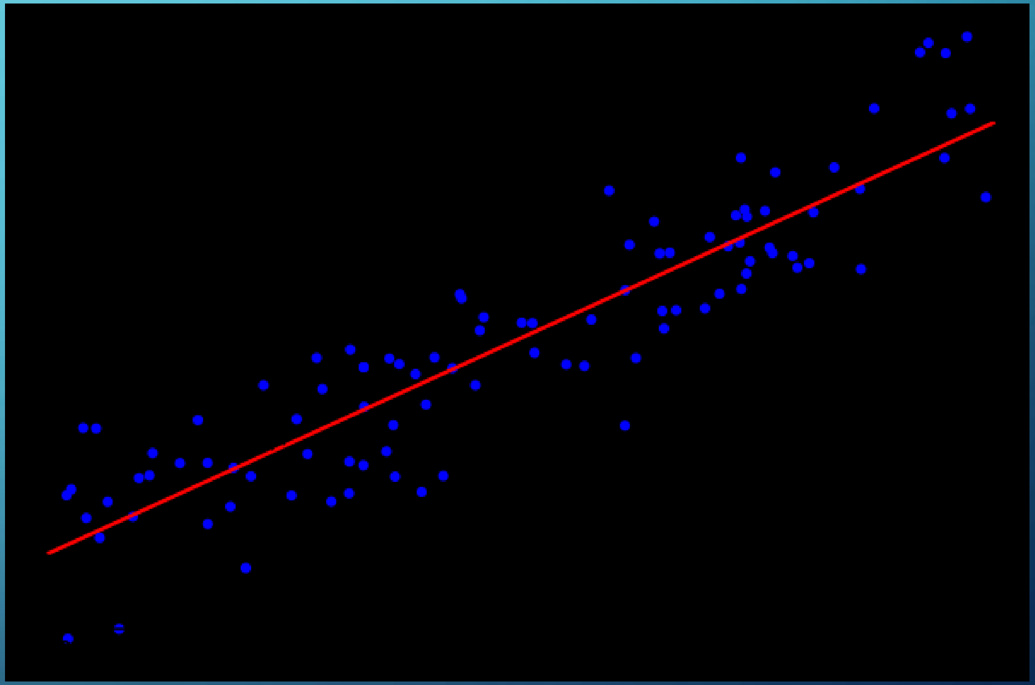
- Example of k-NN classification:
  - The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles.
  - If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.
  - If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

LINEAR REGRESSION

- Linear regression is a special case of regression analysis, which tries to explain the relationship between a dependent variable and one or more explanatory variables
- Linear regression assumes that the output variable, Y, can be expressed as a linear combination of its input variables, X
- $Y = b_0 + b_1x_1 + \ldots + b_nx_n$
  - Each input variable is given a “weight” in determining how important it is to the prediction (be careful, this is not always a correct interpretation)

LINEAR REGRESSION PICTURE

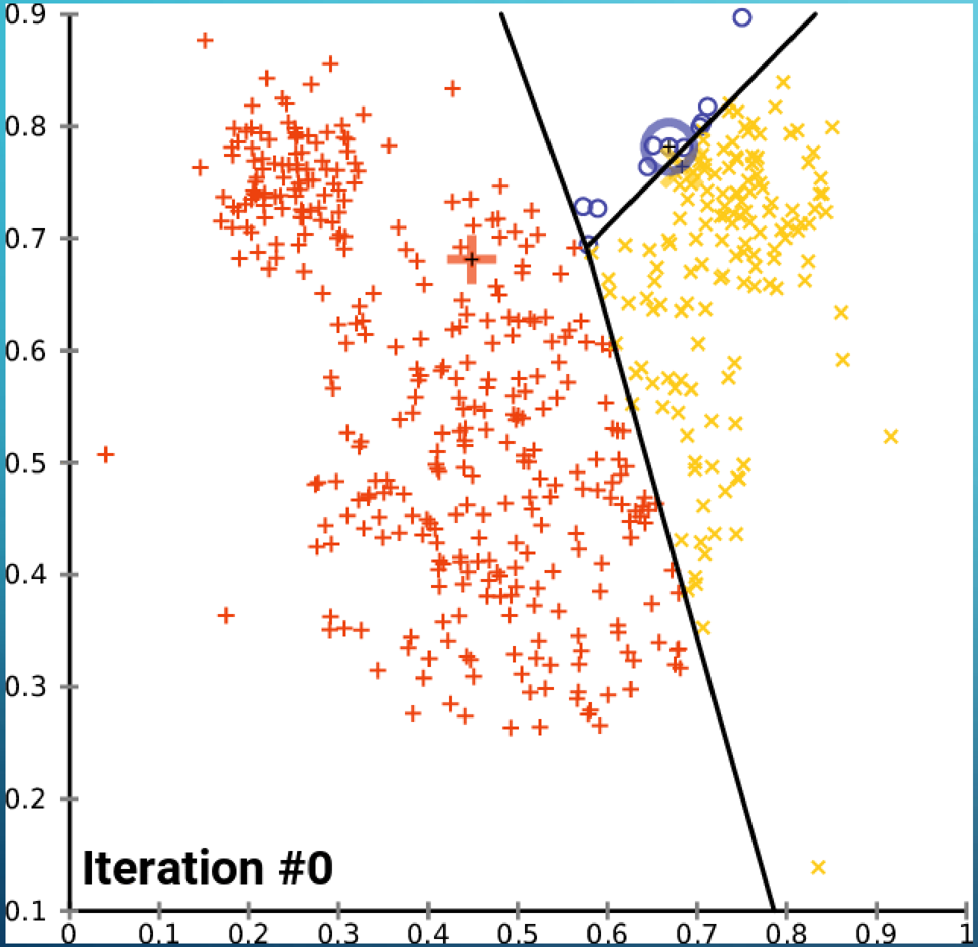
- The idea is to find the red curve, the blue points are actual samples.
- With linear regression all points can be connected using a single, straight line.
- This example uses simple linear regression, where the square of the distance between the red line and each sample point is minimized.



K-MEANS

- K-means is a clustering algorithm that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster (this prototype is called the centroid).
- This is the most commonly used unsupervised machine learning method
- It is hard to determine what to make k be, but there have been methods used overtime to best choose (often guess and check)

THE ALGORITHM



- To run a k-means algorithm, you have to randomly initialize k points to be centroids
- Iteratively repeat the following until convergence:
- Cluster assignment
  - Go through each of the data points and depending on which centroid is closer, it assigns the data points to one of the k cluster centroids.
- Move centroids
  - Calculate the average of all the points in a cluster and moves the centroid to that average location.

SKLEARN TUTORIAL

- Now we will look at using some of these machine learning algorithms using Python and the Sklearn library

REFERENCES

- Some material was taken from Dr. Rasheed's CSCI 4380 lecture material