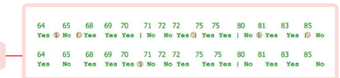
4. Algoritmi

One rule

- Classificatore estremamente semplice che costruisce una singola regola decisionale basata su un solo attributo
- La sua efficacia, sorprendentemente elevata su dataset semplici, lo rende una solida baseline per confronti
- **Meccanismo**
  - Per ciascun attributo, si costruisce una regola che associa ad ogni valore la classe più frequente
  - Si valuta l'accuratezza di ciascuna regola
  - Si sceglie l'attributo con la regola più accurata (minimo errore)
- **Assunzioni**
  - L'attributo scelto è sufficiente per una buona predizione
  - I dati sono etichettati correttamente e senza ambiguità
- **Vantaggi**
  - Semplice da implementare e spiegare
  - Robusto su dataset con forte rilevanza di un singolo attributo
  - Utile come baseline nei confronti sperimentali
- **Svantaggi**
  - Ignora tutte le interazioni tra attributi
  - Rischio elevato di overfitting con dati rumorosi o molti valori distinti (uso di numeri per descrivere una condizione)
    - **Soluzione**
      - Imporre un numero minimo di istanze (ex. 3) in una classe

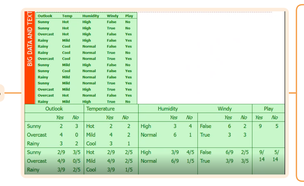


Ogni intervallo prende il valore della classe di maggioranza (ex. se ci sono da 64 a 70 gradi farenheit si gioca (3 si 2 no))

  - Prestazioni limitate su problemi complessi
- **Variazioni**
  - **Hyper-pipes**
    - Si costruisce una regola per ogni classe come congiunzione di test, una per ogni attributo
    - **Per attributi numerici**
      - Il test verifica se il valore dell'istanza si trova all'interno di un intervallo
        - Dato dal minimo e dal massimo osservati nei dati di addestramento
    - **Per gli attributi nominali**
      - Il test verifica se il valore è uno dei sottoinsiemi di valori dell'attributo
        - Sottoinsieme dato da tutti i possibili valori osservati nell'addestramento
    - La classe con il maggior numero di test corrispondenti viene predetta

Modelli statistici

- **Due assunzioni principali**
  - Gli attributi sono ugualmente importanti
  - Gli attributi sono statisticamente indipendenti
    - Conoscere il valore di un attributo non dice nulla sul valore di un altro
    - Questa assunzione non è praticamente mai portabile nella realtà
      - Ma questo schema funziona comunque bene nella pratica
- Esempio



  - Rappresentabile tutto mediante la probabilità di Bayes
    - $Pr[H \mid E] = \frac{Pr[E \mid H]Pr[H]}{Pr[E]}$
      - Dove
        - Pr [H | E] è la probabilità di un evento H data un evidenza E
        - Pr [E | H] probabilità dell'evidenza dato un certo evento
        - Pr [H] è la probabilità che ci sia un evento prima che una evidenza sia visibile
          - Ex. 9/14 e 5/14
        - P[E] è la probabilità dell'evidenza
          - Difficile da calcolare
      - Se gli eventi sono indipendenti la probabilità congiunta è il prodotto delle singole probabilità

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

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

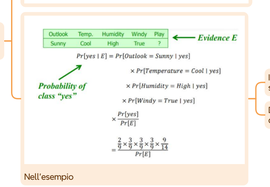
Conversion into a probability by normalization:

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

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

- **Naive Bayes**
  - Assunzione Naive su Bayes
    - L'evidenza si divide in parti (cioè attributi) che sono indipendenti
  - Dato che
    - La probabilità congiunta è il prodotto delle singole probabilità
    - Siamo nel caso in cui gli eventi sono statisticamente indipendenti
  - Allora
    - Possiamo contare la probabilità di E dato H come la sua proiezione rispetto ai singoli attributi

$$Pr[H \mid E] = \frac{Pr[E_1 \mid H]Pr[E_2 \mid H] \dots Pr[E_n \mid H]Pr[H]}{Pr[E]}$$



Nell'esempio

    - Il pezzo sotto non lo calcoliamo, ci basta sapere se è più alto il si o il no
    - Dato che le probabilità sono sullo stesso denominatore ci basta confrontare i numeratori !
  - **Cosa succede se il valore di un attributo non è presente in ogni valore della classe?**
    - Se abbiamo 0 in uno dei valori è un problemal
    - Tutto quanto va a 0 nel prodotto
      - Ex. P[Outlook = Overcast | No] = 0/5
    - Preprocessing necessario
      - Ex. Aumentiamo tutti quanti di 1
        - **Smoothing**
  - **Attenzione all'overfitting**
    - Il modello è più complesso

Decision Tree

- Modelli di classificazione supervisionata che utilizzano una strategia **divide-et-impera**
- **Costruzione**
  - Innanzitutto si seleziona un attributo per il nodo radice
  - Si crea un ramo per ogni possibile valore dell'attributo
    - Ex. Per l'attributo Outlook
      - Sunny, Overcast, Rainy
  - Poi si dividono le istanze in sottoinsiemi, uno per ogni ramo che si estende dal nodo
    - Per ogni attributo scelgo un sottoinsieme di attributi
      - Ex. Humidity o Windy
  - Infine si ripete ricorsivamente per ogni ramo, usando solo le istanze che raggiungono il ramo
  - Ci si ferma se tutte le istanze hanno la stessa classe
    - Ex. Sunny -> Humidity porta sempre a Yes

Outlook	Humidity	Giocare?
Sunny	Normal	Yes
Sunny	Normal	Yes

- Ogni nodo interno rappresenta un test su un attributo
- Ogni ramo rappresenta un esito del test
- Ogni foglia rappresenta una classe di output
- **Criteri di scelta degli attributi di divisione**
  - Voglio avere l'albero più piccolo
    - Più efficace da scorrere
  - **Euristica che seguiamo**
    - Scegliere l'attributo che produce il nodo più puro
      - Ovvero che genera quante più istanze della stessa classe
    - Criterio di purtià popolare
      - Guadagno di informazioni
      - Aumenta con la purtià media dei subset
  - **Strategia**
    - Scegliere l'attributo con il massimo guadagno di informazioni
  - **L'entropia misura la "purezza" di un insieme**
    - L'info è misurata in bit
    - $\text{entropy}(p_1, p_2, \dots, p_n) = -p_1\log p_1 - p_1\log p_2 \dots - p_n\log p_n$
      - Dove p è la frazione di elementi che soddisfano la condizione
    - L'attributo scelto per ogni nodo è quello con il massimo Information Gain
      - Quello che permette di minimizzare l'entropia residua e massimizzare l'informazione acquisita
    - L'informazione sull'attributo è dato da
      - Ex. sull'outlook
        - Outlook = Sunny
          - Entropy(2/5, 3/5) = - 2/5 log(2/5) - 3/5 log(3/5) = 0.971 bits
        - Outlook = Overcast
          - Entropy(1, 0) = -1log(1) - 0log(0) = 0 bits
        - Outlook = Rainy
          - Entropy (3/5, 2/5) = -2/5 log(2/5) - 3/5 log(3/5)
      - 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971 = 0.693
      - Probabilità sul totale / totale misurazioni * quantitativo informativo
    - Si misuta il guadagno di informazione come Entropia prima della divisione - Entropia dopo la divisione
      - Entropia (S) = - 9/14 * log(9/14) - 5/14 * log(5/14)
        - 0.940 bits
      - 0.940 - 0.693 = 0.247 bits
  - **Altra tecnica: Gain Ratio**
    - Si valuta il numero di rami che genera il nodo
      - Se sono tanti rami lo penalizziamo altrimenti no

Algoritmi di copertura

- Convertire i decision tree in un set di regole
- Ogni regola copre t istanze del dataset, di cui p appartengono alla classe target
  - Per questo si chiama approccio di copertura
- **Qualità della regola**
  - Si misura con il rapporto
    - p/t
    - Maggiore è questo valore, più la regola è considerata affidabile
- **Costruzione delle regole**
  - Le regole si costruiscono cercando condizioni che identificano sottoinsiemi "puri"
    - Puri = alta concentrazione di positivi
- **Vantaggi**
  - Ogni regola è indipendente e interpretabile
  - Possono essere più semplici da leggere rispetto agli alberi
- **Svantaggi**
  - Possibili conflitti tra regole.
  - Serve pruning per mantenere il modello semplice e preciso.
- **Esempio: PRISM**
  - Il numero totale di istanze coperte dalla regola (t)
    - Ovvero il numero di elementi a cui si applica la regola
  - L'obiettivo è massimizzare l'accuratezza della regola tenendo in considerazione
    - Gli esempi positivi della classe coperti dalla regola (p)
    - Numero di errori fatti dalla regola t-p
  - Abbiamo finito quando p/t = 1 o l'insieme delle istanze non può essere suddiviso ulteriormente

Clustering

- Le tecniche di clustering si usano quando non ci sono classi da predire
- L'obiettivo è di dividere le istanze in gruppi naturali
- Tipi di cluster
  - **Disjoint vs Overlapping**
  - **Deterministic vs Probabilistic**
  - **Flat vs Hiearchical**
- Un dataset adatto al clustering è una collezione di punti, che sono oggetti che appartengono a uno spazio
  - Uno spazio è un insieme universale di punti, dal quale sono stati estratti i punti nel dataset
- Possiamo dividere gli algoritmi di clustering in due gruppi
  - **Gerarchici**
    - O algoritmi agglomerativi
    - Ogni punto inizia come cluster separato
    - I cluster vengono poi uniti iterativamente in base alla distanza
  - **Point assignment**
    - I punti vengono assegnati al cluster tramite un criterio di ottimizzazione
- Considerazioni
  - Le tecniche gerarchiche sono più flessibili ma meno scalabili
  - Il k-means è veloce ma sensibile alla scelta iniziale e al valore di k
  - Entrambi i metodi richiedono una buona misura di distanza coerente con il dominio dei dati
- **Clustering gerarchico**
  - Richiede
    - Una metrica di distanza (spazio euclideo o altro)
    - Una regola di distanza inter-cluster (es. distanza tra centroidi)
      - Un centroide è il punto centrale (o medio) di un cluster
    - Una condizione di stop
      - Ad ex. numero desiderato di cluster o soglia di dissimilarità
  - Il risultato è rappresentato tramite un **dendrogramma**

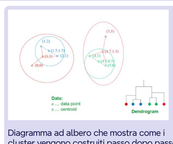


Diagramma ad albero che mostra come i cluster vengono costruiti passo dopo passo

- **K-means**
  - Algoritmo flat che richiede in input il numero k di cluster desiderati
  - **Fasi**
    - Si inizializzano casualmente k centroidi
    - Si assegna ogni punto al centroide più vicino
    - Si ricalcolano i centroidi come media dei punti assegnati
    - Si ripete finché i centroidi non cambiano più (convergenza)

Classificazione con funzioni matematiche

- Possiamo separare le istanze quasi perfettamente se introduciamo un confine che sia una linea retta, ma non perpendicolare agli assi
- È un classificatore lineare ed è una somma ponderata dei valori dei vari attributi
- **Funzioni discriminanti lineari**
  - L'equazione di una retta in due dimensioni è y = mx + b
    - m è la pendenza
    - b è l'intercetta di y
  - È chiamato discriminante lineare perché discrimina tra le classi
  - Il perimetro decisionale è una combinaizone lineare (somma pesata) degli attributi
  - Non è banale la scelta della retta migloire
    - Ce ne sono infinite che classificano potenzialmente il training set in maniera perfetta
  - **Procedura**
    - Definire una funzione obiettivo che
      - Rappresenti un obiettivo
      - Possa essere calcolata per un particolare set di pesi e un particolare set di dati
    - Crearla è complesso
      - Possibilità
        - Linear regression
        - Logistic regrassion
        - SVM
      - Ognuna usa una funzione obiettivo diversa
- **Support Vector Machines**
  - Modelli lineari che separano le classi con il massimo margine possibile
  - Costruisce una banda delimitata da due rette parallele

$$w \cdot x + b = +1 \quad \text{e} \quad w \cdot x + b = -1$$

  - La retta centrale è la funzione decisionale

$$w \cdot x + b = 0$$

  - I punti che toccano la banda sono i support vectors e determinano la posizione dell'iperpiano
  - Ovviamente avremo punti che usciranno dal lato sbagliato della banda
    - **Due casi**
      - Set linearmente separabili
        - Nessuna penalità, margine massimo
      - Set non linearmente separabili
        - L'adattamento migliore è un equilibrio tra un margine abbondante e un basso errore totale
    - Penalizziamo un punto di addestramento perché si trova sul lato sbagliato del confine di decisione
- **Logistic Regression**
  - Modello di classificazione binaria che calcola la probabilità di appartenenza alla classe positiva usando la funzione sigmoide

$$P(y = 1|\mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{dove} \quad z = w_0 + w_1x_1 + \cdots + w_nx_n$$

  - **Classificazione**
    - Un esempio viene classificato come positivo se σ(z) ≥ 0.5, altrimenti negativo
- **Linear Regression**
  - Nel modello di regressione lineare, il valore da prevedere (detto anche **score**) si ottiene calcolando una combinazione lineare degli attributi

$$\hat{z} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

    - I pesi sono calcolati in fase di addestramento
    - Questo valore ẑ è la stima finale della variabile target
  - Dobbiamo minimizzare la distanza tra questo valore e il valore ottenuto in addestramento
    - Ex. Absolute Error
      - Sottrarre l'uno dall'altro e prendere il valore assoluto
      - Si può poi minimizzare la somma degli errori assoluti o la loro media
    - La regressione lineare standard usa i quadrati di questi errori
      - **Least squares regression**
      - Penalizza errori molto grandi
      - **Inconveniente**
        - Molto sensibile ai dati
        - In punti errati o fuori norma possono alterare gravemente la funzione lineare risultante.
  - **Minimizzazione dell'errore al quadrato**

$$\sum_{i=1}^{n} (x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2$$

    - Si scelgono k+1 coefficienti che minimizzino l'errore al quadrato dai dati di addestramento
- **Tipi**
- **Instance Based Learning**
  - Gli esempi di training sono salvati da qualche parte
  - Non costruisco un modello, trovo l'esempio più vicino a me.
  - Se l'attributo è categorico è più difficile trovare il valore più vicino, se invece è numerico si può usare la distanza euclidea
  - I valori del data set vanno normalizzati
    - Differenti attributi sono misurati su scale diverse

$$a_i = \frac{v_i - min v_i}{max v_i - min v_i}$$

      - v_i : the actual value of attribute i
      - Nominal attributes: distance either 0 or 1
      - Usual policy for missing values
        - assumed to be maximally distant (given normalized attributes)
  - **k- Nearest Neighbors**
    - Approccio k-NN
      - Per classificare una nuova istanza, si considerano i k esempi più vicini nel dataset e si assegna la classe più frequente tra questi.
      - **Funzionamento**
        - Si calcola la distanza (tipicamente euclidea) tra la nuova istanza e tutte le istanze del training set
        - Si selezionano i k esempi più vicini
        - Si applica una funzione di combinazione (es. voto di maggioranza per classificazione, media per regressione)
      - **Pesatura dei vicini**

$$\text{peso} = \frac{1}{\text{distanza}^2}$$

      - **Quanti vicini (k) usare?**
        - Il nome k-NN dipende dal numero di k
          - k=3 -> 3-NN
        - **Numero dispari**
          - Convenienti per evitare vogti di maggioranza in classificazioni in due classi
        - Maggiore è k
          - **Maggiore sono i vicini con cui confrontarsi**
        - Se aumentiamo k (=n) al massimo possibile, si usa l'intero dataset per ogni predizione
          - Per classificazione
            - Si predice sempre la classe maggioritaria nell'intero set di dati
          - Per regressione
            - La media di tutti i valori target
      - **Overfitting**
        - Due casi
          - k = n
            - Il modello n-NN predice sempre il valore medio del dataset per ogni caso
            - Nessuna complessità
          - k= 1
            - Modello esremamente complesso
            - Pone confini complicati
            - Ogni esempio si troverà in una regione etichettata dalla propria classe
        - K alto
          - Basso overfitting