Università Degli Studi Di Milano

Metodi di Ensemble Gerarchici per la Predizione Strutturata della Funzione delle Proteine

Relatore

Prof. Giorgio Valentini

Correlatore

Dr. Marco Notaro

Candidato

Marco Odore

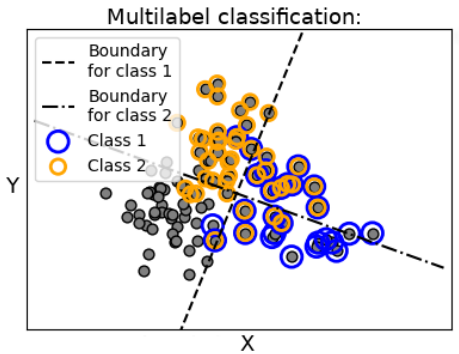
10 Luglio 2018

Il problema della predizione della funzione delle proteine

- Identificare la funzione delle proteine attraverso le analisi di laboratorio è **costosa** e richiede **molto tempo**

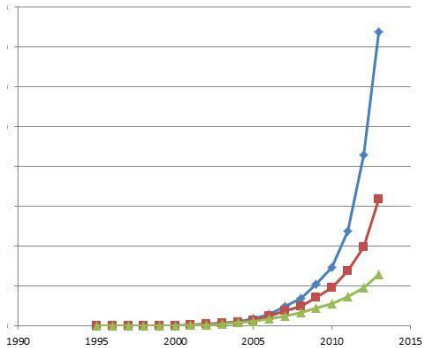
Il problema della predizione della funzione delle proteine

- Identificare la funzione delle proteine attraverso le analisi di laboratorio è **costosa** e richiede **molto tempo**
- Esistono centinaia di funzioni a cui poter associare un gene/proteina, anche contemporaneamente **(problema multiclasse e multietichetta)**



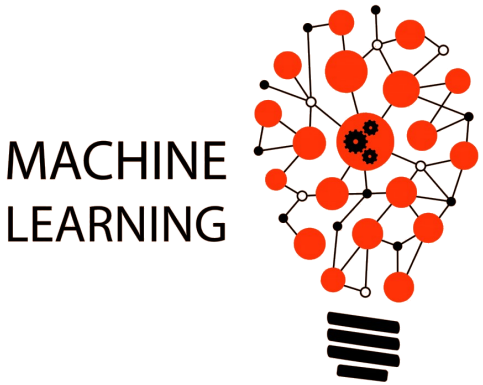
Il problema della predizione della funzione delle proteine

- Identificare la funzione delle proteine attraverso le analisi di laboratorio è **costosa** e richiede **molto tempo**
- Esistono centinaia di funzioni a cui poter associare un gene/proteina, anche contemporaneamente **(problema multiclasse e multietichetta)**
- Il quantitativo di dati genomici cresce molto rapidamente.



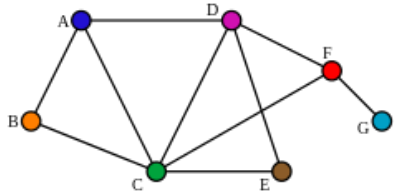
Il problema della predizione della funzione delle proteine

- Identificare la funzione delle proteine attraverso le analisi di laboratorio è **costosa** e richiede **molto tempo**
- Esistono centinaia di funzioni a cui poter associare un gene/proteina, anche contemporaneamente **(problema multiclasse e multietichetta)**
- Il quantitativo di dati genomici cresce molto rapidamente.
- La **classificazione manuale** delle proteine è quindi infattibile. È necessario quindi un approccio **automatico**.



Il problema della predizione della funzione delle proteine

- Identificare la funzione delle proteine attraverso le analisi di laboratorio è **costosa** e richiede **molto tempo**
- Esistono centinaia di funzioni a cui poter associare un gene/proteina, anche contemporaneamente **(problema multiclasse e multietichetta)**
- Il quantitativo di dati genomici cresce molto rapidamente.
- La **classificazione manuale** delle proteine è quindi infattibile. È necessario quindi un approccio **automatico**.
- A complicare ulteriormente il problema è il modo in cui sono *relazionate* tra loro le funzioni delle proteine.



Tassonomie per le funzioni delle proteine

- Esistono infatti due tassonomie principali per l'organizzazione delle funzioni:

Tassonomie per le funzioni delle proteine

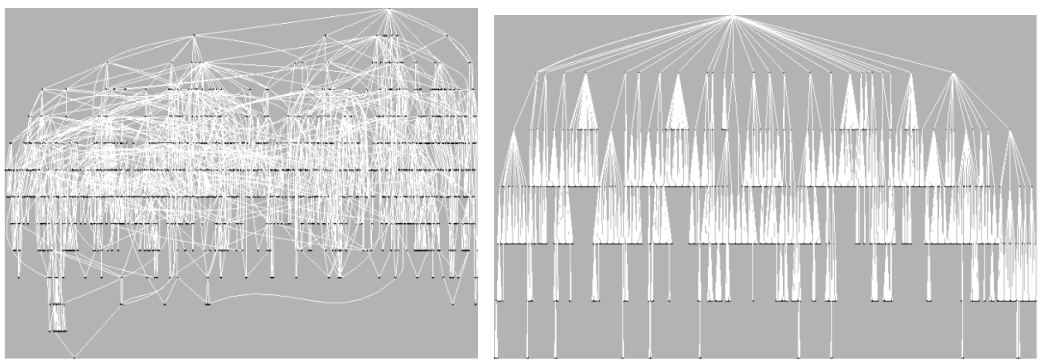
- Esistono infatti due tassonomie principali per l'organizzazione delle funzioni:
  - **Gene Ontology** (GO): che organizza le funzioni come un grafo diretto aciclico (DAG), varia per ogni specie, e possiede tre ontologie differenti (e quindi 3 DAG), e cioè *Biological Process* (BP), *Molecular Function* (MF) e *Cellular Component* (CC).

Tassonomie per le funzioni delle proteine

- Esistono infatti due tassonomie principali per l'organizzazione delle funzioni:
  - **Gene Ontology** (GO): che organizza le funzioni come un grafo diretto aciclico (DAG), varia per ogni specie, e possiede tre ontologie differenti (e quindi 3 DAG), e cioè *Biological Process* (BP), *Molecular Function* (MF) e *Cellular Component* (CC).
  - **Functional Catalogue** (FunCat): che è organizzato invece come un albero, non varia in base alle specie, e descrive le funzioni in maniera più sintetica rispetto alla Gene Ontology.

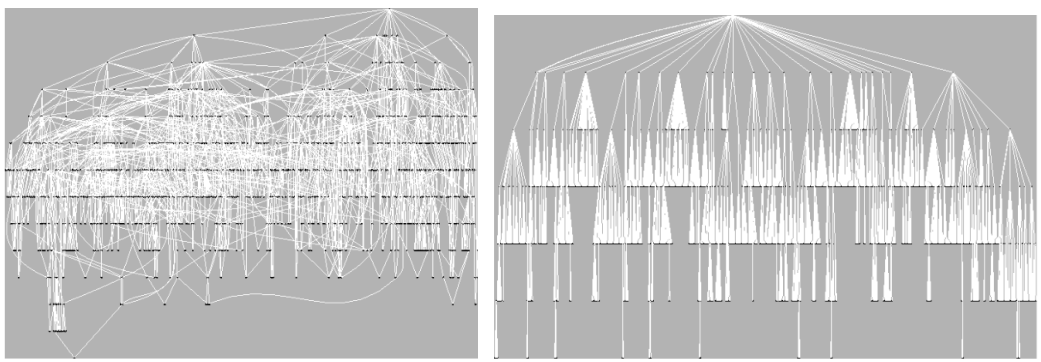
Tassonomie per le funzioni delle proteine

- Esistono infatti due tassonomie principali per l'organizzazione delle funzioni:
  - **Gene Ontology** (GO): che organizza le funzioni come un grafo diretto aciclico (DAG), varia per ogni specie, e possiede tre ontologie differenti (e quindi 3 DAG), e cioè *Biological Process* (BP), *Molecular Function* (MF) e *Cellular Component* (CC).
  - **Functional Catalogue** (FunCat): che è organizzato invece come un albero, non varia in base alle specie, e descrive le funzioni in maniera più sintetica rispetto alla Gene Ontology.



Tassonomie per le funzioni delle proteine

- Esistono infatti due tassonomie principali per l'organizzazione delle funzioni:
  - **Gene Ontology** (GO): che organizza le funzioni come un grafo diretto aciclico (DAG), varia per ogni specie, e possiede tre ontologie differenti (e quindi 3 DAG), e cioè *Biological Process* (BP), *Molecular Function* (MF) e *Cellular Component* (CC).
  - **Functional Catalogue** (FunCat): che è organizzato invece come un albero, non varia in base alle specie, e descrive le funzioni in maniera più sintetica rispetto alla Gene Ontology.



- Data la granularità e specificità superiori della GO e il suo largo utilizzo nella comunità scientifica, all'interno della tesi ci si è soffermati sulla predizione delle sue funzioni.

La predizione della funzione delle proteine tramite metodi automatici

I metodi più noti in letteratura per effettuare predizioni della funzione delle proteine in maniera automatica sono:

La predizione della funzione delle proteine tramite metodi automatici

I metodi più noti in letteratura per effettuare predizioni della funzione delle proteine in maniera automatica sono:

- I metodi basati sulla **comparazione di biosequenze**: si basano sull'idea che sequenze simili condividano funzioni simili.

La predizione della funzione delle proteine tramite metodi automatici

I metodi più noti in letteratura per effettuare predizioni della funzione delle proteine in maniera automatica sono:

- I metodi basati sulla **comparazione di biosequenze**: si basano sull'idea che sequenze simili condividano funzioni simili.
- I metodi **basati su reti**: sono metodi applicati a dati rappresentati sotto forma di reti, che si basano sugli algoritmi di propagazione delle etichette.

La predizione della funzione delle proteine tramite metodi automatici

I metodi più noti in letteratura per effettuare predizioni della funzione delle proteine in maniera automatica sono:

- I metodi basati sulla **comparazione di biosequenze**: si basano sull'idea che sequenze simili condividano funzioni simili.
- I metodi **basati su reti**: sono metodi applicati a dati rappresentati sotto forma di reti, che si basano sugli algoritmi di propagazione delle etichette.
- I metodi **Kernel per spazi di output strutturato**: sono metodi che sfruttano funzioni kernel congiunte per predire in spazi di output strutturato.

La predizione della funzione delle proteine tramite metodi automatici

I metodi più noti in letteratura per effettuare predizioni della funzione delle proteine in maniera automatica sono:

- I metodi basati sulla **comparazione di biosequenze**: si basano sull'idea che sequenze simili condividano funzioni simili.
- I metodi **basati su reti**: sono metodi applicati a dati rappresentati sotto forma di reti, che si basano sugli algoritmi di propagazione delle etichette.
- I metodi **Kernel per spazi di output strutturato**: sono metodi che sfruttano funzioni kernel congiunte per predire in spazi di output strutturato.
- I metodi **Ensemble Gerarchici**: i metodi trattati in questa tesi.

Metodi Ensemble Gerarchici 1/2

I Metodi di Ensemble Gerarchici sono metodi caratterizzati da due step principali:

Metodi Ensemble Gerarchici 1/2

I Metodi di Ensemble Gerarchici sono metodi caratterizzati da due step principali:

1. **Predizione flat** delle diverse classi dell'ontologia, generando diversi predittori *indipendenti*.

Metodi Ensemble Gerarchici 1/2

I Metodi di Ensemble Gerarchici sono metodi caratterizzati da due step principali:

1. **Predizione flat** delle diverse classi dell'ontologia, generando diversi predittori *indipendenti*.
2. **Combinazione e correzione gerarchica delle predizioni** sfruttando il DAG dei termini della GO.

Metodi Ensemble Gerarchici 1/2

I Metodi di Ensemble Gerarchici sono metodi caratterizzati da due step principali:

1. **Predizione flat** delle diverse classi dell'ontologia, generando diversi predittori *indipendenti*.
2. **Combinazione e correzione gerarchica delle predizioni** sfruttando il DAG dei termini della GO.

Il secondo step rappresenta la componente *ensemble* del metodo. Tale step si rende necessario in quanto le predizioni flat non tengono in considerazione la struttura gerarchica dei DAG della GO, portando a risultati *inconsistenti*.

Metodi Ensemble Gerarchici 1/2

I Metodi di Ensemble Gerarchici sono metodi caratterizzati da due step principali:

1. **Predizione flat** delle diverse classi dell'ontologia, generando diversi predittori *indipendenti*.
2. **Combinazione e correzione gerarchica delle predizioni** sfruttando il DAG dei termini della GO.

Il secondo step rappresenta la componente *ensemble* del metodo. Tale step si rende necessario in quanto le predizioni flat non tengono in considerazione la struttura gerarchica dei DAG della GO, portando a risultati *inconsistenti*.

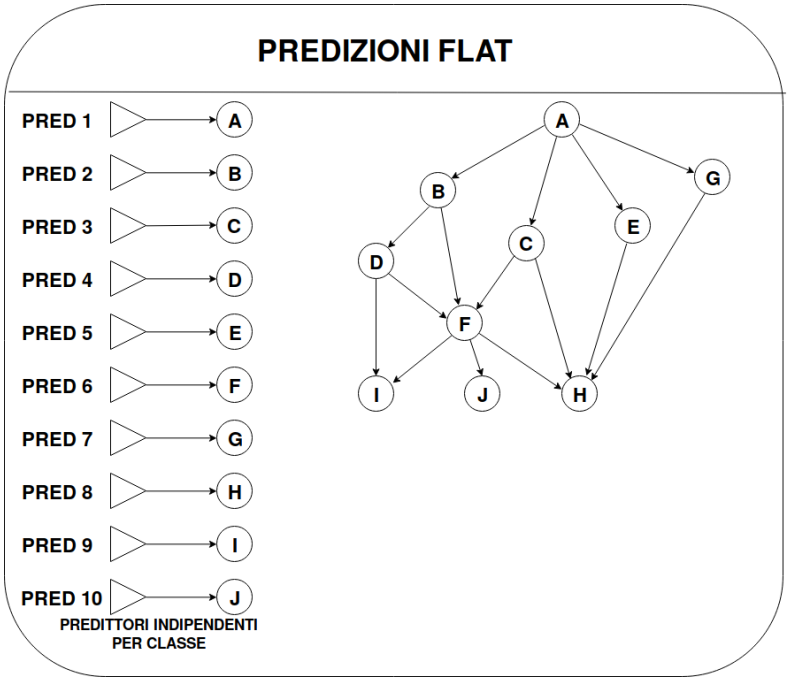
Consistenza & True Path Rule

Un insieme di predizioni $\hat{y} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|N|} >$, dove $|N|$ è la cardinalità dei termini della gerarchia, è definito *consistente*, se rispetta la *True Path Rule*, e cioè:

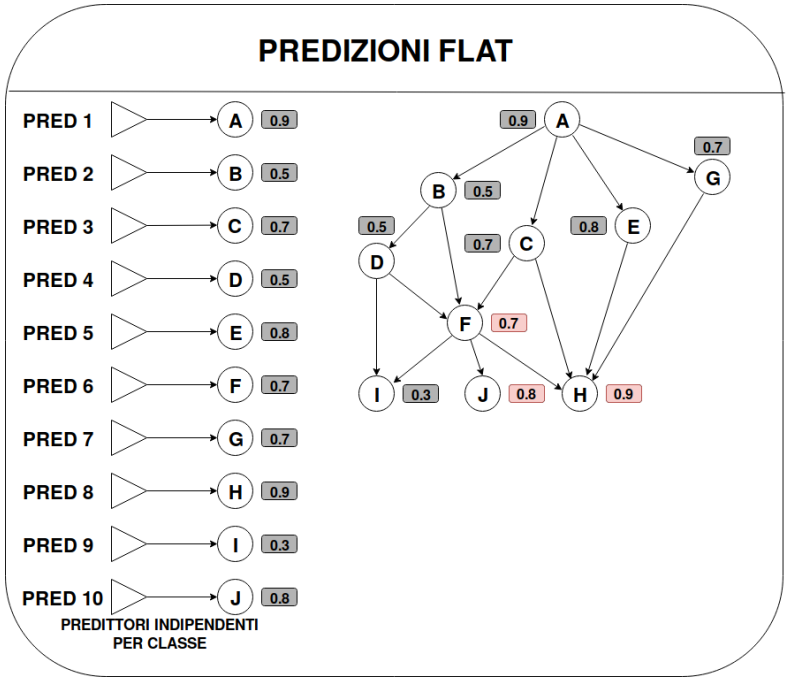
$$y \quad consistente \quad \leftrightarrow \forall i \in N, j \in par(i) \rightarrow y_j \geq y_i$$

Dove $par(i)$ indica l'insieme dei termini genitori del nodo i nella gerarchia.

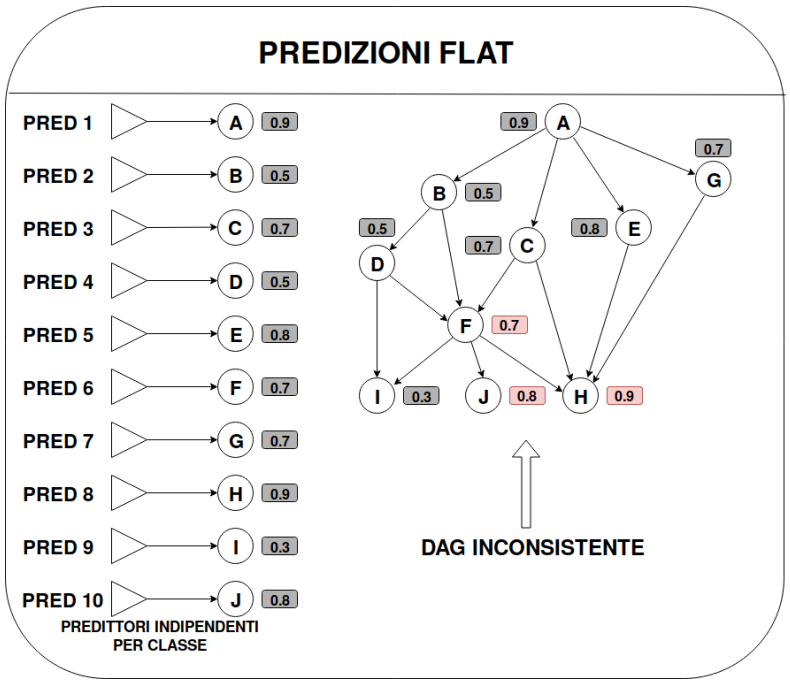
Metodi Ensemble Gerarchici (Esempio) 2/2



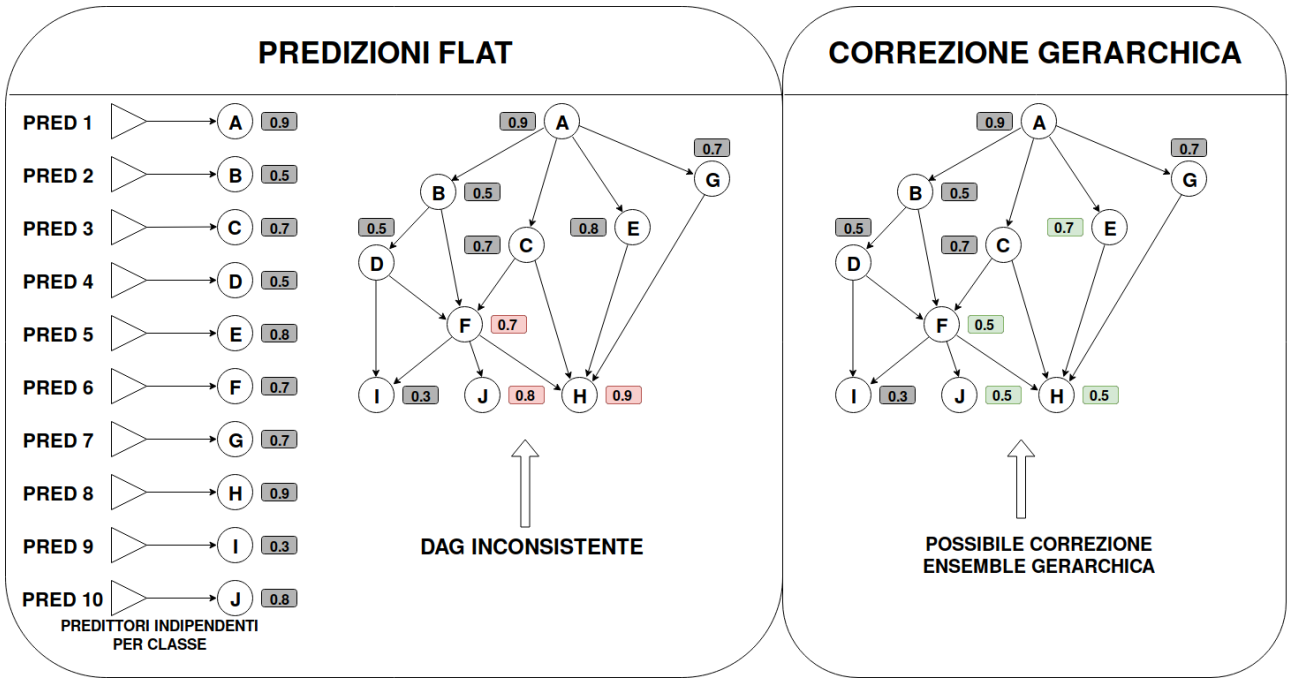
Metodi Ensemble Gerarchici (Esempio) 2/2



Metodi Ensemble Gerarchici (Esempio) 2/2



Metodi Ensemble Gerarchici (Esempio) 2/2



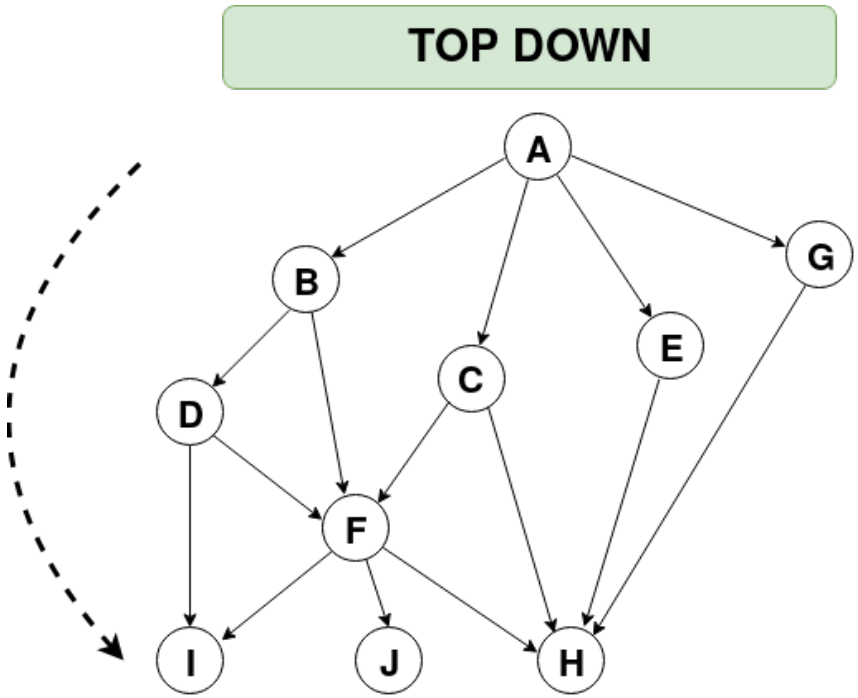
Metodi Ensemble Gerarchici: Approcci

Esistono fondamentalmente due approcci per la correzione:

Metodi Ensemble Gerarchici: Approcci

Esistono fondamentalmente due approcci per la correzione:

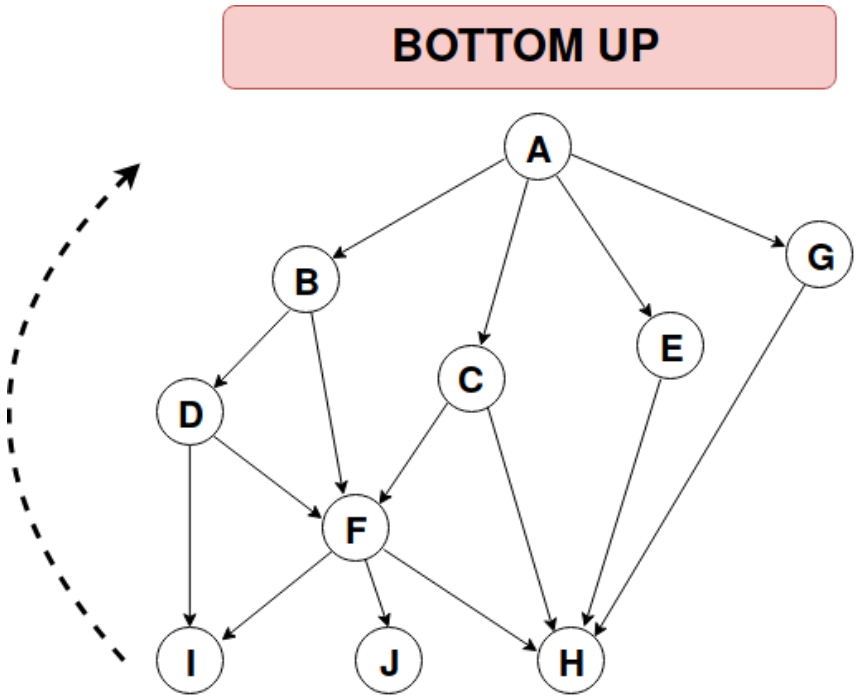
- *Top-down*: le predizioni vengono corrette dai nodi più generali a quelli più specifici.



Metodi Ensemble Gerarchici: Approcci

Esistono fondamentalmente due approcci per la correzione:

- *Top-down*: le predizioni vengono corrette dai nodi più generali a quelli più specifici.
- *Bottom-up*: Le predizioni vengono corrette dai nodi più specifici verso quelli più generali.



Metodo Top-Down Gerarchico (HTD-DAG)

- È un metodo che utilizza l'approccio *Top-Down*.

[1] **Dove il livello è quello del cammino massimo dalla radice**

Metodo Top-Down Gerarchico (HTD-DAG)

- È un metodo che utilizza l'approccio *Top-Down*.
- La correzione avviene ricorsivamente, percorrendo il grafo per *livelli*[1]. Più precisamente, dato il grafo $G = (N, E)$, gli score flat $f(x) = \hat{y}$ sono corretti gerarchicamente a $\bar{y}$, applicando la seguente regola:

Aggiornamento con HTD-DAG

$$\bar{y}_i := \begin{cases} \hat{y}_i & \text{if } \ i \in root(G) \\ min_{j \in par(i)} \bar{y}_j & \text{if } \ min_{j \in par(i)} \bar{y}_j < \hat{y}_i \\ \hat{y}_i & \text{altrimenti} \end{cases}$$

Dove $par(i)$ specifica i genitori del nodo i.

[1] **Dove il livello è quello del cammino massimo dalla radice**

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.
- È suddiviso in due step sequenziali:

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.
- È suddiviso in due step sequenziali:
  1. **Step bottom-up**: che partendo dai nodi più specifici del DAG, propaga quelle predizioni flat che sono considerate *positive*.

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.
- È suddiviso in due step sequenziali:
  1. **Step bottom-up**: che partendo dai nodi più specifici del DAG, propaga quelle predizioni flat che sono considerate *positive*.
  2. **Step top-down**: È il medesimo step utilizzato dal metodo HTD-DAG.

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.
- È suddiviso in due step sequenziali:
  1. **Step bottom-up**: che partendo dai nodi più specifici del DAG, propaga quelle predizioni flat che sono considerate *positive*.
  2. **Step top-down**: È il medesimo step utilizzato dal metodo HTD-DAG.
- Lo step top down si rende necessario in quanto la propagazione delle predizioni positive dal basso verso l'alto non garantisce la consistenza delle predizioni necessarie alla True Path Rule.

Metodo True Path Rule per DAG (TPR-DAG)

- È un metodo che combina gli approcci top-down e bottom-up per la correzione delle predizioni flat.
- È suddiviso in due step sequenziali:
  1. **Step bottom-up**: che partendo dai nodi più specifici del DAG, propaga quelle predizioni flat che sono considerate *positive*.
  2. **Step top-down**: È il medesimo step utilizzato dal metodo HTD-DAG.
- Lo step top down si rende necessario in quanto la propagazione delle predizioni positive dal basso verso l'alto non garantisce la consistenza delle predizioni necessarie alla True Path Rule.
- La selezione dei nodi considerati *positivi* può avvenire in diverse maniere: con *soglia adattiva*, *senza soglia* e con *soglia fissa*.

Generalized Pool Adjacent Violator (GPAV) (1/4)

- Per il passo Top-Down dell'algoritmo TPR-DAG (e HTD) è stato sviluppato un nuovo metodo all'interno di questa tesi, e cioè **Generalized Pool Adjacent Violator** (GPAV), un algoritmo che permette di risolvere i problemi di **Isotonic Regression**, definiti come:

Isotonic Regression (caso generale con ordinamento parziale)

Dato un DAG, $G(N, E)$, con il set di nodi $N = \{1, 2, ..., n\}$, si deve trovare il vettore $x^* \in R^n$ tale che:

$$min \sum_{i=1}^{n} w_i (x_i - a_i)^2$$

$$\text{such that } x_i \leq x_j \ \forall (i, j) \in E$$

Generalized Pool Adjacent Violator (GPAV) (1/4)

- Per il passo Top-Down dell'algoritmo TPR-DAG (e HTD) è stato sviluppato un nuovo metodo all'interno di questa tesi, e cioè **Generalized Pool Adjacent Violator** (GPAV), un algoritmo che permette di risolvere i problemi di **Isotonic Regression**, definiti come:

Isotonic Regression (caso generale con ordinamento parziale)

Dato un DAG, $G(N, E)$, con il set di nodi $N = \{1, 2, ..., n\}$, si deve trovare il vettore $x^* \in R^n$ tale che:

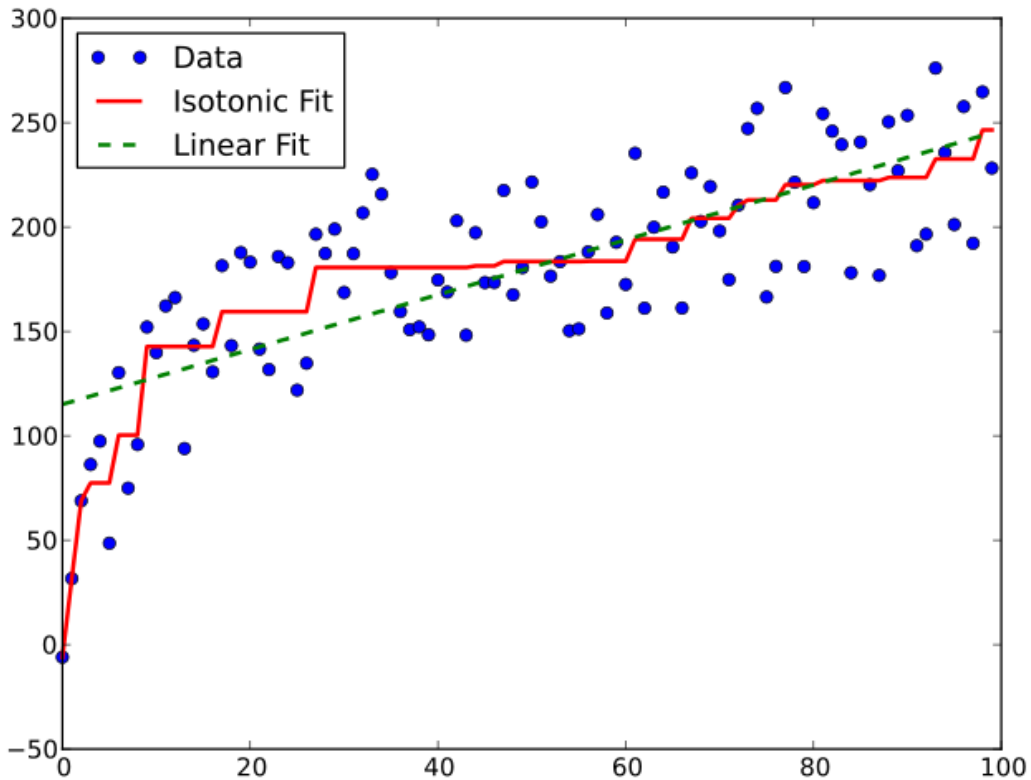
$$min \sum_{i=1}^{n} w_i (x_i - a_i)^2$$

$$\text{such that } x_i \leq x_j \ \forall (i, j) \in E$$

- Con una complessità pari a $O(n^2)$.

Generalized Pool Adjacent Violator (GPAV) (2/4)

Esempio di Isotonic Regression con *ordinamento totale*:



Generalized Pool Adjacent Violator (GPAV) (3/4)

Funzionamento Generale:

Generalized Pool Adjacent Violator (GPAV) (3/4)

Funzionamento Generale:

- Per far funzionare l'algoritmo è necessario effettuare un *ordinamento topologico* del grafo.

Generalized Pool Adjacent Violator (GPAV) (3/4)

Funzionamento Generale:

- Per far funzionare l'algoritmo è necessario effettuare un *ordinamento topologico* del grafo.
- L'algoritmo genera uno split del set di nodi N del DAG, in un insieme di *blocchi disgiunti definiti da* H (inizialmente $H = N$).

Generalized Pool Adjacent Violator (GPAV) (3/4)

Funzionamento Generale:

- Per far funzionare l'algoritmo è necessario effettuare un *ordinamento topologico* del grafo.
- L'algoritmo genera uno split del set di nodi N del DAG, in un insieme di *blocchi disgiunti definiti da* H (inizialmente $H = N$).
- Seguendo l'ordinamento topologico del grafo, un blocco *assorbe* un suo blocco *predecessore* se si verificano determinate condizioni.

Generalized Pool Adjacent Violator (GPAV) (3/4)

Funzionamento Generale:

- Per far funzionare l'algoritmo è necessario effettuare un *ordinamento topologico* del grafo.
- L'algoritmo genera uno split del set di nodi N del DAG, in un insieme di *blocchi disgiunti definiti da* H (inizialmente $H = N$).
- Seguendo l'ordinamento topologico del grafo, un blocco *assorbe* un suo blocco *predecessore* se si verificano determinate condizioni.
- I nodi presenti nel medesimo blocco B_i *condividono il medesimo valore* x_i, e quindi a seguito dell'assorbimento sarà necessario un aggiornamento di tale valore.

Generalized Pool Adjacent Violator (GPAV e ISO-TPR) (4/4)

Algorithm 1 GPAV

1: **procedure** GPAV
2: $H = N$
3: **for** (each $i \in N$) **do**
4: $B_i = \{i\}$
5: $B_i^- = i^-$
6: $x_i = \hat{y}_i$
7: $W_i = w_i$
8: **for** $k = 1, 2, ..., n$ **do**
9: *//finché esiste un predecessore di B_k che viola la monotonicità*
10: **while** $\{i \in B_k^- : x_i > x_k\} \neq 0$ **do**
11: *// Trova l'elemento che viola maggiormente il vincolo*
12: **Find** $j \in B_k^- : x_j = max\{x_i : i \in B_k^-\}$
13: **Absorb(k, j)** *// j viene assorbito da B_k*

Generalized Pool Adjacent Violator (GPAV e ISO-TPR) (4/4)

Algorithm 2 GPAV

procedure GPAV
  $H = N$
  for (each $i \in N$) **do**
    $B_i = \{i\}$
    $B_i^- = i^-$
    $x_i = \hat{y}_i$
    $W_i = w_i$
  for $k = 1, 2, ..., n$ **do**
    //finché esiste un predecessore di B_k che viola la monotonicità
    while $\{i \in B_k^- : x_i > x_k\} \neq 0$ **do**
      // Trova l'elemento che viola maggiormente il vincolo
      Find $j \in B_k^- : x_j = max\{x_i : i \in B_k^-\}$
      Absorb(k, j) *// j viene assorbito da B_k*

- Riassumendo, l'algoritmo effettua degli assorbimenti di blocchi adiacenti, finché questi violano i vincoli del problema quadratico, generando di fatto una partizione dei nodi, in cui le parti condividono lo stesso valore.

Generalized Pool Adjacent Violator (GPAV e ISO-TPR) (4/4)

Algorithm 3 GPAV

procedure GPAV
  $H = N$
  for (each $i \in N$) **do**
    $B_i = \{i\}$
    $B_i^- = i^-$
    $x_i = \hat{y}_i$
    $W_i = w_i$
  for $k = 1, 2, ..., n$ **do**
    //finché esiste un predecessore di B_k che viola la monotonicità
    while $\{i \in B_k^- : x_i > x_k\} \neq 0$ **do**
      // Trova l'elemento che viola maggiormente il vincolo
      Find $j \in B_k^- : x_j = max\{x_i : i \in B_k^-\}$
      Absorb(k, j) *// j viene assorbito da B_k*

- Riassumendo, l'algoritmo effettua degli assorbimenti di blocchi adiacenti, finché questi violano i vincoli del problema quadratico, generando di fatto una partizione dei nodi, in cui le parti condividono lo stesso valore.
- Sostituendo GPAV allo step Top-Down dell'algoritmo TPR-DAG visto in precedenza (invece che HTD), si ottiene l'algoritmo **ISO-TPR**, un altro nuovo metodo utilizzato in questa tesi.

Predizione della funzione delle proteine in C.elegans (WORM)

- Si è eseguita la sperimentazione sul genoma della specie *Caenorhabditis elegans* (WORM), utilizzando come insieme delle istanze e input del problema una matrice simmetrica generata dal network di interazione proteina-proteina *STRING*[2].

[2] **Search Tool for the Retrieval of Interacting Genes/Proteins**

Predizione della funzione delle proteine in C.elegans (WORM)

- Si è eseguita la sperimentazione sul genoma della specie *Caenorhabditis elegans* (WORM), utilizzando come insieme delle istanze e input del problema una matrice simmetrica generata dal network di interazione proteina-proteina *STRING*[2].
- Tale matrice STRING ha dimensione 15752×15752 (WORM). Il nostro problema ha quindi 15752 istanze.

[2] **Search Tool for the Retrieval of Interacting Genes/Proteins**

Predizione della funzione delle proteine in C.elegans (WORM)

- Si è eseguita la sperimentazione sul genoma della specie *Caenorhabditis elegans* (WORM), utilizzando come insieme delle istanze e input del problema una matrice simmetrica generata dal network di interazione proteina-proteina *STRING*[2].
- Tale matrice STRING ha dimensione 15752×15752 (WORM). Il nostro problema ha quindi 15752 istanze.
- In base al tipo do ontologia, si hanno DAG con un quantitativo diverso di nodi(termini) e archi(relazioni):

ontologia	numero di termini	numero di archi
BP	4068	8066
MF	1163	1567
CC	578	1082

[2] **Search Tool for the Retrieval of Interacting Genes/Proteins**

Annotazioni per il dataset C.elegans (WORM)

Per evitare di avere problemi nella fase di cross-validazione, i DAG sono stati ridotti a quei termini per cui si hanno almeno 10 annotazioni. Un po' di statistiche a seguito della selezione:

Onto	**numero di termini**	**media**	**d.std.**	**massimo**	**minimo**
BP	1335	71,33	151,68	2597	10
MF	186	61,23	191,84	1806	10
CC	221	131,9	302,25	1924	10

Tabella: La colonna *numero di termini* indica il numero di termini ottenuti dopo la selezione, *media* la media delle annotazioni per classe, per l'ontologia di riferimento, *d.std.* la deviazione standard delle annotazioni per l'ontologia di riferimento, *massimo* e *minimo* rispettivamente il massimo e minimo numero di annotazioni.

Algoritmi di ML per le predizioni flat

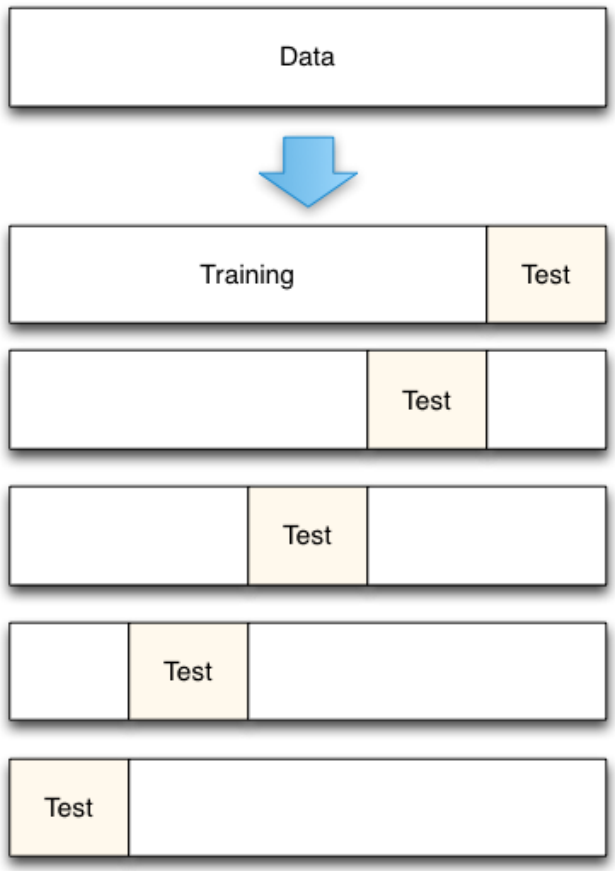
- Per quanto riguarda gli algoritmi di apprendmento automatico da utilizzare per le predizioni flat, si sono considerati i seguenti metodi:
  1. *K-Nearest Neighbors*
  2. *Logit Boost*
  3. *Linear Discriminant Analysis*
  4. *eXtreme Gradient Boosting*
  5. *C5.0* (Alberi di decisione)
  6. *Random Forest*
  7. *Multilayer Perceptron*
  8. *Support Vector Machine lineare*
  9. *Bagged CART* (Bagged ensemble di alberi di decisione)
  10. *AdaBoost.M1*
  11. *Naive Bayes*
  12. *Glmnet*

Algoritmi di ML per le predizioni flat

- Per quanto riguarda gli algoritmi di apprendmento automatico da utilizzare per le predizioni flat, si sono considerati i seguenti metodi:
  1. *K-Nearest Neighbors*
  2. *Logit Boost*
  3. *Linear Discriminant Analysis*
  4. *eXtreme Gradient Boosting*
  5. *C5.0* (Alberi di decisione)
  6. *Random Forest*
  7. *Multilayer Perceptron*
  8. *Support Vector Machine lineare*
  9. *Bagged CART* (Bagged ensemble di alberi di decisione)
  10. *AdaBoost.M1*
  11. *Naive Bayes*
  12. *Glmnet*
- Dato l'elevato numero di algoritmi selezionati per la sperimentazione, si è deciso di non effettuare il tuning dei parametri, questo per evitare di allungare ulteriormente i tempi dell'intero processo di valutazione e generazione degli score flat.

Cross-validazione e metriche usate per la valutazione degli algoritmi

- Per stimare le performance dei nostri predittori si è utilizzata la tecnica della *cross-validazione*, la quale permette di valutare l'errore di un algoritmo di apprendimento stimando l'errore medio dei predittori prodotti dall'algoritmo.

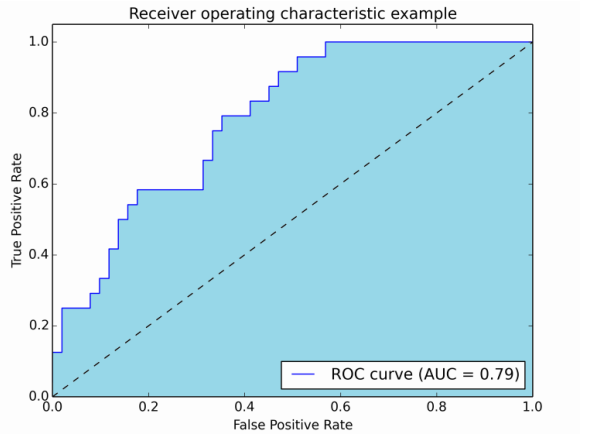


Cross-validazione e metriche usate per la valutazione degli algoritmi

- Per stimare le performance dei nostri predittori si è utilizzata la tecnica della *cross-validazione*, la quale permette di valutare l'errore di un algoritmo di apprendimento stimando l'errore medio dei predittori prodotti dall'algoritmo.
- Come metriche si sono poi usate la:

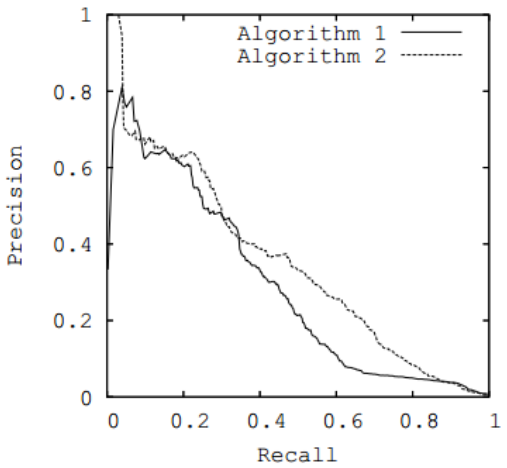
Cross-validazione e metriche usate per la valutazione degli algoritmi

- Per stimare le performance dei nostri predittori si è utilizzata la tecnica della *cross-validazione*, la quale permette di valutare l'errore di un algoritmo di apprendimento stimando l'errore medio dei predittori prodotti dall'algoritmo.
- Come metriche si sono poi usate la:
  1. *AUROC*: una misura di robustezza del classificatore, che mette in relazione le misure di Recall e False Positive Rate, al variare di una soglia applicata all'output del modello.



Cross-validazione e metriche usate per la valutazione degli algoritmi

- Per stimare le performance dei nostri predittori si è utilizzata la tecnica della *cross-validazione*, la quale permette di valutare l'errore di un algoritmo di apprendimento stimando l'errore medio dei predittori prodotti dall'algoritmo.
- Come metriche si sono poi usate la:
  1. *AUROC*: una misura di robustezza del classificatore, che mette in relazione le misure di Recall e False Positive Rate, al variare di una soglia applicata all'output del modello.
  2. *AUPRC*: è una misura che mette in relazione la variazione di Precision al variare della Recall ed è utile nel caso di problemi sbilanciati.



Cross-validazione e metriche usate per la valutazione degli algoritmi

- Per stimare le performance dei nostri predittori si è utilizzata la tecnica della *cross-validazione*, la quale permette di valutare l'errore di un algoritmo di apprendimento stimando l'errore medio dei predittori prodotti dall'algoritmo.
- Come metriche si sono poi usate la:
  1. *AUROC*: una misura di robustezza del classificatore, che mette in relazione le misure di Recall e False Positive Rate, al variare di una soglia applicata all'output del modello.
  2. *AUPRC*: è una misura che mette in relazione la variazione di Precision al variare della Recall ed è utile nel caso di problemi sbilanciati.
  3. *F-Score gerarchica*: è una metrica centrata sui geni, massimizzata al variare di una soglia t in (0, 1). Tale misura si basa sulla Precisione e Recall centrate sui geni e non su le classi. Tale metrica è stata usata per il fit degli iperparametri dei metodi ensemble (soglia e pesi).

$$Precision(t) = \frac{1}{n}\sum_{j=1}^{n}\frac{TruePositive_j(t)}{TruePositive_j(t) + FalsePositive_j(t)}$$

$$Recall(t) = \frac{1}{n}\sum_{j=1}^{n}\frac{TruePositive_j(t)}{TruePositive_j(t) + FalseNegative_j(t)}$$

$$Fmax = \max_t \frac{2Precision(t)Recall(t)}{Precision(t) + Recall(t)}$$

Stima preliminare dei tempi di calcolo degli algoritmi flat

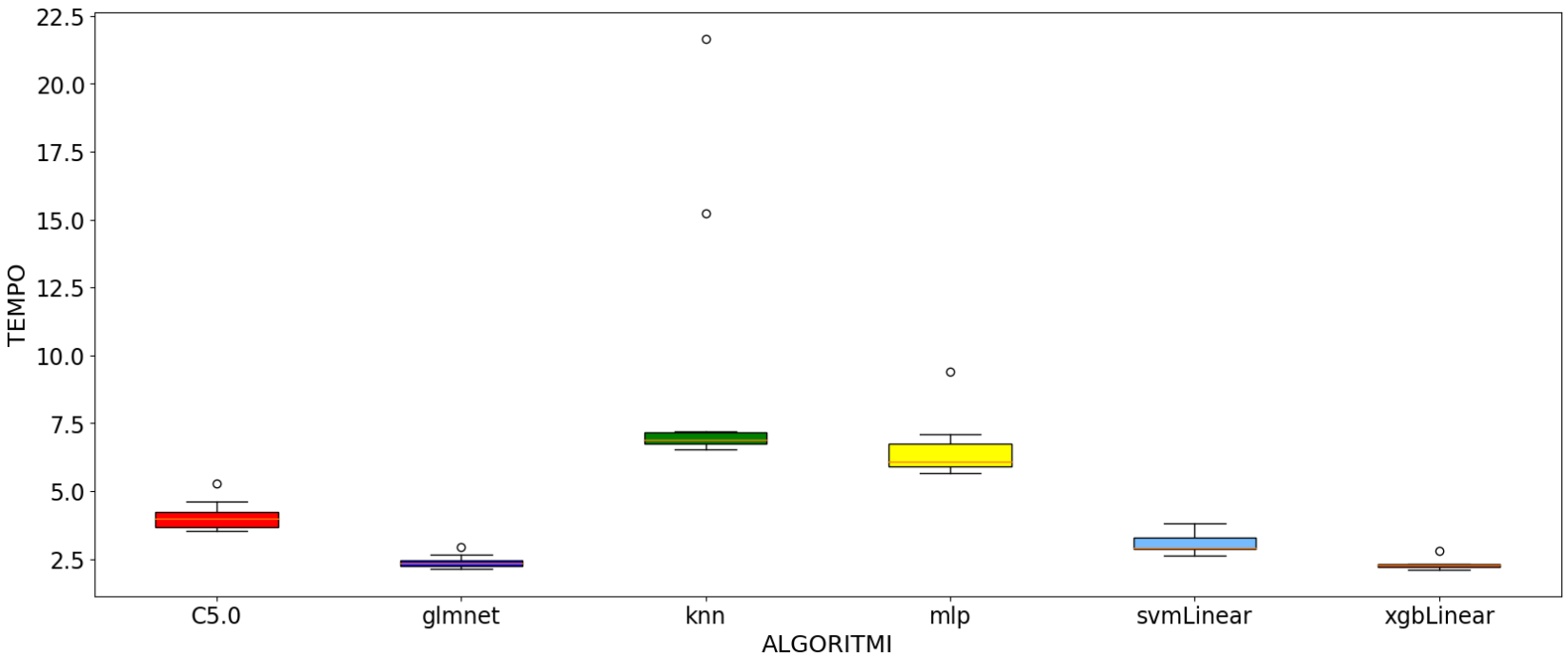


Figura: Il box plot dei tempi di esecuzione, con cross-validation a 10 fold per l' ontologia BP. I tempi sono da intendersi in ore e per classe, per un campione di 10 classi.