

Metodi di Ingegneria della Conoscenza applicati alle Homepage delle scuole superiori italiane

# Corso didattico

* Ingegneria della Conoscenza [063507], Facoltà di “Informatica”
* A.A. 2022/23

# Gruppo di lavoro

* Vincenzo Di Bisceglie [745751] [v.dibisceglie3@studenti.uniba.it](mailto:v.dibisceglie3@studenti.uniba.it)

# Repository

* <https://github.com/vodibe/icon-74571>

# Sommario

Sommario

[Corso didattico 1](#_Toc151326046)

[Gruppo di lavoro 1](#_Toc151326047)

[Repository 1](#_Toc151326048)

[Sommario 2](#_Toc151326049)

[Introduzione 3](#_Toc151326050)

[Idea del progetto 3](#_Toc151326051)

[Metriche di usabilità già esistenti 3](#_Toc151326052)

[Metrica di usabilità adottata per questo progetto 3](#_Toc151326053)

[Elenco argomenti di interesse 4](#_Toc151326054)

[Costruzione del ground truth 5](#_Toc151326055)

[Sommario 5](#_Toc151326056)

[Decisioni di progetto 5](#_Toc151326057)

[Preprocessing del dataset delle scuole 7](#_Toc151326058)

[Rappresentazione dello spazio di ricerca con grafo e ricerca soluzioni 8](#_Toc151326059)

[Sommario 8](#_Toc151326060)

[Strumenti utilizzati: modello NaiveDOM 8](#_Toc151326061)

[Decisioni di progetto 13](#_Toc151326062)

[Valutazione 14](#_Toc151326063)

[Apprendimento Supervisionato 15](#_Toc151326064)

[Sommario 15](#_Toc151326065)

[Strumenti utilizzati e decisioni di progetto 15](#_Toc151326066)

[Valutazione 15](#_Toc151326067)

[Ragionamento con Incertezza: Rete Bayesiana 19](#_Toc151326068)

[Sommario 19](#_Toc151326069)

[Strumenti utilizzati 19](#_Toc151326070)

[Decisioni di progetto 19](#_Toc151326071)

[Valutazione 19](#_Toc151326072)

[Conclusioni 20](#_Toc151326073)

[Bibliografia 21](#_Toc151326074)

# Introduzione

## Idea del progetto

**L’idea di fondo** da cui si è partiti per lo sviluppo di questo progetto è l’applicazione di alcuni metodi di Ingegneria della Conoscenza su un dominio di interesse, in questo caso l’usabilità di una pagina web. Questo richiede che prima si vada a circoscrivere un ambito di riferimento, che nel nostro caso, è l’insieme delle Homepage delle scuole superiori pubbliche italiane (aggiornate a settembre 2023).

## Metriche di usabilità già esistenti

Le **metriche rilevanti proposte in letteratura** e che potrebbero essere applicate nel contesto di questo progetto sono le Euristiche di Nielsen  [[1]](#b01) le WCAG 2.1  [[2]](#b02) per le quali però gli strumenti software ad essi correlati ([qui elencati](https://www.w3.org/WAI/ER/tools/?q=wcag-21-w3c-web-content-accessibility-guidelines-21)) non sono adatti alla natura di questo progetto perché non esprimono una valutazione numerica, ma analizzano il codice sorgente della pagina e danno consigli per rimediare le linee guida non rispettate. Altri strumenti controllano condizioni di accessibilità da parte di utenti con handicap (ad es. verificano che la palette di colori sia accessibile, controllano l’interazione con hardware ausiliari, …)

Altre metriche rilevanti (SUS Score  [[3]](#b03)) non sono state prese in considerazione perché richiedono un campione di persone alle quali sottoporre un questionario.

## Metrica di usabilità adottata per questo progetto

Ai fini del progetto assumeremo che questa **nuova metrica di usabilità** corrisponde a un voto assegnato da una persona che non ha mai interagito con la Homepage prima d’ora, tenendo conto di quanto l’interfaccia sia ordinata e funzionale. **Nel corso di questo progetto questa metrica è vista come un qualcosa di condiviso dai visitatori (concetto oggettivo), e non come una valutazione soggettiva che uno specifico utente dà alla pagina**. Approfondiremo questa metrica nelle sezioni successive.

# Elenco argomenti di interesse

Fasi del progetto e per ciascuna di esse gli argomenti coinvolti:

1. [**Costruzione del ground truth**](#_Costruzione_del_ground).  
   Poiché all’inizio non disponiamo di una valutazione per tutte le Homepage, ci immedesimiamo in un visitatore della pagina, ne osserviamo gli aspetti grafici e funzionali (in altre parole osserviamo il valore di alcune feature iniziali), e diamo una valutazione. Gli step seguiti sono:
   1. Preprocessing del dataset delle scuole.
   2. Raccolta dei dati in un dataset rappresentante il ground truth.
2. Emulazione del ground truth.  
   La fase 1 prevede un’osservazione diretta della grafica, e ciò ovviamente non può essere automatizzato, ma deve essere valutato con criterio. Pertanto in questa fase riproduciamo il ground truth utilizzando strumenti che si prestano meglio all’elaborazione e apprendimento automatico. Gli step seguiti sono:
   1. Osservazione di caratteristiche della pagina ottenibili in modo automatico per ciascun sito, mediante [**Rappresentazione dello spazio di ricerca tramite grafo**](#_Rappresentazione_dello_spazio).
   2. Costruzione e valutazione di [**Modelli di apprendimento supervisionato**](#_Apprendimento_Supervisionato) che, a partire dalle feature per ciascun sito (individuate al punto 2.1) simulano la sua valutazione.
3. Deduzione di informazioni utili.
   1. [**Costruzione di un modello probabilistico**](#_Modello_probabilistico:_Rete) e suo impiego per task di inferenza probabilistica.
   2. Operazioni di deduzione con ragionamento relazionale e basi di conoscenza.

# Costruzione del ground truth

## Sommario

Si è ipotizzato che in generale un visitatore quando osserva la pagina, può assegnare il grado di usabilità con una scala [1, 5] con una precisione (step) di 0.1 punti.

* [1, 2): **sito estremamente confuso**  
  Non esiste un menu; la disposizione di tutti gli elementi è disordinata, per cui è difficile individuare le sezioni che l’utente vuole visitare.
* [2, 3): **sito confuso es:** [**https://www.galileiferrari.it/**](https://www.galileiferrari.it/)  
  Esiste un menu; la disposizione di quasi tutti gli elementi della pagina è disordinata e la pagina dà l’impressione di essere troppo lunga.
* [3, 4): **sito accettabile es:** [**https://www.isii.it/**](https://www.isii.it/)  
  Esiste un menu che reindirizza il visitatore a gran parte delle sezioni di suo interesse; la pagina però contiene un discreto numero di elementi non raggruppati e quindi confusionari.
* [4, 5]: **sito ordinato es:** [**https://www.einsteinrimini.edu.it/**](https://www.einsteinrimini.edu.it/)  
  Sito accettabile e che inoltre contiene pochi o nessun elemento non raggruppato.

## Decisioni di progetto

Si suppone che l’utente vada ad assegnare un valore di usabilità alla pagina ragionando su alcuni fattori. Per comodità, è utile raccogliere i fattori di decisione e la valutazione finale in un dataset che chiamiamo ds2\_gt.

La prima cosa che consideriamo vedendo una pagina web scolastica può essere la presenza di un menu, per cui si introduce la feature discreta **page\_menu\_or** che ne descrive l’orientamento.

A questo punto, viene introdotto un secondo fattore, il più rilevante, dovuto al fatto che nella quasi totalità dei siti scolastici ritroviamo il “trend” di inserire dei banner che linkano a una sezione del sito. Spesso, tali banner sono difficili da leggere e posti sulla pagina in modo disordinato, cioè non raggruppati in un menu o in una sezione specifica della pagina. Per generalizzare (includendo qualsiasi contenuto multimediale, e quindi anche video) introduciamo la feature **page\_ungrouped\_multim**.



Figura 1. Esempi di banner.

Per ultimo, c’è la feature **page\_template**, utile a fornire un contesto in cui “inquadrare” la feature page\_ungrouped\_multim. Questo accade in quanto possono esistere più pagine che, seppur hanno lo stesso numero di elementi multimediali non raggruppati, risultano in una valutazione diversa perché basate appunto su template diversi.

Potremmo ipotizzare che la valutazione possa dipendere anche da quanto sia lunga la pagina, tuttavia un visitatore non viene mai a conoscenza dell’altezza precisa (in pixel). Pertanto non è stata considerata.

|  |  |  |  |
| --- | --- | --- | --- |
| *Fattore di decisione* | *Descrizione* | *Dominio* | *u.m.* |
| page\_template | Template adottato (vedi Figura 2) 1,…,8=template ID 9=non segue un template |  |  |
| page\_menu\_or | Orientamento menu. 0=non esiste 1=solo orizzontale 2=solo verticale 3=orizzontale e verticale |  |  |
| page\_ungrouped\_multim | Elementi grafici non raggruppati. |  |  |

A quale pagina web sono associati questi fattori?

|  |  |  |  |
| --- | --- | --- | --- |
| *Feature PK* | *Descrizione* | *Dominio* | *u.m.* |
| school\_id | Codice della scuola. | Stringhe |  |
| page\_url | URL della pagina. | Stringhe |  |

La valutazione è la seguente:

|  |  |  |  |
| --- | --- | --- | --- |
| *Feature* | *Descrizione* | *Dominio* | *u.m.* |
| metric | Valutazione di usabilità della pagina. |  |  |

Di seguito è presente la Figura 2, con tutti i template ad oggi impiegati dai siti scolastici italiani:

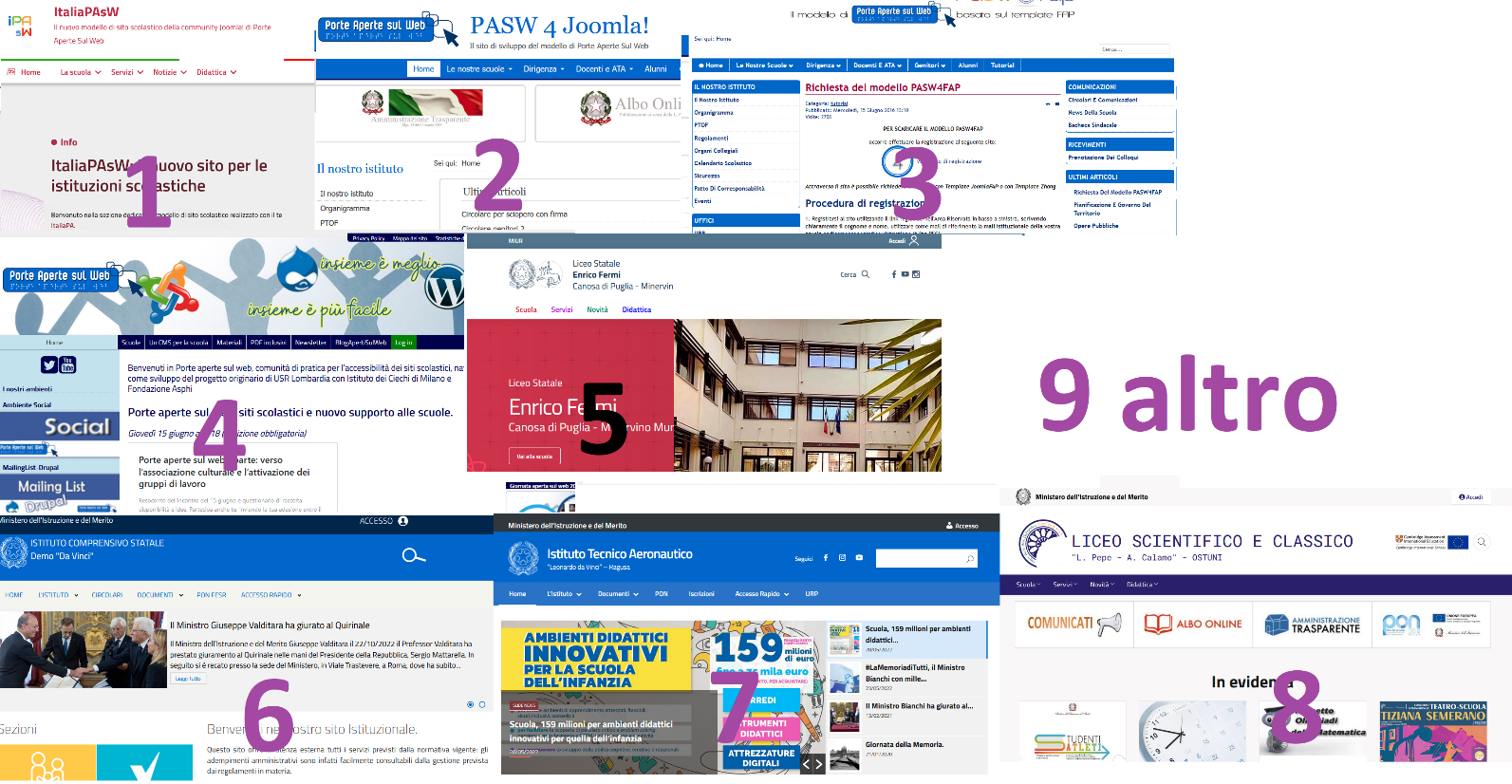


Figura 2.

1. <https://paswjoomla.net/Jipa4school/>   
2. <http://paswjoomla.net/pasw/>   
3. <http://paswjoomla.net/jfap/>   
4. <https://www.porteapertesulweb.it/>   
5. <https://italia.github.io/design-scuole-pagine-statiche/scuole-la-scuola.html>  
6. <https://italiajoo.demoargoweb.com/>   
7. <https://italiawp.demoargoweb.com/>   
8. <https://www.liceopepecalamo.edu.it/>

## Preprocessing del dataset delle scuole

Vedere: /agent/preproc/dataset\_creator.py

Una volta introdotta la metrica, è necessario parlare della fase di preprocessing del dataset iniziale. Il catalogo offerto dal MIUR raggruppa le informazioni su tutte le scuole (elementari, medie e superiori) pubbliche. Durante la fase di preprocessing si vanno a creare, in ordine, i seguenti DS:

1. **ds1**: <https://dati.istruzione.it/opendata/opendata/catalogo/elements1/?area=Scuole>  
   Le feature di questo DS sono descritte in [questa pagina web](https://dati.istruzione.it/opendata/opendata/catalogo/elements1/leaf/?area=Scuole&datasetId=DS0400SCUANAGRAFESTAT#tracciato).
2. **ds1\_clean** ottenuto inserendo solo le scuole superiori ed effettuando un preprocessing sull’URL che consiste nel vedere se il sito corrente è rintracciabile con una semplice richiesta HTTP. Se non lo è, ed inoltre il sito ha un TLD diverso da .edu, si sostituisce il TLD corrente con .edu. Se un sito non è rintracciabile neanche dopo aver effettuato la sostituzione, lo si esclude dal DS.
3. **ds1\_clean\_unique** ottenuto rimuovendo i siti duplicati. Operazione necessaria in quanto se un plesso scolastico offre più corsi di studio (ad es. istituto tecnico e istituto professionale) e ha un singolo sito web, ciascun corso ricopre una riga nel ds1.
4. **ds2\_gt** ottenuto richiedendo i fattori di valutazione e la valutazione per ciascun sito presente in ds1\_clean\_unique.
5. **ds3\_gt** ottenuto inserendo tutte le features necessarie per addestrare un modello di apprendimento. Verranno descritte nel capitolo dedicato.
6. **ds3\_gt\_final**. E’ possibile che il ds3\_gt contenga qualche URL non valido, dovuto al fatto che il sito è in manutenzione, ha subito un cambio dominio o che faccia riferimento a una scuola superiore erroneamente catalogata nel ds1 (ad esempio esclusivamente serale). Pertanto, queste righe vengono rimosse in questo nuovo DS.

In sintesi:

* Siti di scuole superiori ma che non sono raggiungibili, oppure siti di altre scuole (medie, convitti, …)
* (accanto a *unique*) Siti duplicati.
* (accanto a *final*) Siti raggiungibili ma non validi.
* Siti delle scuole superiori raggiungibili e validi.



Figura 3. Proporzione del numero di elementi dei vari DS.

# Rappresentazione dello spazio di ricerca con grafo e ricerca soluzioni

## Sommario

Per eseguire il task di apprendimento del ground truth, dobbiamo individuare alcune features che possano essere osservate automaticamente a partire dalla pagina web. Per cui, notando che una pagina web equivale a un DOM, possiamo usare in questo progetto il concetto di rappresentazione dello spazio di ricerca tramite grafo.

## Strumenti utilizzati: modello NaiveDOM

Vedere: /agent/ndom/NaiveDOM.py

In questo progetto è stato introdotto il concetto di NaiveDOM (NDOM) che è un modello DOM semplificato di una pagina web ottenuto dal parsing del codice sorgente HTML. I suoi dettagli teorici sono presentati nella prossima sezione.

### Struttura del NDOM

Un NDOM è un grafo diretto e pesato, avente struttura ad albero. E’ tale per cui:

* Ha numero finito di nodi ed è aciclico (diretta conseguenza del fatto che è un DOM semplificato)
* Ciascun nodo è un elemento della pagina, e quindi è identificato univocamente dal suo XPath  [[4]](#b04).

A ciascun nodo sono associati una label (per fini di rappresentazione grafica) e le sue coordinate (x, y) all’interno della pagina renderizzata.

* Il **nodo radice** è l’XPath del tag <body>.
* I **nodi interni** sono gli XPath dei tag che contengono potenzialmente, tra i loro discendenti, un testo leggibile. Ad es. <body>, <header>, <section>, <nav> ecc… Sono esclusi i tag <div>, visto che sono assai frequenti e non semplificano (ma complicano) la struttura del NDOM.
* I **nodi foglia** possono essere di tre tipi:
  + XPath dei tag che non contengono un testo leggibile, ad es. <img>, ecc…
  + XPath dei tag che contengono sicuramente un testo leggibile, ad es. <a>, <h1>, ecc…
  + Il testo leggibile, a patto che abbia una lunghezza breve.
* Come un qualsiasi albero, ha una sua **altezza**, cioè un numero indicante la massima profondità di un nodo.
* Per quanto riguarda gli **archi** del NDOM e il loro costo, è necessario prima osservare direttamente un esempio di NDOM costruito per una pagina. Si veda la prossima sezione.

I dettagli implementativi di questo modello sono descritti nella sezione *Decisioni di progetto*.

### Calcolo del costo degli archi

Visualizziamo un sito scolastico, e rappresentiamo in forma stilizzata il suo NDOM.



Figura 4. <https://www.liceotedone.edu.it/> . Rappresentazione stilizzata del NDOM.

Per questa sezione osserviamo lo screenshot di un sito web (Figura 3) e notiamo che il nodo radice <body> ha ovviamente coordinate (0,0). I rettangoli arancioni indicano elementi della pagina innestati all’interno del tag <body>. Solo per questi elementi (figli diretti della radice del NDOM), la distanza tra padre e figlio è puramente verticale: questo è ovvio perché l’occhio umano inizia osservando dal basso verso l’alto. In tutti gli altri casi si provvede a calcolare la distanza euclidea.

Il **costo dell’arco tra padre-figlio** è una funzione della distanza padre-figlio (di seguito chiamata ), ed è calcolata in \_calc\_arc\_cost. Essenzialmente si riconduce alla seguente funzione:

La funzione descritta di calcolo del costo padre-figlio ha il seguente comportamento. La linea verde chiaro fa riferimento agli schermi con risoluzione 1600x900, quella verde scuro agli schermi 1020x1080. Man mano che aumenta la distanza in pixel tra un elemento, aumenta il costo in termini di usabilità. A parità di distanza, il costo (su schermi con risoluzione minore) aumenta.



Figura 5. Grafico funzione di costo dell’arco padre-figlio. <https://www.desmos.com/calculator/aaqy3tao8g>

### Calcolo di un task con algoritmo di ricerca

Vedere: /agent/ndom/NaiveDOM.py  
 /agent/ndom/NaiveDOMSearcher.py

La costruzione del NDOM di una pagina web richiede un’istanza di un browser automatizzato che disponga di un interprete JS. Per questo motivo, è possibile ricavare altre due feature inerenti ad essa. Al termine, siamo in grado di ingegnerizzare 13 nuove feature:

|  |  |  |  |
| --- | --- | --- | --- |
| *Feature* | *Descrizione* | *Dominio* | *u.m.* |
| page\_load\_time\_ms | Tempo di caricamento della pagina. |  | ms |
| page\_width | Larghezza della pagina. |  | px |
| page\_height | Altezza della pagina. |  | px |
| NDOM\_nodes | Numero di nodi del NDOM associato alla pagina. |  |  |
| NDOM\_height | Altezza del NDOM associato alla pagina. |  |  |
| task1 | Costo in termini di usabilità per svolgere il task #1. |  |  |
| … | | | |
| task8 | Costo in termini di usabilità per svolgere il task #8. |  |  |

**Come si calcola il valore della feature taskx?**

Innanzitutto, un Task è una sezione che l’utente è interessato a raggiungere e che, se individuata, in un certo senso rispecchia l’usabilità della pagina. Un Task contiene un Task ID e delle Task Keywords, cioè una lista di stringhe tali per cui, se l’utente ne individua una all’interno della pagina, porta a termine il suddetto Task. Il dizionario dei Task è mostrato di seguito, e raccoglie alcune sezioni tipiche di un sito scolastico.



Figura 6.

A questo punto, l’algoritmo di ricerca proposto (chiamato nel codice come NaiveDOMSearcher) cerca di emulare il comportamento dell’occhio umano, e ciò è rappresentato dall’immagine in Figura 7: un percorso per il quale l’ultimo nodo ha profondità 0 o 1 (vale a dire, il nodo radice e tutti i percorsi dal nodo radice ai suoi figli diretti) viene aggiunto a una coda con priorità, in cui il percorso a costo minore sarà il primo ad essere esaminato. Questo è ovvio perché una persona passa ad esaminare prima le voci del menu principale rispetto alle voci del footer (che si trovano a fine pagina). Successivamente, gli alberi radicati nei figli diretti della radice vengono esaminati in modalità DFS.

Di seguito è illustrato il suo funzionamento.

****

Minore per prima

**DFS**

Figura 7. Metodo plot() chiamato sul NDOM di <https://www.liceofermicanosa.edu.it/> e illustrazione del funzionamento dell’algoritmo di ricerca.

Questo algoritmo di ricerca gode delle seguenti proprietà:

* E’ completo, cioè certo di trovare un nodo obiettivo se esso esiste.
* Non va in loop (diretta conseguenza della struttura del NDOM)
* Come l’algoritmo DFS, ha complessità di spazio ove è il branching factor e è la profondità del nodo goal; complessità di tempo .
* E’ ottimale?

Se esiste un percorso dal nodo radice a un nodo obiettivo per il task, chiamiamo con il costo del percorso (cioè la somma di tutti i costi degli archi) e applichiamo la seguente funzione che aggiunge alcune penalità.

Ad esempio:

La prima penalità dipende dal numero di percorsi già esaminati prima di individuare quello dalla radice al nodo obiettivo. La seconda è dovuta al fatto che, se ipotizziamo che il nodo A ha come figlio il nodo B, il passare dall’interagire con il nodo A all’interagire con il nodo B richiede raramente (ma comunque non è impossibile) un’operazione scomoda da fare, come un click, l’attesa di un’animazione ecc…   
Idealmente, un sito con costi molto bassi per i task è tale per cui tutte le sezioni utili sono elencate chiaramente o in menu a tendina.

Se non esiste almeno un nodo obiettivo per un Task, si assegna al task un costo di default. Questa casistica avviene quando nella pagina non c’è una stringa visibile che soddisfa il task. Le cause sono:

* il designer del sito non prevede l’inserimento di una sezione correlata al Task (grave)
* il nodo obiettivo non è una stringa visibile, ma un’immagine (comprensibile).

In questo progetto NON sono state implementate tecniche per gestire la seconda causa (ad es. tecniche OCR), per cui il costo di default è di media entità, risultante dal compromesso tra la prima e la seconda causa.

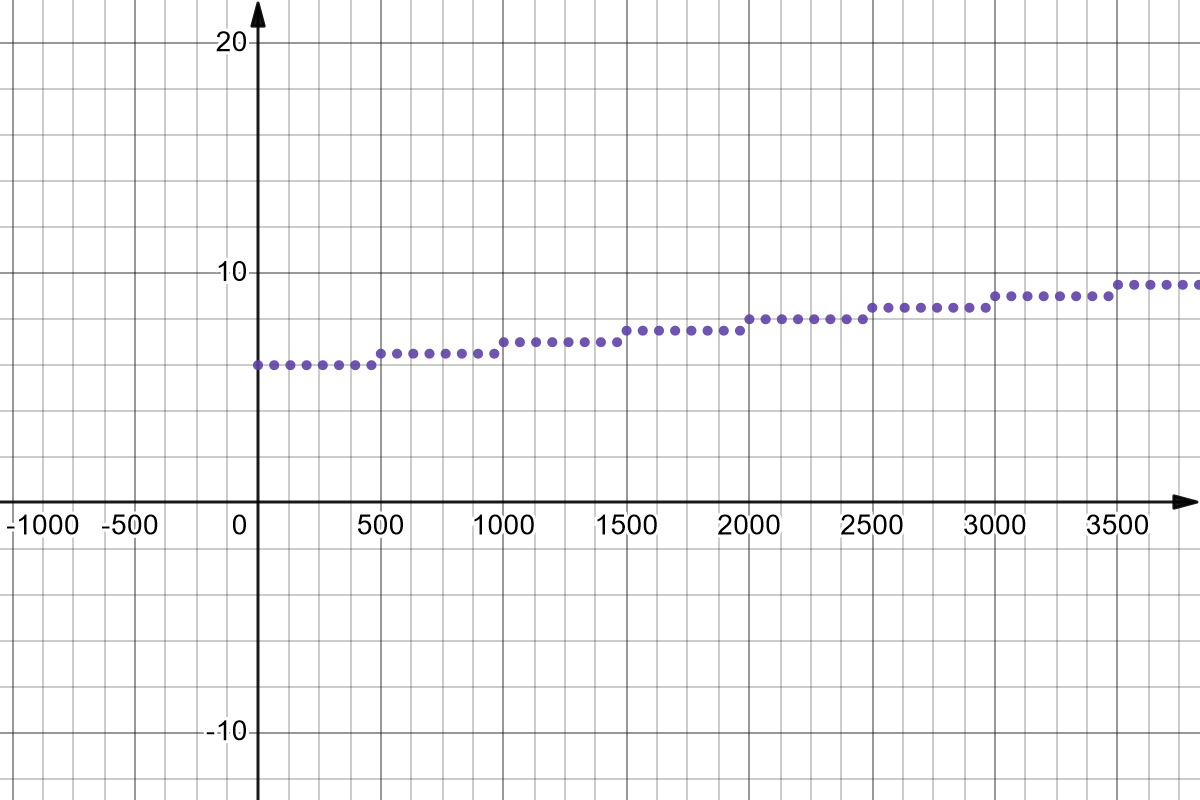
****

Figura 8. Grafico funzione .  
<https://www.desmos.com/calculator/2epakrbyrj>

## Decisioni di progetto

Le librerie utilizzate in questa sezione sono [Selenium](https://www.selenium.dev/) per la creazione di un’istanza del browser Firefox che contiene un’interprete JS. Quest’ultimo è utile per calcolare le coordinate di un nodo e inserirle in un dizionario. Ovviamente ciò è fattibile solo dopo aver renderizzato la pagina ed eseguito del codice JS.  
Questo viene fatto mediante funzione \_create\_driver(width, height) ove width e height sono inizializzate a 1600 e 900.

La libreria [Beautifulsoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) è stata usata per il parsing del codice sorgente e la definizione di una funzione ricorsiva di creazione del NDOM. In merito a Beautifulsoup, è necessario scegliere il parser lxml o html5lib, visto che sono gli unici in grado di gestire eventuali tag non chiusi. Prima di utilizzare questa libreria comunque, è stato fatto un preprocessing del codice sorgente che rimuove i commenti, spazi inutili e i tag proibiti.

Si è pensato di programmare il modello NDOM come una classe Python avente la seguente interfaccia (a sinistra). Al centro c’è la lista di tag HTML che vengono rimossi prima ancora di iniziare la costruzione del NDOM, a destra ci sono i tag HTML che si possono assumere nodi foglia e nodi interni.

Figura 9.

Il metodo \_browse\_DOM è quello adibito alla costruzione del NDOM ed è presentato in pagina successiva. Si sfoglia il codice sorgente in modalità DFS, si stabilisce se un nodo è il nodo radice/interno/foglia del NDOM e si ottiene la sua label. Una volta esaminato il nodo corrente ed aver individuato i suoi nodi figli, si invoca una chiamata ricorsiva.





Come possiamo vedere dall’interfaccia in Figura 9, oltre agli attributi che descrivono la struttura del modello (nodes\_goal, start, location, nodes, nodes\_coords, arcs), c’è l’attributo features (dizionario delle 13 feature dette prima) e gli attributi search\_alg e nodes\_expanded\_per\_task.

search\_alg è una stringa che identifica uno tra gli algoritmi di ricerca di un Task non informati che è possibile applicare: “NaiveDOMSearcher”, “DFS”, “BFS”, “LCFS”. Per poter garantire questa funzionalità è stata modificata la classe Searcher della libreria [AIPython](https://artint.info/AIPython/) aggiungendo al costruttore il parametro algorithm e modificando i metodi chiamati al momento dell’inserimento/rimozione di un percorso in frontiera. E’ un attributo che risulta utile per il confronto degli algoritmi. (sezione successiva).

nodes\_expanded\_per\_task è un attributo (dizionario) autoesplicativo: per un dato algoritmo di ricerca impiegato, associa ad ogni task il numero di nodi che si sono esaminati prima di giungere a un nodo obiettivo.

## Valutazione

Vedere: /agent/ndom/benchmark.py

In questa sezione valutiamo l’algoritmo di ricerca costruito mettendolo a confronto con altri algoritmi non informati DFS, BFS e LCFS. Consideriamo un insieme di indirizzi web rappresentanti il 30% del DS. Successivamente costruiamo automaticamente 4 NDOM per ciascun sito, ciascuno dei quali usa un algoritmo di ricerca dei nodi obiettivo diverso. Il numero di nodi esaminati per ogni NDOM (e per ogni Task) è salvato nel file /agent/ndom/benchmark/benchmark\_graph.xlsx.

Il grafico mostra come per il Task 1, tutti gli algoritmi esaminano in media lo stesso numero di nodi prima di giungere a un nodo obiettivo. Per il Task 2 la situazione è diversa: potremmo ipotizzare che un nodo obiettivo possa trovarsi a una profondità maggiore, e quindi l’algoritmo BFS perde tempo esplorando l’albero in larghezza. Questo difetto della ricerca BFS non viene però assorbito dalla sua efficienza al Task 7 (circa 40 nodi in meno esaminati rispetto agli altri algoritmi), per cui BFS è da scartare.

Possiamo assumere che l’algoritmo che abbiamo costruito (linea nera) è da considerarsi una migliore alternativa al LCFS, anche a fronte del fatto che può sfruttare una complessità polinomiale di spazio e tempo. Seppur NaiveDOMSearcher impiega due frontiere (una PQ per i percorsi con profondità < 2 e uno stack LIFO), la prima di queste non desta problemi ed ha complessità di tempo trascurabile, in quanto dipende solamente dalla profondità del livello successivo alla radice. E’ improbabile infatti che i template dei siti web dispongano tutti gli elementi come figli diretti del <body>.

# Apprendimento Supervisionato

## Sommario

La rappresentazione tramite modello NDOM discussa nella sezione precedente ci ha permesso, di fatto, di ingegnerizzare e aggiungere al DS iniziale [13 nuove feature](#_Calcolo_di_un). In questa sezione costruiamo e valutiamo dei modelli di apprendimento supervisionato (SL) che possano predire il valore della feature target metric. Si impiegheranno diversi approcci di apprendimento supervisionato, come l’approccio classico, l’approccio Case-Based e l’approccio con metodi Ensemble.

## Strumenti utilizzati e decisioni di progetto

Vedere: /agent/models/nb\_supervised\_learning.ipynb

*Queste due sezioni sono trattate separatamente nel file indicato perché richiedono l’esecuzione di codice.*

## Valutazione

Vedere: /agent/models/charts/charts.xlsx

Per questo task di regressione sono state utilizzate diverse metriche, calcolate sia sui dati di training che sui dati di test.

1. Mean Average Error (MAE): media delle differenze assolute tra le previsioni e i valori reali. A differenza del MSE, il MAE non penalizza tanto gli errori grandi, rendendolo una metrica più robusta alla presenza di valori anomali.
2. Mean Squared Error (MSE): misura la media dei quadrati delle differenze tra i valori previsti e i valori reali. Penalizza in modo più severo gli errori grandi rispetto a quelli piccoli.
3. Root Mean Squared Error (RMSE): Il RMSE è semplicemente la radice quadrata del MSE. Questa metrica è particolarmente utile quando si desidera interpretare l'errore nel contesto delle variabili originali, dato che riporta l'errore alla stessa unità di misura delle variabili stesse.
4. Coefficiente di Determinazione (R2): Questa metrica fornisce una misura di quanto bene le previsioni del modello si adattano ai dati reali. Un R2 di 1 indica che il modello è in grado di prevedere perfettamente i dati, mentre un R2 = 0 indica che il modello non è in grado di prevedere i dati meglio di un modello costante.

Di seguito verranno presentate e commentate le performance dei migliori modelli.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Modello** | **Combinazione ottimale di iperparametri** | **MAE (TS)** | **MSE (TS)** | **RMSE (TS)** | **R^2 (TS)** | **MAE (TE)** | **MSE (TE)** | **RMSE (TE)** | **R^2 (TE)** |
| regLin |  | 0,360450 | 0,198498 | 0,445513 | 0,232727 | 0,367821 | 0,214031 | 0,462326 | 0,167726 |
| regKNN | {'algorithm': 'kd\_tree', 'n\_neighbors': 12, 'weights': 'distance'} | 0,000000 | 0,000000 | 0,000000 | 1,000000 | 0,282082 | 0,150461 | 0,387708 | 0,416453 |
| regDeT | {'criterion': 'friedman\_mse', 'max\_depth': 10, 'min\_samples\_leaf': 5, 'min\_samples\_split': 40, 'random\_state': 1, 'splitter': 'random'} | 0,266001 | 0,130166 | 0,360735 | 0,496544 | 0,283585 | 0,150130 | 0,386581 | 0,416471 |
| regSVR | {'C': 100, 'epsilon': 0.2, 'gamma': 'scale', 'kernel': 'rbf'} | 0,272003 | 0,131184 | 0,362131 | 0,492899 | 0,306309 | 0,175513 | 0,418576 | 0,316725 |
| regRaF | {'bootstrap': True, 'criterion': 'friedman\_mse', 'max\_depth': 10, 'min\_samples\_leaf': 5, 'min\_samples\_split': 2, 'n\_estimators': 100, 'random\_state': 66} | 0,194428 | 0,069630 | 0,263857 | 0,730862 | 0,272786 | 0,137379 | 0,370500 | 0,466306 |
| regGrB | {'criterion': 'friedman\_mse', 'learning\_rate': 0.05, 'max\_depth': 10, 'min\_samples\_leaf': 20, 'min\_samples\_split': 120, 'n\_estimators': 100, 'random\_state': 66} | 0,203744 | 0,075641 | 0,275016 | 0,707577 | 0,275188 | 0,139881 | 0,373839 | 0,455710 |

**regLin**. Il modello ha dimostrato un’accuratezza discreta, con punteggi di usabilità che differiscono in media di 3,6 punti decimali sia sul TS (0,360450) che sul TE (0,367821). E’ un modello sicuramente da scartare perché ha un valore di R2 insufficiente, che implica bassa capacità di generalizzazione. Non a caso i valori di R2, sia sul TS che sul TE, sono i minimi registrati.

**regKNN**. Per il regressore KNN possiamo fare una prima osservazione: siamo certi del fatto che i siti con lo stesso page\_template tendono ad avere una valutazione simile, salvo quando un sito stravolge il suo assetto grafico continuando a utilizzare il template (raro). Se utilizziamo la configurazione con k=12 + esempi simili ponderati sulla distanza vettoriale, possiamo pensare che questo algoritmo sia quello preferibile, e infatti ottiene un MAE di 0 sul TS e 0,282082 sul TE, indicando previsioni accurate. Il MSE è anch'esso molto basso, anche sul TE (0,150461), il che suggerisce una buona gestione delle deviazioni.

**regDeT**. Per questo modello, gli iperparametri migliori sono risultati il criterio di riduzione delle impurità friedman\_mse, una altezza massima di 10, e l’attuazione dello splitting solo se la partizione corrente ha almeno 40 esempi.  
Con un R2 di 0,496544 sul TS e 0,416471 sul TE, questo modello ha mostrato un buon equilibrio tra overfitting e underfitting. Compie in media un errore assoluto di poco più 2,5 punti decimali (0,266001 sul TS e 0,283585 sul TE) e il MSE (0,130166 sul TS e 0,150130 sul TE) indica che il modello è in grado di gestire le deviazioni. Questo ci suggerisce che un modello Ensemble basato su alberi di decisione porterà a prestazioni migliori.

**regSVR**. La migliore configurazione del SVR è un Soft-Margin SVM (visto che C>0) e usa una funzione kernel gaussiana. Risulta essere migliore solo del regressore lineare, visto che compie un errore assoluto pari a 3 punti decimali.

**regRaF** e **regGrB**: Questi due modelli Ensemble basati su alberi di decisione risultano avere prestazioni simili. Sul TS compiono un errore assoluto esiguo di 2 punti decimali mentre sul TE di 2,7 punti decimali, denotando una notevole precisione. Entrambi i modelli hanno presentato le performance più elevate in termini di R2 (0,730862 sul TS e 0,466306 sul TE), mostrando che sono in grado di spiegare una significativa parte della varianza nei dati.

Abbiamo osservato che i coefficienti di determinazione R2 dei modelli si aggirano intorno a 0,45. Una possibile spiegazione risiede nel fatto che i modelli considerano un insieme di feature nessuna delle quali, presa singolarmente, è fortemente esplicativa. Ricordiamoci infatti che la feature page\_ungrouped\_multim è esclusivamente un fattore di decisione sul quale dipende la valutazione del ground truth, ed è stata volutamente esclusa dalle feature considerate (vedi sezione *Caricamento DS e Feature Selection*). Testando il caso opposto comunque, sono state misurate valori di R2 più alti (in media pari a 0,85), ma così facendo staremmo “barando”.  
Il valore di R2 di uno dei nostri modelli indica la proporzione di dispersione che tale modello riesce a spiegare, rispetto a un modello di base che ipotizza sempre il valore medio. Per quanto nei nostri modelli si aggiri intorno a 0,45, il fatto che sia maggiore di 0 è comunque un segno indicante un comportamento più “intelligente” rispetto a un modello baseline “stupido” che predice calcolando il valore medio.

# Modello probabilistico: Rete Bayesiana

## Sommario

Nella fase iniziale di questo progetto, legata alla creazione del Grund Truth e alla scelta di feature da inserire nei dataset, è stato possibile visionare tutti i template disponibili e tutte le Homepage, e pertanto si è potuta intuire l’esistenza di dipendenze tra le varie features. I modelli grafici probabilistici, mediante i task di inferenza probabilistica che è possibile fare su di essi, ci possono dare informazioni utili sull’usabilità dei siti scolastici delle scuole superiori.

## Strumenti utilizzati

Il modello grafico usato in questo progetto è una Rete Bayesiana, cioè un DAG che rappresenta un insieme di variabili aleatorie con le loro dipendenze condizionali. I nodi sono le variabili, gli archi rappresentano la dipendenza condizionata. La probabilità condizionata dei valori assumibili da un nodo (a partire da quelli dei suoi genitori) è espressa in una CPT.  
Tutte le operazioni sulla BN sono state fatte con la libreria Python [pgmpy](https://pgmpy.org/).

## Decisioni di progetto

Vedere: /agent/pgm/bn\_main.py  
 /agent/pgm/bif/bn\_gt.bif

Nelle sezioni seguenti sono elencati in ordine i passi seguiti per la realizzazione della BN e l’inferenza.

### Struttura e parametri della BN

La struttura della BN, ovvero i nodi e gli archi, non è stata appresa dal DS, ma progettata, per i motivi espressi nel Sommario. Le considerazioni sulla struttura sono descritte di seguito.

In generale, quando osserviamo un template (ad es. uno tra quelli in Figura 2), possiamo notare come questo influenza fortemente l’orientamento del menu della pagina e in parte anche l’altezza della pagina stessa. Inoltre, **sarebbe corretto dire che esso contribuisce in maniera differente anche al numero di elementi non raggruppati?** Sì perché ad esempio le pagine con template #8 sono tutte caratterizzate da una sezione non intitolata contenente un numero vario di elementi confusionari (Figura 10 e 11).

A questo punto ci chiediamo: **il numero di elementi raggruppati dipende solamente dal template della pagina?** No perché il webmaster del sito ha pieno controllo sulla disposizione degli elementi. In figura 10 il banner “Circolari” può essere integrato in una voce del menu. Idem in figura 11 dove il banner “Registro Elettronico” può essere integrato nella voce del menu “Studenti e Genitori”.  
Questo porta all’aggiunta nella BN di una variabile latente \_designer\_taste.

|  |  |  |  |
| --- | --- | --- | --- |
| *Variabile latente* | *Descrizione* | *Dominio* | *u.m.* |
| \_designer\_taste | Grado di ordine immesso dal designer/webmaster della pagina. 0=non ottimale 1=ottimale |  |  |



Figura 10. <https://www.patettacairo.edu.it/>



Figura 11. <https://www.alberghierolaspezia.edu.it/>

**Da quali variabili dovrebbe dipendere la feature metric?** Di certo sappiamo l’utente non è a conoscenza del Template ID, quindi non assegna un punteggio che dipende da esso; piuttosto osserva ciò che *dipende* dal template: l’orientamento del menu, altezza della pagina e numero di elementi confusionari.

Venendo ora alle variabili NDOM\_nodes e NDOM\_height, entrambe sono indipendenti (può esistere un NDOM con tanti nodi ed altezza 1 o viceversa). Non vengono influenzate dal template della pagina ma solo da un’astrazione di essa, cioè solamente dall’altezza.

Le variabili page\_load\_time\_ms e page\_width non sono invece incluse nella BN perché si sono assunte indipendenti da tutte le altre.



Figura 12. In viola le cardinalità delle variabili.

Per l’apprendimento dei parametri della BN è stato utilizzato l’algoritmo EM, in quanto esiste la variabile latente \_designer\_taste. A grandi linee, l'algoritmo EM opera in due fasi, E-step e M-step. Nell'E-step, viene calcolata l'aspettazione del logaritmo della verosimiglianza, considerando la distribuzione corrente dei parametri. Successivamente, nella M-step, i parametri vengono aggiornati per massimizzare questa aspettazione. Questo processo viene ripetuto finché non si raggiunge la convergenza, ovvero quando i parametri non cambiano significativamente tra un'iterazione e l'altra. Per quanto riguarda la stima della CPT della variabile latente, l'algoritmo utilizza un approccio di massima verosimiglianza, scegliendo i parametri che massimizzano la probabilità dei dati osservati, data la distribuzione corrente dei parametri.

### Inferenza probabilistica: query

Per quanto riguarda i task di inferenza probabilistica (per rispondere alle query sottoposte) si è deciso di utilizzare un metodo di inferenza approssimato come il Forward Sampling in cui la grandezza del campione è uguale al numero di righe del dataset. Avremmo potuto anche utilizzare un metodo di inferenza esatta, anche a fronte del numero limitato di righe, tuttavia si è osservato che la differenza tra le predizioni è irrisoria.

Le query sottoposte alla BN sono elencate nel dizionario BN\_QUERIES\_DEFAULT, in cui oltre alla variabile query e alle variabili di evidence è presente la descrizione formale simbolica (in chiave query\_desc)

1. **Descrizione**: Qual è la probabilità che la pagina abbia un determinato template sapendo che ha ottenuto un punteggio compreso tra 4 e 5?  
   **In simboli**:   
   Risultato:
2. **Descrizione**: Qual è la probabilità che la pagina abbia un determinato template sapendo che ha ottenuto un punteggio compreso tra 4 e 5?  
   **In simboli**:   
   Risultato:
3. Probabilità

## Valutazione

Dd

# Conclusioni

Un paragrafo che riassuma le valutazioni e delinei possibili sviluppi, ad. es. problematiche non affrontate per questioni di tempo (per eventuali estensioni da parte di altri gruppi).

# Bibliografia

|  |  |
| --- | --- |
| [1] | [Online]. Available: https://en.wikipedia.org/wiki/Heuristic\_evaluation#Nielsen's\_heuristics. |
| [2] | [Online]. Available: https://www.w3.org/TR/WCAG21/. |
| [3] | [Online]. Available: https://en.wikipedia.org/wiki/System\_usability\_scale. |
| [4] | [Online]. Available: https://en.wikipedia.org/wiki/XPath. |