

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](#_Toc150282145)

[Gruppo di lavoro 1](#_Toc150282146)

[Repository 1](#_Toc150282147)

[Sommario 2](#_Toc150282148)

[Introduzione 3](#_Toc150282149)

[Idea del progetto 3](#_Toc150282150)

[Metriche di usabilità già esistenti 3](#_Toc150282151)

[Metrica di usabilità adottata per questo progetto 3](#_Toc150282152)

[Elenco argomenti di interesse 4](#_Toc150282153)

[Costruzione del ground truth 5](#_Toc150282154)

[Sommario 5](#_Toc150282155)

[Decisioni di progetto 5](#_Toc150282156)

[Preprocessing del dataset delle scuole 7](#_Toc150282157)

[Rappresentazione dello spazio di ricerca con grafo e ricerca soluzioni 8](#_Toc150282158)

[Sommario 8](#_Toc150282159)

[Strumenti utilizzati: modello NaiveDOM 8](#_Toc150282160)

[Decisioni di Progetto 11](#_Toc150282161)

[Valutazione 11](#_Toc150282162)

[Sezione argomento 1 12](#_Toc150282163)

[Sommario 12](#_Toc150282164)

[Strumenti utilizzati 12](#_Toc150282165)

[Decisioni di Progetto 12](#_Toc150282166)

[Valutazione 12](#_Toc150282167)

[Conclusioni 13](#_Toc150282168)

[Bibliografia 14](#_Toc150282169)

# 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 (aggiornato al 09/2023).

## 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. **Dobbiamo precisare che questa metrica è vista come un qualcosa di condiviso dai visitatori (concetto oggettivo), infatti in questo progetto non si parla di valutazione (soggettiva) che uno specifico utente dà alla pagina**. Approfondiremo questa metrica nelle sezioni successive.

# Elenco argomenti di interesse

Indichiamo le 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**.
   2. Costruzione e valutazione di **Modelli di apprendimento supervisionato** che, a partire dalle feature per ciascun sito (individuate al punto 2.1) simulano la sua valutazione.
3. Deduzione di informazioni utili e statistiche.
   1. Costruzione + Valutazione di modelli di apprendimento non supervisionato.
   2. Operazioni di deduzione con ragionamento relazionale e basi di conoscenza.

# 1. Costruzione del ground truth

## Sommario

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

* [1, 1.9]: **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, 2.9]: **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, 3.9]: **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**.

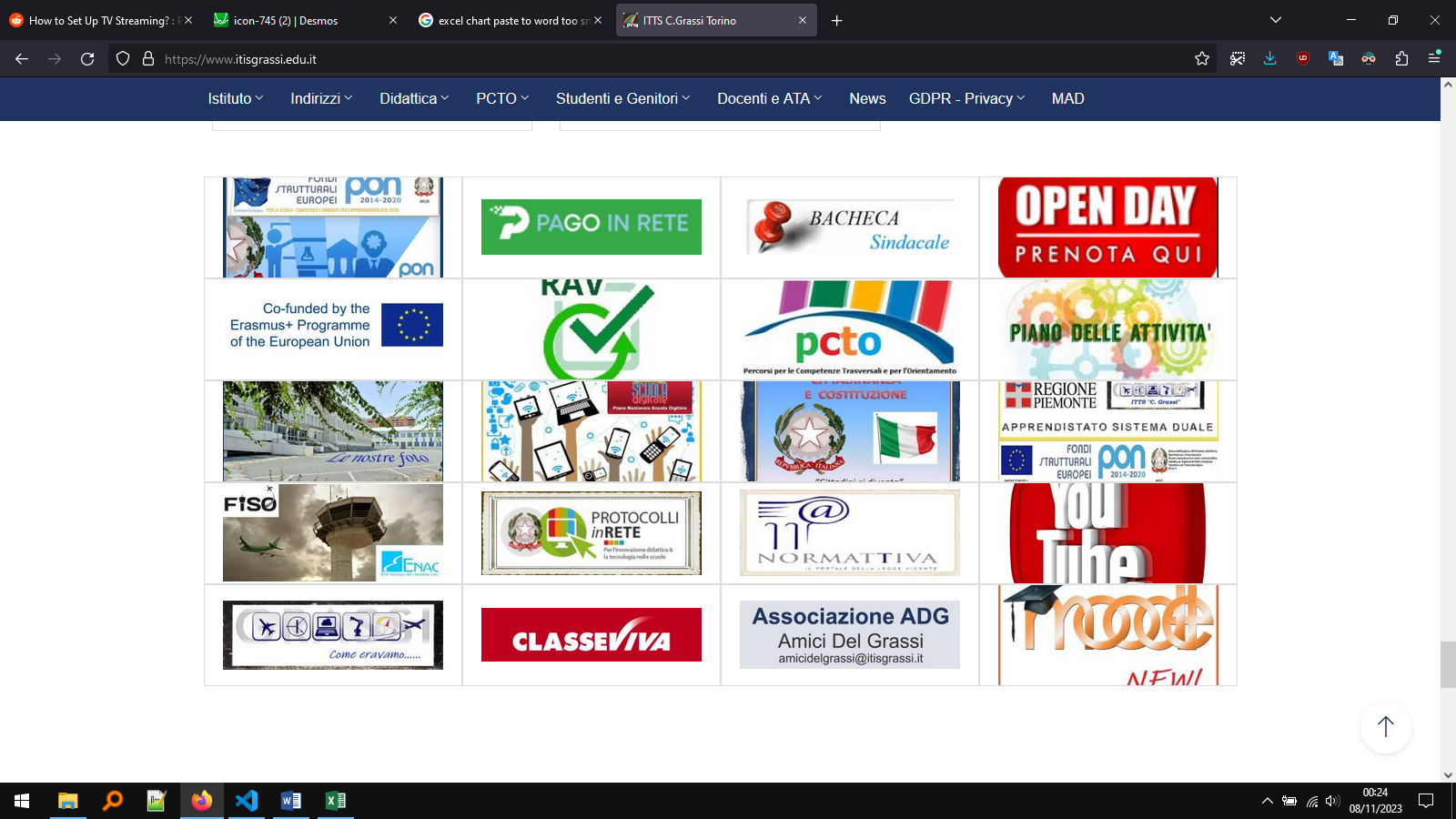


Figura 1. Esempi di banner.

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.

Riassumendo, i fattori di decisione sono i seguenti:

|  |  |  |  |
| --- | --- | --- | --- |
| *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. |  |  |

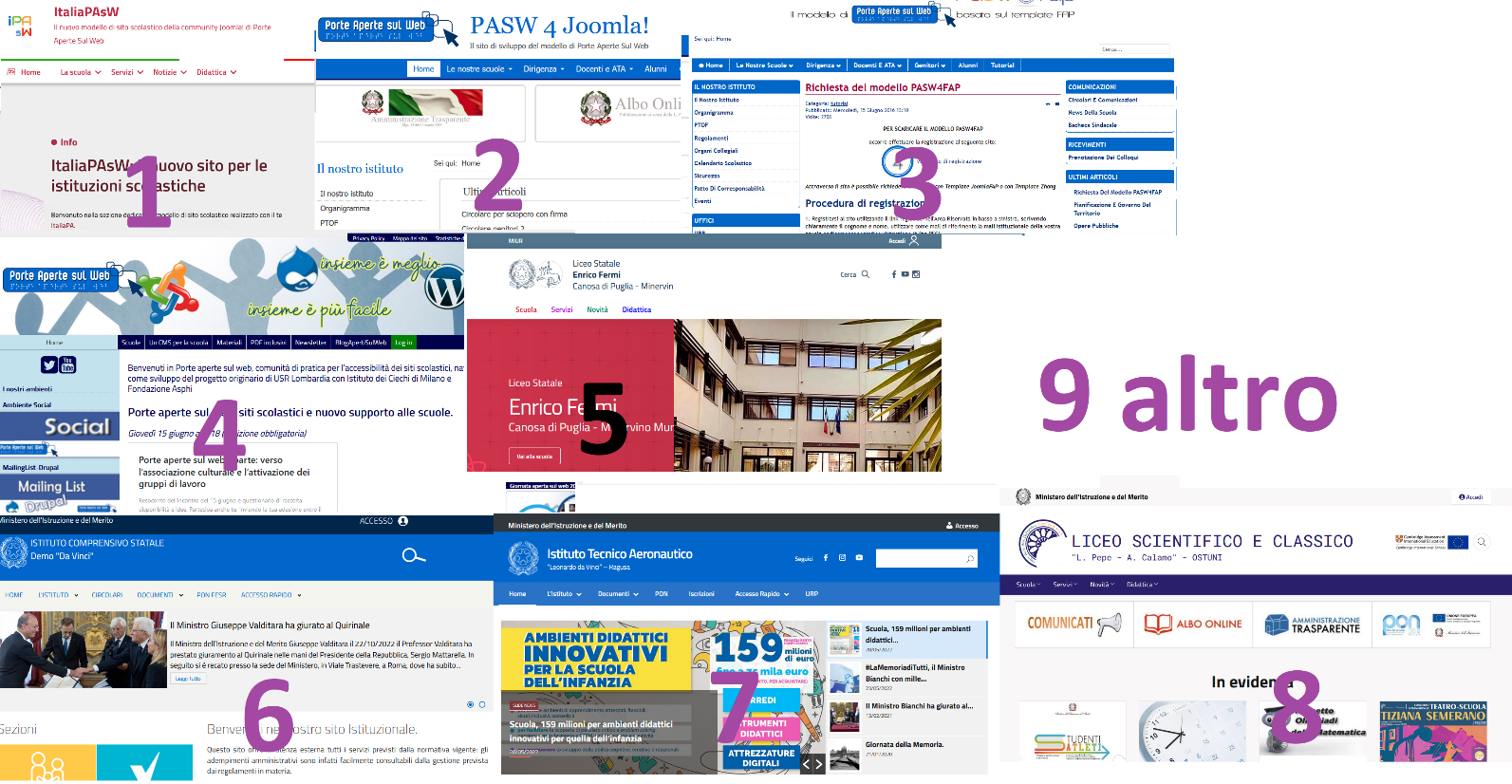


Figura 2. Template tipici dei siti web scolastici italiani.

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/>

## 1.1 Preprocessing del dataset delle scuole

Vedi: /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 va 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 [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 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 nella prossima sezione.
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 (elementari, …)
* (accanto a *unique*) Siti duplicati.
* (accanto a *final*) Siti raggiungibili ma non validi.
* Siti delle scuole superiori raggiungibili e validi.

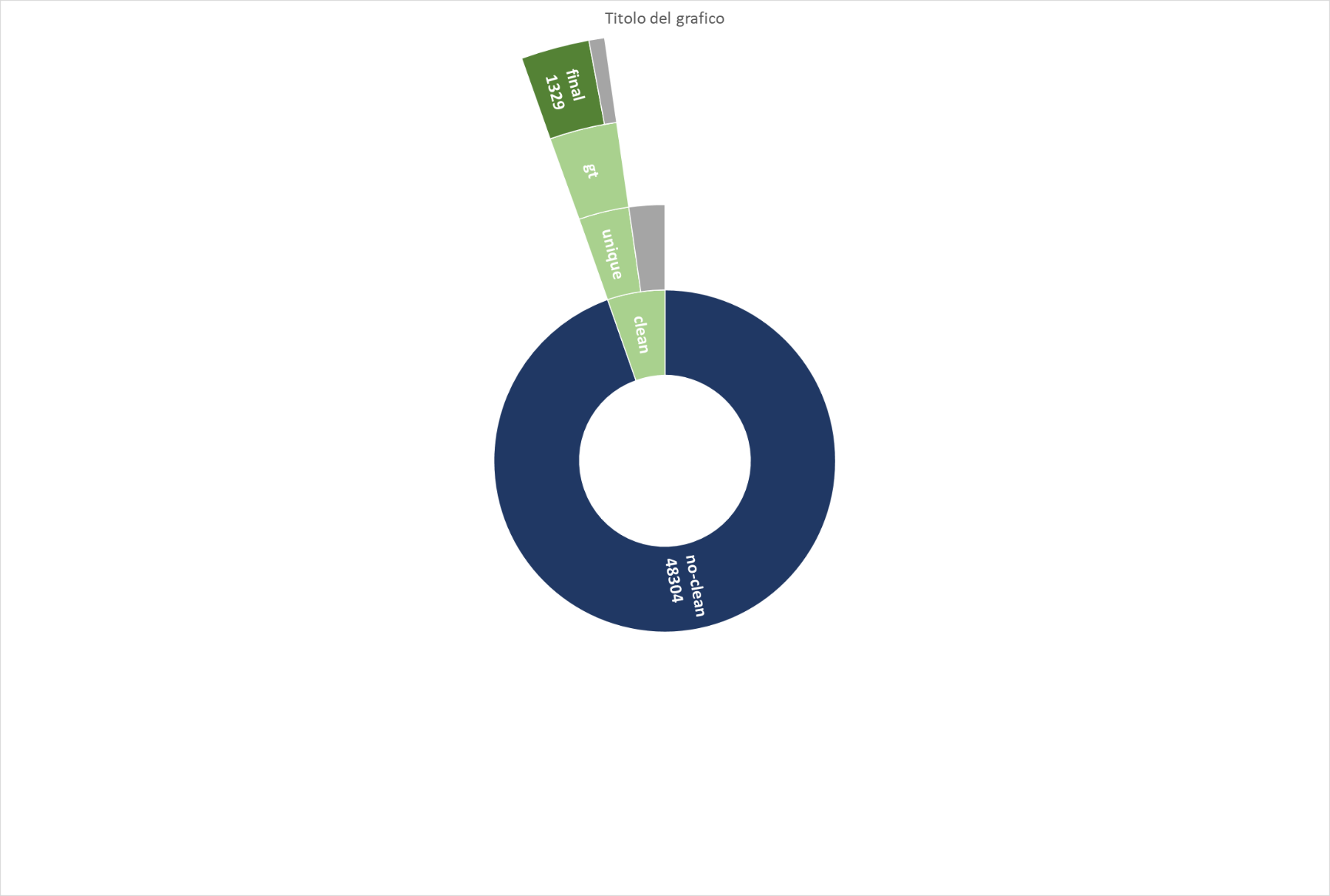


Figura 3. Proporzione del numero di elementi dei vari DS.

# 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

Vedi: /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.

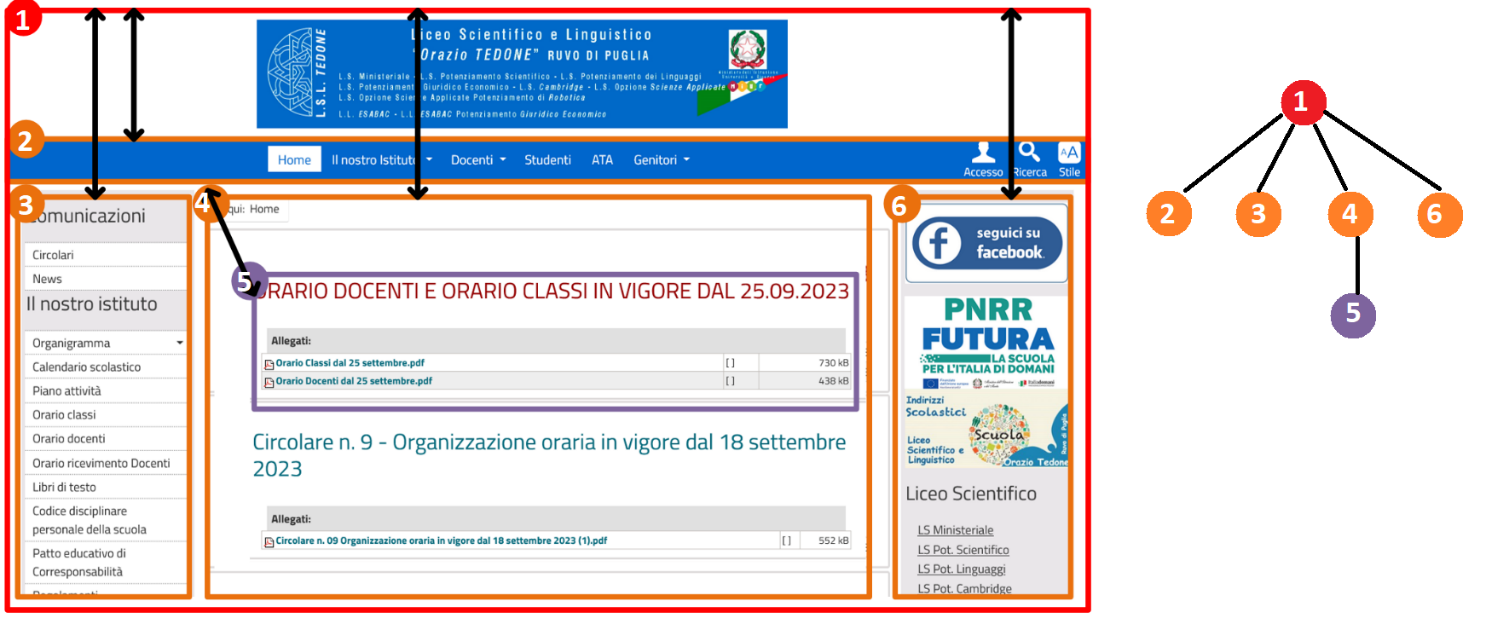


Figura 4. Pagina web <https://www.liceotedone.edu.it/> e rappresentazione stilizzata del NDOM.

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.

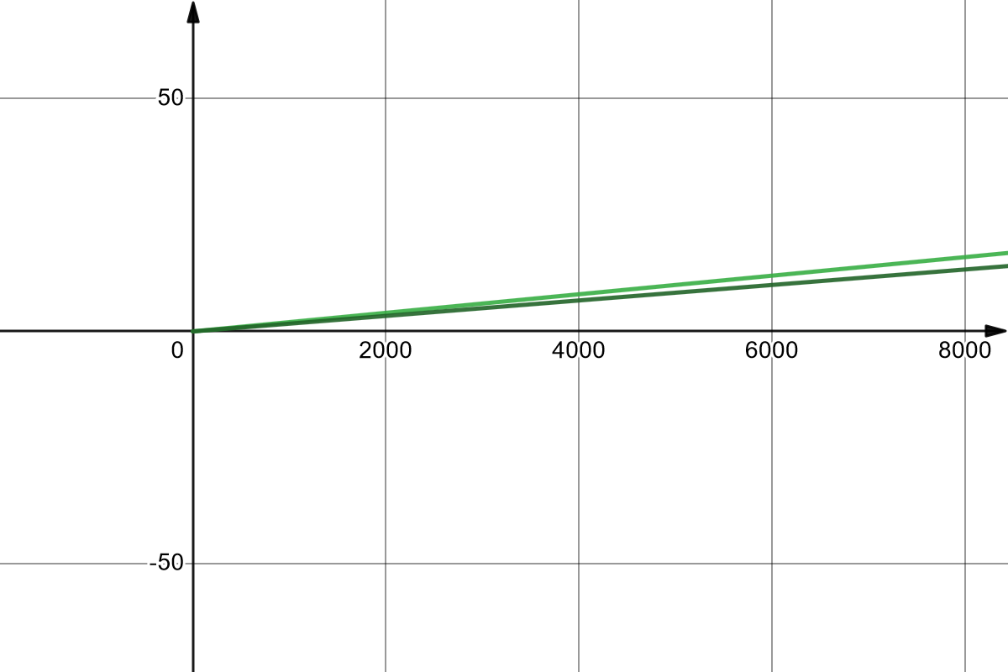


Figura 5. Grafico funzione di costo dell’arco padre-figlio. <https://www.desmos.com/calculator/aaqy3tao8g>

### Calcolo di un task con algoritmo di ricerca

Vedi /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.

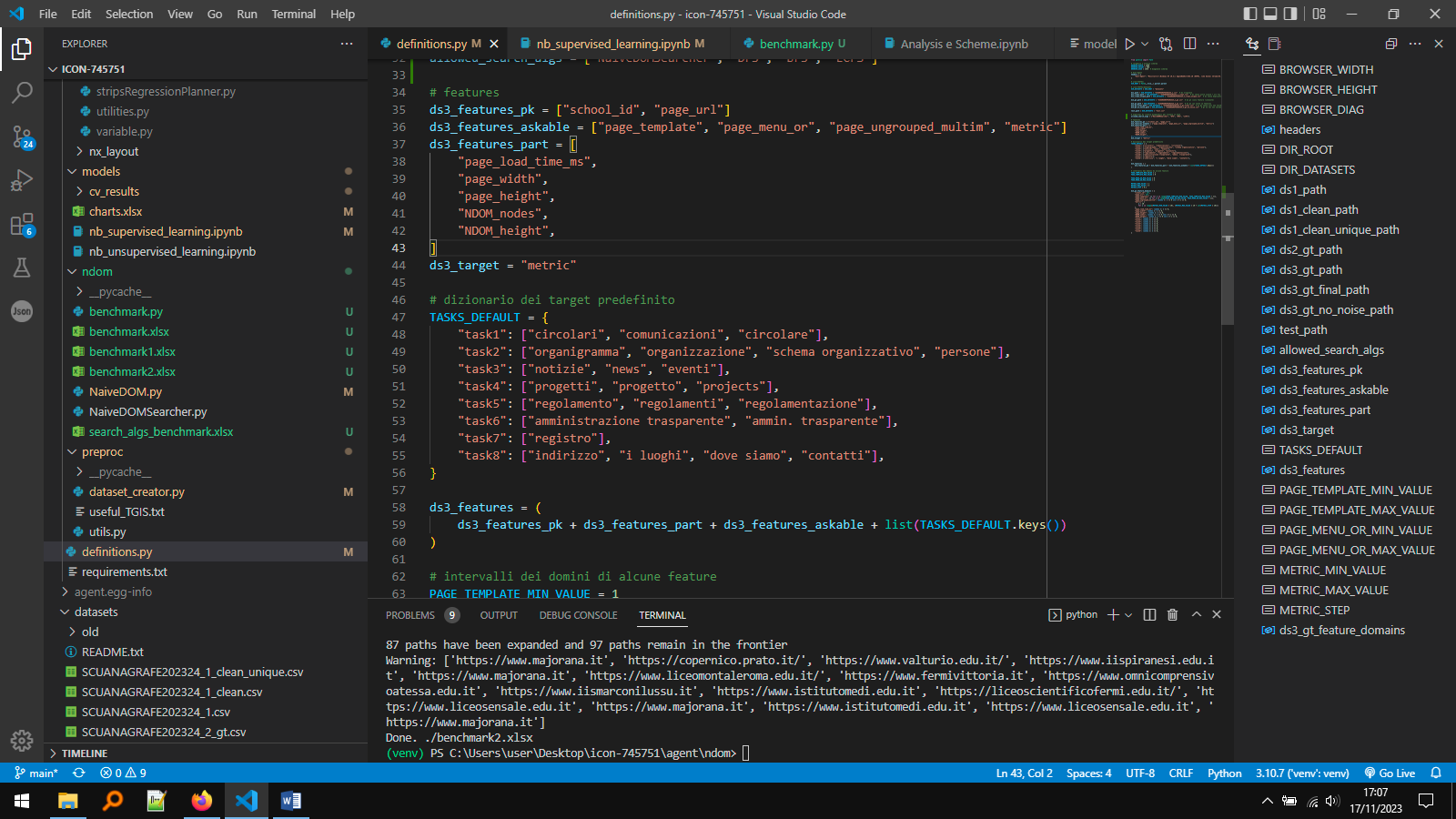
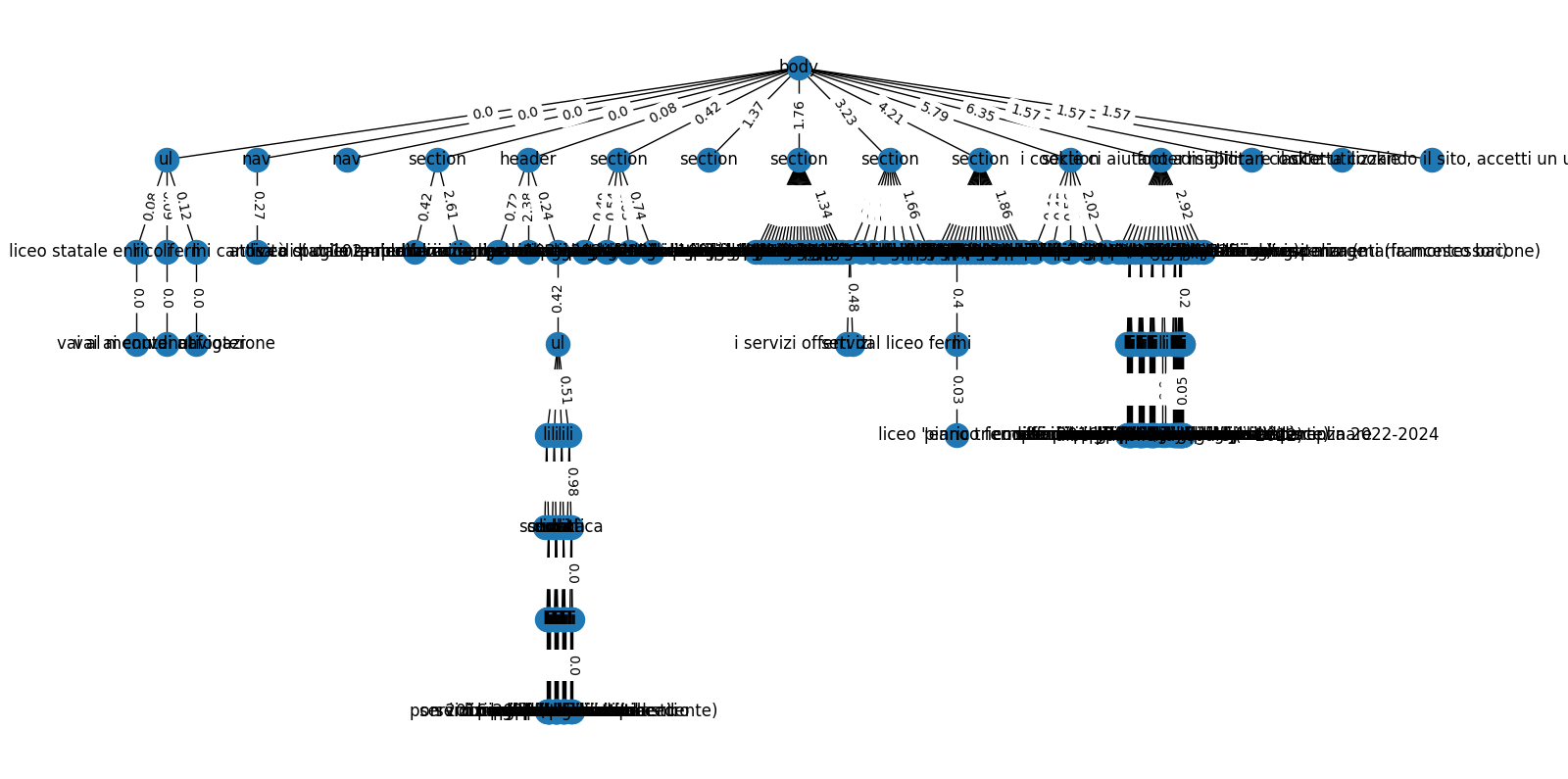


Figura 6.

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.

****

**DFS**

Minore per prima

Figura 7. Risultato del metodo plot() applicato al 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 esempio OCR), per cui il costo di default è di media entità, risultante dal compromesso tra la prima e la seconda causa.

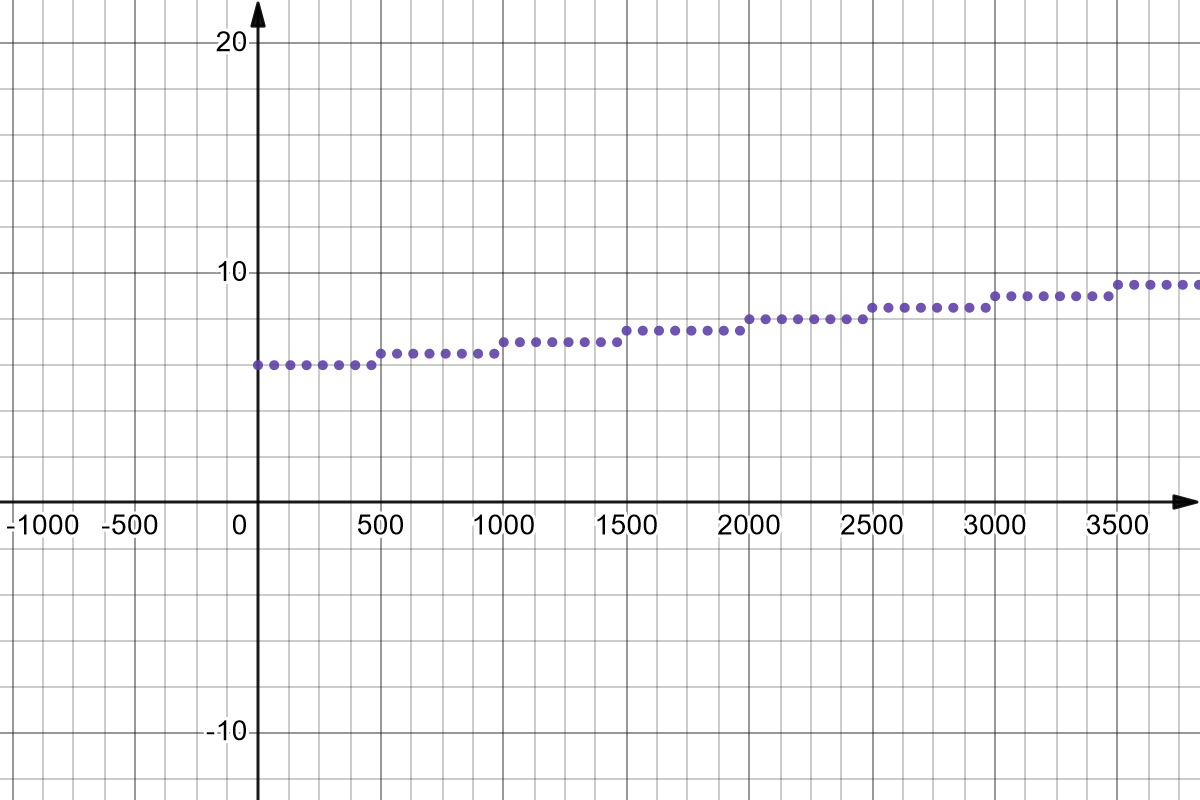
****

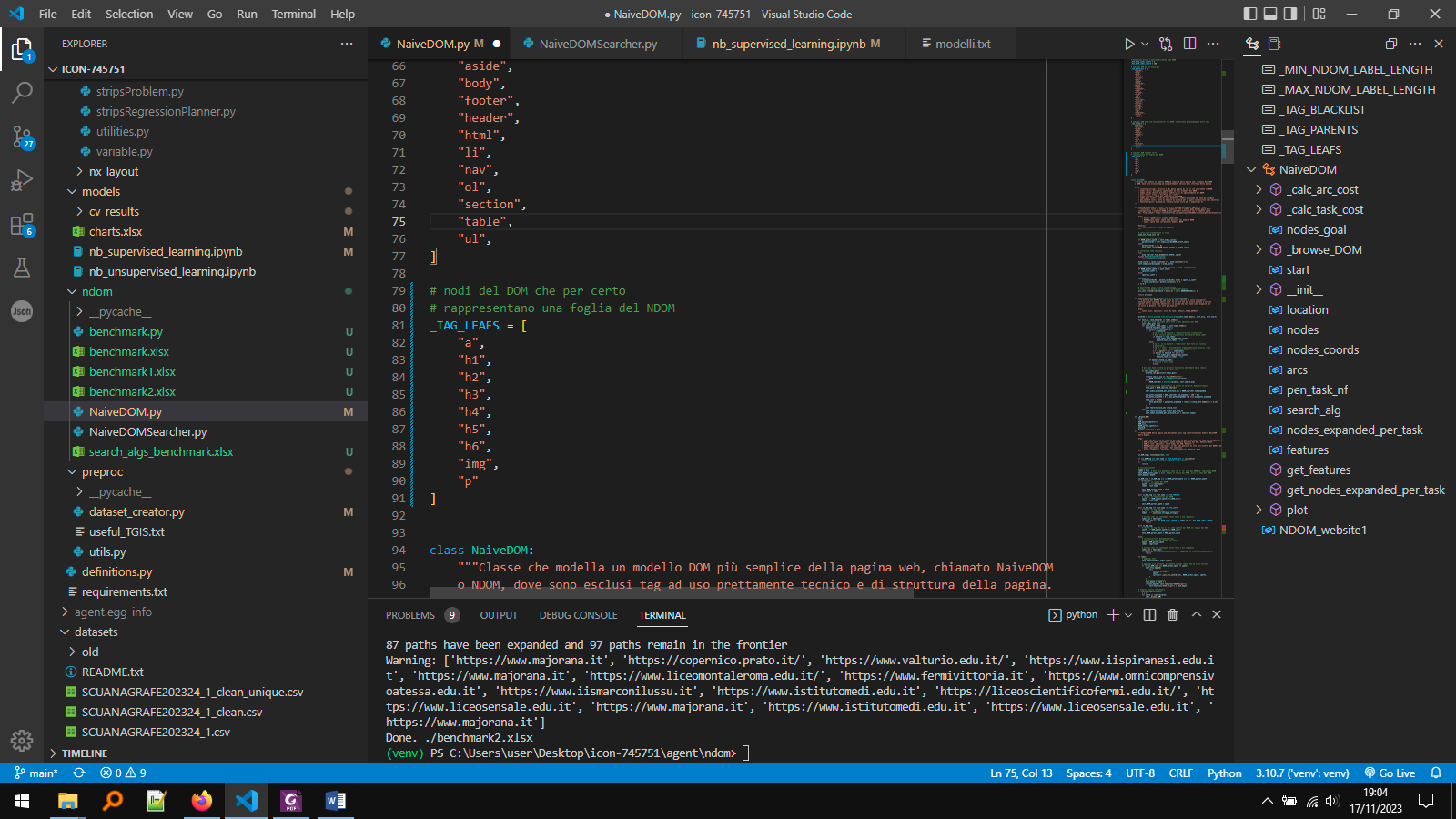
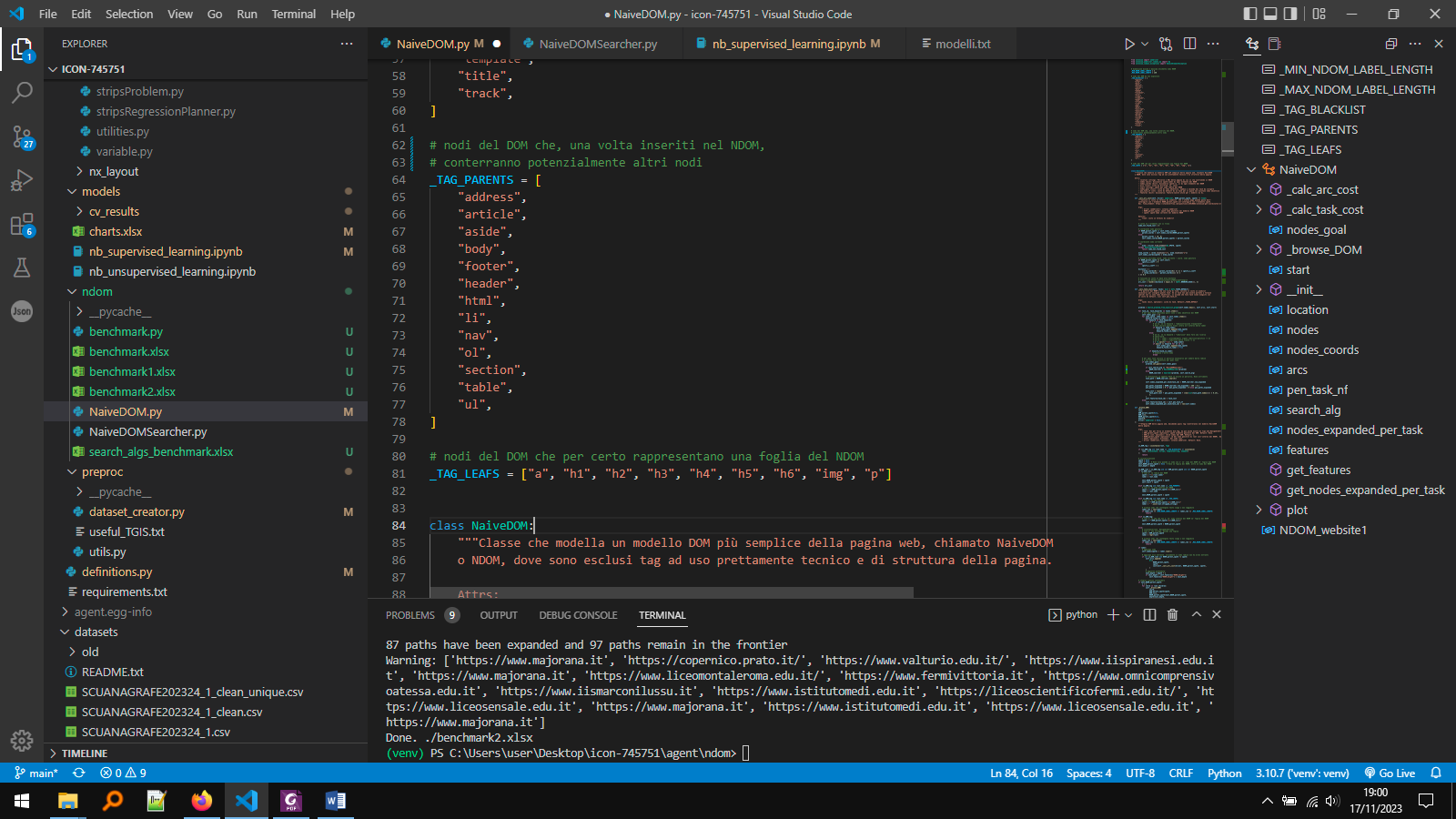
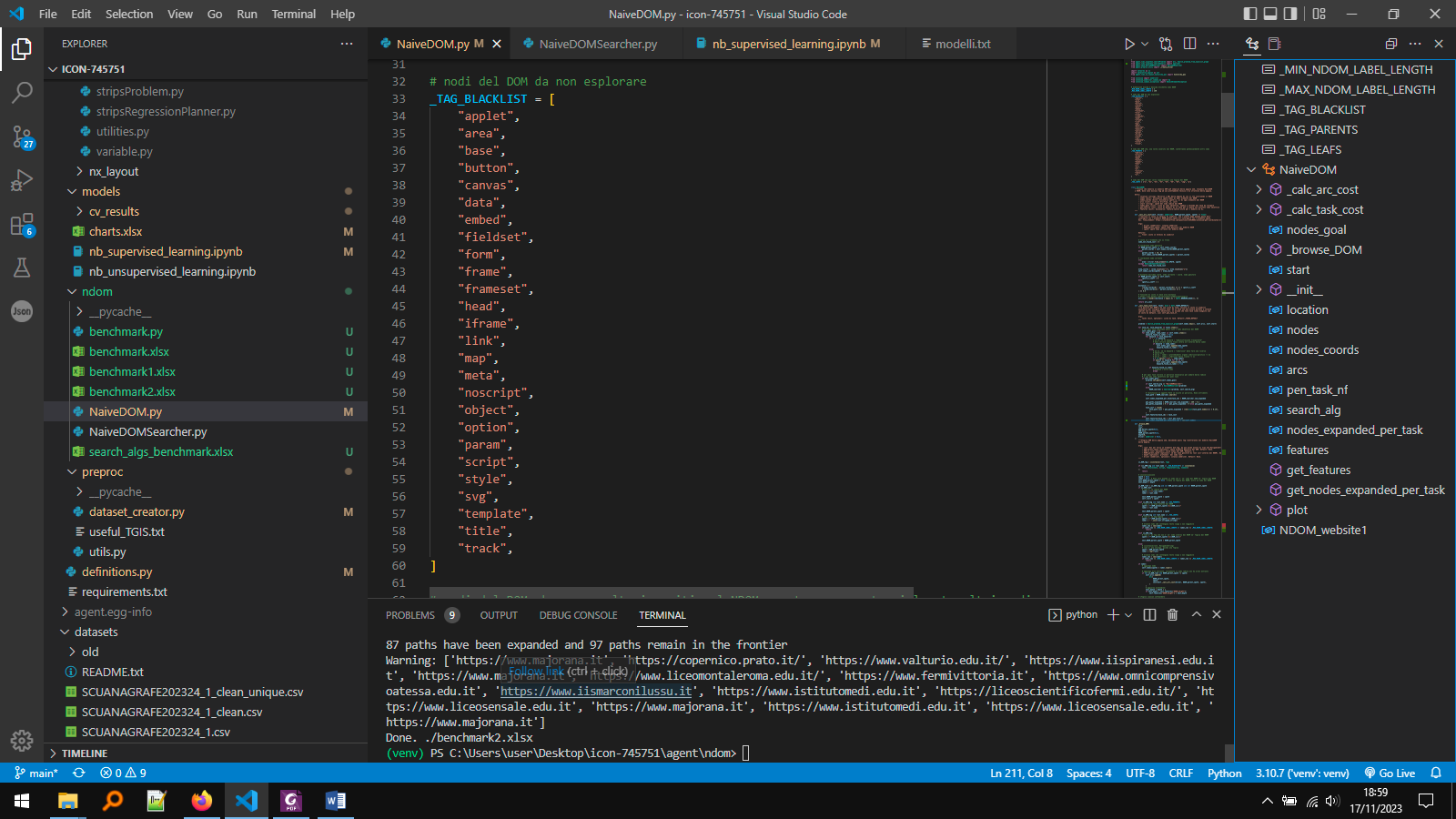
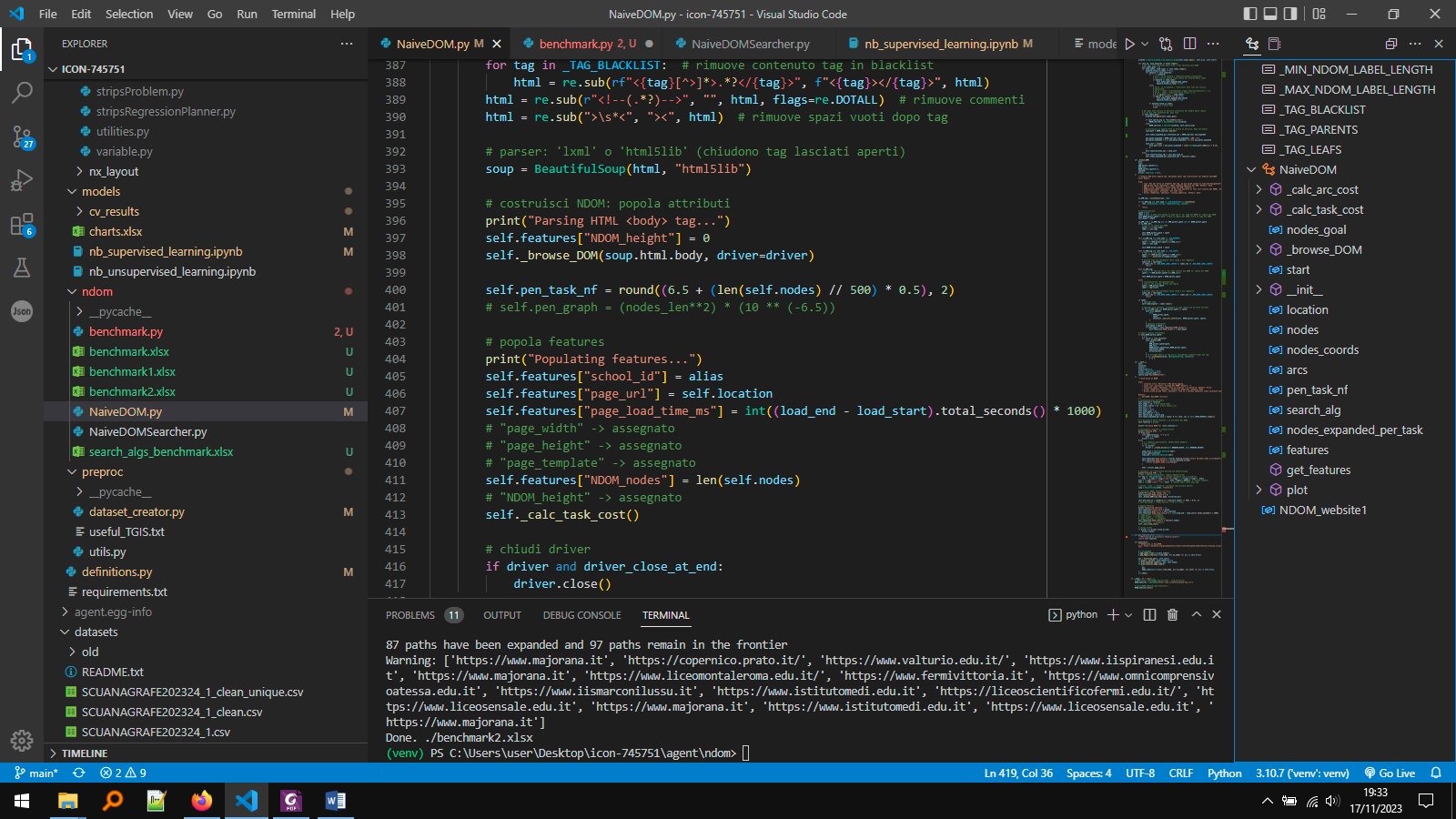
Figura 8. Grafico funzione .  
<https://www.desmos.com/calculator/2epakrbyrj>

## 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.  
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 inerenti ai nodi foglia e ai nodi interni.

Come possiamo vedere dall’interfaccia, 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 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’ quindi 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

Vedi /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.

# 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

*Queste due sezioni sono trattate nel file* **agent/models/nb\_supervised\_learning.ipynb** *in quanto verranno fatte varie osservazioni che dipendono dall’output del codice.*

## Valutazione

Una volta individuate le migliori configurazioni di iperparametri per i modelli, li valutiamo tutti sulle stesse metriche (diverse da quelle usate nei task di classificazione):

* Mean Absolute Error (MAE): rappresenta la media degli errori assoluti tra le previsioni del modello e i valori effettivi, in qualche modo analoga alla precisione nei task di classificazione.
* Mean Squared Error (MSE): calcola la media dei quadrati degli errori tra le previsioni del modello e i valori effettivi, attribuendo maggior peso agli errori più grandi.
* RMSE (Root Mean Squared Error): fornisce una stima dell'errore medio tra le previsioni del modello e i valori reali, esprimendo l'errore nella stessa unità della variabile di risposta.
* R2: misura la proporzione della varianza nella variabile dipendente che può essere prevista dalle variabili indipendenti del modello. R2 assume valori compresi tra 0 e 1, dove 1 indica una perfetta adattabilità del modello ai dati.

In figura 7 mostriamo i risultati.

Figura 9. Performance dei vari modelli.

# 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. |