Hansberger

Topological Data Analysis Web Application

Matteo Belenchia & Sebastiano Verdolini

July 6, 2019

Contents

1 Introduzione - Scopo del progetto

HansBerger è una applicazione web locale basata sul framework Python per il web Django. HansBerger implementa una pipeline automatizzata che semplifica l'Analisi dei Dati Topologica (TDA - Topological Data Analysis) fornendo un' interfaccia semplice da utilizzare per una serie di operazioni di analisi e visualizzazione dei dati.

Le funzionalità fornite sono:

- Visualizzazione di datasets in formato CSV, testuale o EDF;
- Supporto al caricamento di matrici precomputate;
- Vietoris-Rips Filtration con Ripser;
- Clique Weight Rank Filtration con Ripser;
- Algoritmo Mapper con Kepler Mapper;
- Calcolo del valore di entropia di un diagramma di persistenza;
- Calcolo della distanza di bottleneck tra diagrammi di persistenza con Persim;
- Salvataggio dei risultati su database relazionale.

Il nome dell'applicazione è una dedica all'omonimo medico tedesco, noto per aver effettuato la prima registrazione delle onde cerebrali e per aver inventato l'elettroencefalogramma. Berger operò in molti ambiti di ricerca: neurologia, circolazione cerebrale, psicologia. Il suo principale contributo alla medicina è stata la ricerca sull'attività elettrica del cervello e la scoperta dell'EEG. La prima registrazione delle onde cerebrali umane, effettuata da Berger, è avvenuta attraverso uno strumento da lui stesso chiamato elettroencefalografo, che riportava una rappresentazione grafica, l'elettroencefalogramma.

1.1 Analisi di Dati Topologica

L'Analisi di Dati Topologica è un approccio all'analisi di dataset utilizzando tecniche topologiche. L'estrazione di informazioni da dataset ad alta dimensionalità, incompleti e soggetti a rumore è generalmente complicato, ma la TDA offre un framework generale per analizzare questi dati in una maniera non suscettibile alla particolare metrica scelta e che permette una riduzione della dimensionalità e resistenza al rumore. Inoltre, la TDA eredità dalla topologia la proprietà della funtorialità (ossia la capacità di passare dalla categoria degli spazi topologici a quella dei gruppi abeliani), che permette di adattarla a nuovi strumenti matematici. La TDA permette:[1]

- Una rappresentazione matematica compressa di un dataset. È possibile studiare la struttura globale di un dataset, fino ai dettagli di un singolo punto, senza sovraccaricarsi cognitivamente.
- Resistenza a rumore e dati mancanti. La TDA riesce a mantenere le caratteristiche significative dei dati.
- Invarianza. L'unica cosa che conta per la TDA è la correttezza, mentre l'inclinazione, la dimensione o l'orientamento dei dati non influiscono significativamente.
- Uno strumento di esplorazione dei dati.
- Una metodologia di studio della forma dei dati e delle varietà. La TDA ha una base teorica solida e gode di funtorialità.

1.2 Omologia persistente[2]

1.2.1 Nuvole di punti

Un dataset è molto spesso rappresentato come una sequenza non ordinata di punti in uno spazio Euclideo n-dimensionale. Questa collezione di punti nello spazio $\mathbb{E}^n$ viene comunemente chiamata `nuvola di punti` o `point cloud data`. Per convertire una nuvola di punti in un singolo oggetto è naturale utilizzare i punti come vertici di un grafo e aggiungere archi sulla base della prossimità tra essi. Due vertici vengono collegati da un arco se risiedono entro una certa distanza ϵ.

[1] `https://kepler-mapper.scikit-tda.org/theory.html`

[2] `https://www.math.upenn.edu/~ghrist/preprints/barcodes.pdf`

1.2.2 Complessi simpliciali

Un grafo del genere riesce a catturare la connettività tra i dati ma ignora una serie di altre peculiarità, che possono però essere catturate immaginando il grafo come una base per oggetti di dimensioni superiori. Infatti il grafo viene trasformato in un complesso simpliciale, uno spazio assemblato utilizzando dei pezzi semplici (i simplessi) identificati combinatoriamente.

Un simplesso di dimensione n è un politopo n-dimensionale col minor numero di vertici, quindi per $n = 0$ è un punto, $n = 1$ un segmento, $n = 2$ un triangolo e $n = 3$ un tetraedro. Generalmente un simplesso n-dimensionale ha $n + 1$ vertici.

Esistono due modi per costruire un complesso simpliciale, risultanti in un complesso di Čech o in un complesso di Rips (o Vietoris-Rips). Per i nostri scopi siamo interessati a quest'ultimo tipo di complesso che è definito come il complesso simpliciale astratto tale che i suoi k-simplessi corrispondono a $(k+1)$-tuple di punti $\{x_\alpha\}_0^k$ (definiti nello spazio $\mathbb{E}^n$) che sono a due a due entro una distanza ϵ. I complessi di Rips sono meno costosi computazionalmente malgrado utilizzino più simplessi dei corrispondenti complessi di Čech. Ciò è dovuto dal fatto che il complesso di Rips è un `flag complex`, ovvero non ha simplessi vuoti ed è massimale tra tutti i complessi dato l'1-scheletro. In questo modo il calcolo combinatorio sull' 1-scheletro determina completamente il complesso, il quale può essere salvato come un grafo e ricostruito con minor costo di un complesso di Čech.

1.2.3 Persistenza

Resta ancora da definire un valore per il parametro ϵ. Con valori di ϵ bassi otteniamo un insieme discreto, mentre con valori alti otteniamo un unico simplesso ad alta dimensionalità. Non è chiaro, fissato un valore di ϵ quali buchi siano importanti e quali invece possono essere ignorati. Lo strumento topologico per lo studio dei buchi a dimensionalità alte è l'omologia, per la sua facilità di computazione e risoluzione topologica, ma non ci fornisce alcun aiuto in questo contesto: per l'omologia un buco è un buco, a prescindere dalla sua importanza.

La risposta a questo problema è stata data dalla persistenza, introdotta da Edelsbrunner, Letscher e Zomorodian[ELZ02]. Nella persistenza, data una famiglia parametrizzata di spazi, le caratteristiche topologiche che persistono in una significativa serie di valori dei parametri sono considerate un segnale, mentre le caratteristiche con vita breve sono considerate rumore. Un `complesso di persistenza` è una sequenza di complessi di catene $C = (C_*^i)_i$ assieme a mappe $x : C_*^i \longrightarrow C_*^{i+1}$. In questo modo abbiamo una sequenza

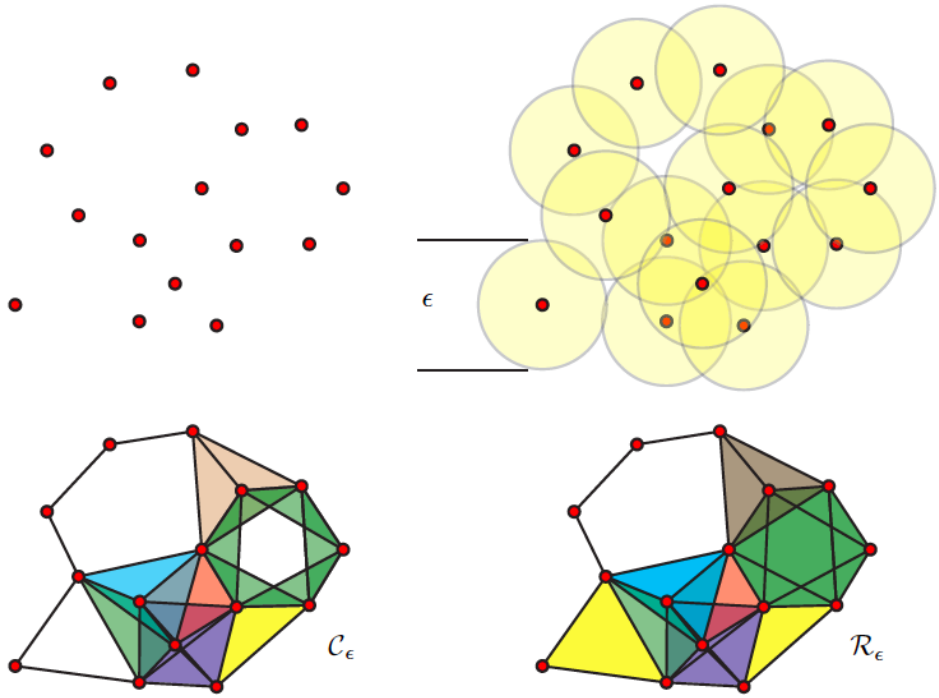

Figure 1.1: Un insieme di punti (in alto a sinistra) viene trasformato in un complesso di Čech (in basso a sinistra) o un complesso di Rips (in basso a destra) basandosi su un parametro di prossimità ϵ (in alto a destra).

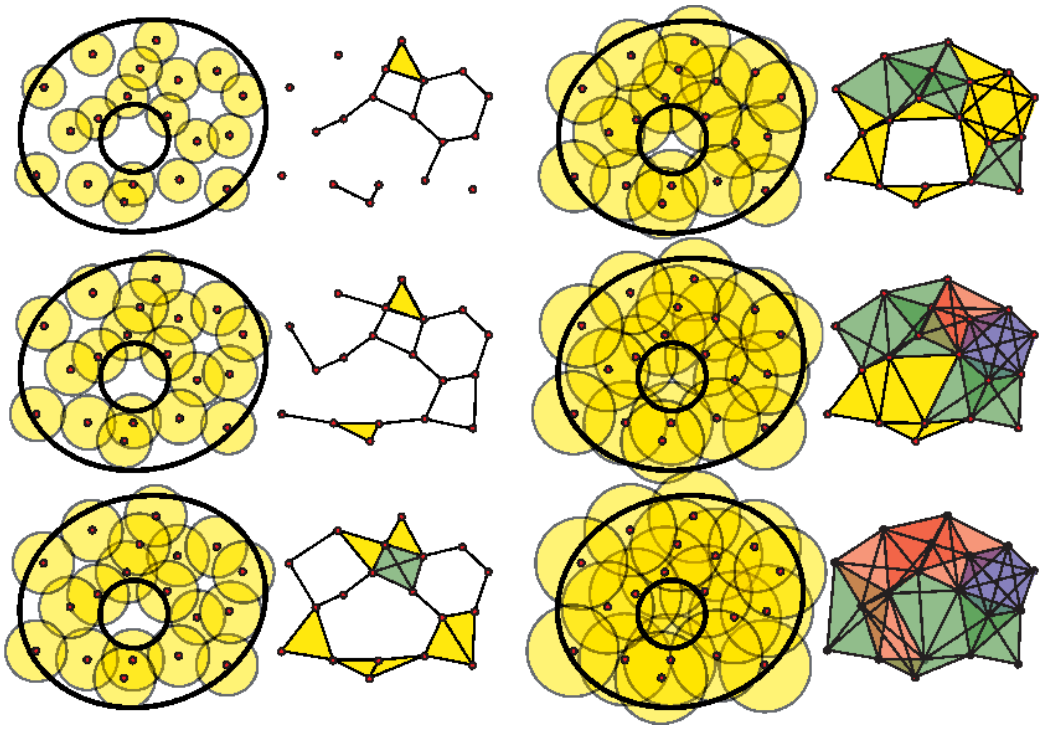

Figure 1.2: Una sequenza di complessi di Rips su una nuvola di punti rappresentante un annulus. Incrementando il valore di ϵ, buchi appaiono e scompaiono.

di complessi di Rips (nota come `Vietoris-Rips Filtration`) con valori crescenti di ϵ campionati ad una crescente sequenza di parametri ϵ_i. Dato che i complessi di Rips crescono insieme a ϵ, le mappe sono naturalmente identificate come inclusioni. Quindi possiamo definire l'omologia persistente:

Definition. *Per $i < j$ la (i, j)-omologia persistente di C, $H_*^{i \to j}(C)$ è definita come l'immagine dell'omomorfismo indotto $x_* : H_*(C_*^i) \to H_*(C_*^j)$.*

1.2.4 Barcode e diagrammi di persistenza

Una rappresentazione grafica dell'omologia persistente è data dal `barcode` come una collezione di segmenti orizzontali su un piano il cui asse orizzontale corrisponde al parametro ϵ e l'asse verticale corrisponde ad un ordinamento arbitrario di generatori di omologia. In un barcode, il momento di inizio di una barra è detto nascita, mentre il momento di fine è detto morte.

Una rappresentazione grafica alternativa è quella dei diagrammi di persistenza, dove ogni generatore di omologia è rappresentato da un punto nel diagramma. L'asse orizzontale rappresenta il tempo di nascita mentre quello verticale di morte, e una retta tratteggiata nella parte superiore rappresenta l'infinito. Una retta viene anche tratteggiata sulla diagonale, che nessun punto può attraversare.

1.2.5 Distanza di bottleneck[3]

Nello spazio dei diagrammi di persistenza esiste una metrica d_B chiamata distanza di bottleneck che misura la somiglianza tra due diagrammi. Essa è definita come:

$$d_B(Dgm_p(f), Dgm_p(g)) = \inf_{\eta} \sup_{x} ||x - \eta(x)||_\infty$$

dove $Dgm_p(f)$ e $Dgm_p(g)$ sono diagrammi di persistenza e η e una biiezione tra i due diagrammi. La distanza di bottleneck prende l'estremo inferiore tra tutti gli estremi superiori dell'insieme delle L_∞-distanze tra punti messi in corrispondenza, includendo anche tutti i punti sulla diagonale dato che i due diagrammi potrebbero avere un numero diverso di punti.

1.2.6 Clique Weight Rank Filtration

Una Clique Weight Rank Filtration viene utilizzata per scoprire proprietà derivate dalla topologia e struttura pesata di reti pesate. Una n-cricca è un

[3] `https://www.maths.ed.ac.uk/~v1ranick/papers/edelhare.pdf`

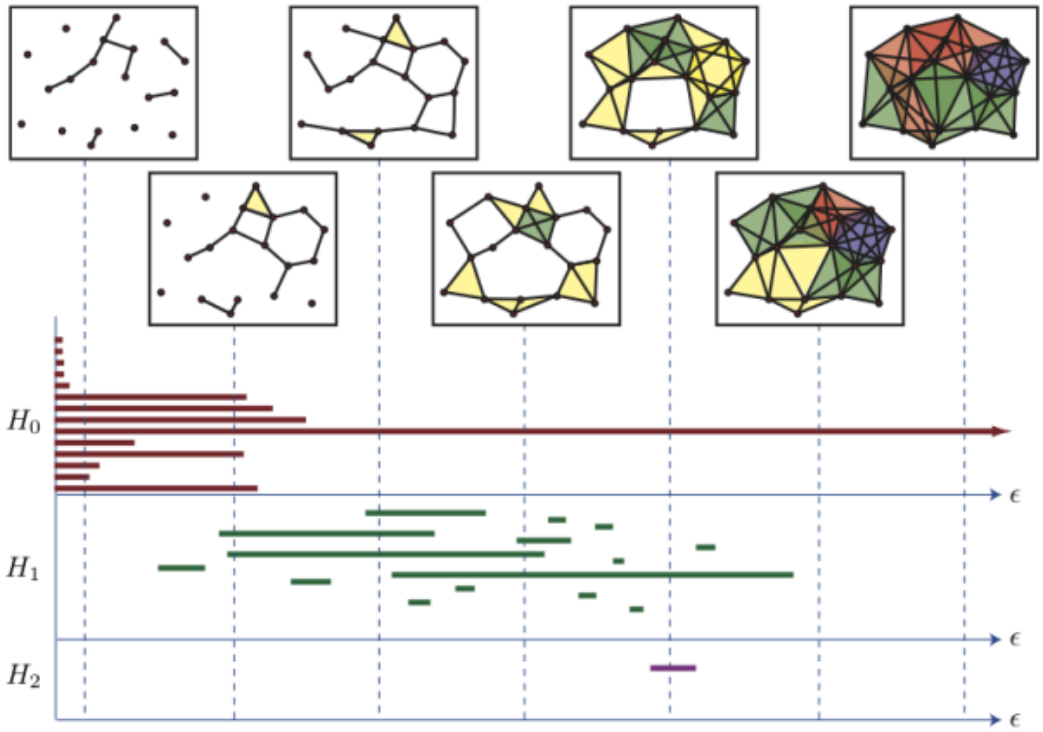

Figure 1.3: Un barcode per $H_*(R)$ sul complesso di Rips dell'annulus. Nell'immagine sopra possiamo vedere come nell' omologia 0 ogni barra corrisponde a una componente connessa e come, col variare di ϵ componenti connesse nascono tutte col valore di ϵ minimo (come singoli punti) e poi successivamente convergono in un unica componente connessa visibile dal terzo complesso simpliciale. Nell' omologia 1, ogni barra corrisponde invece a un buco 1-dimensionale, ossia una regione di spazio compresa tra simplessi 1-dimensionali. Possiamo vedere come questi generatori di omologia nascono in vari momenti ed eventualmente scompaiono nuovamente per valori alti di ϵ. Infine, l'omologia 2 considera buchi compresi tra simplessi 2-dimensionali, ma non sono chiaramente visibili.

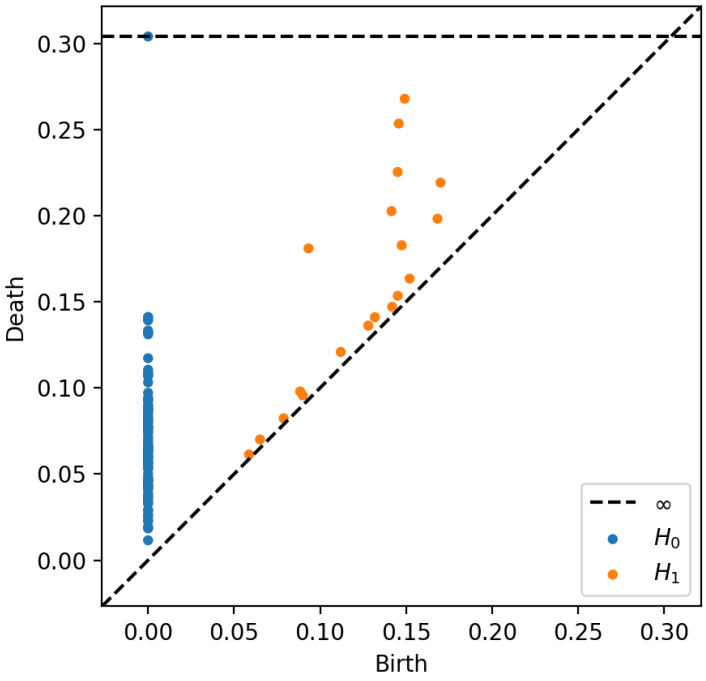

Figure 1.4: Un diagramma di persistenza.

sottografo completo su $n+1$ vertici, ed un complesso di cricche è un complesso simpliciale costruito a partire dalle cricche di un grafo. Per ogni $n+1$-cricca nel grafo c'è una n-faccia nel complesso simpliciale.

Una Clique Weight Rank Filtration su una rete pesata Ω combina la costruzione di un complesso di cricche con una soglia imposta sui pesi seguendo 3 passi:

- Ordina i pesi degli archi da ω_{max} a ω_{min} utilizzando come indice il parametro discreto ϵ_t.
- Ad ogni passo t nell'ordinamento decrescente degli archi considera il grafo $G(\omega_{ij}, \epsilon_t)$, ossia il sottografo di Ω con archi di peso maggiore di ϵ_t.
- Per ogni grafo $G(\omega_{ij}, \epsilon_t)$ costruisce il complesso di cricche $K(G, \epsilon_t)$.

I complessi di cricche sono annidati seguendo la crescita di t e determinano la Clique Weight Rank Filtration. La differenza concettuale tra una Vietoris-Rips Filtration e questa sta nel fatto che nella prima i generatori di omologia che persistono a lungo sono considerati più rilevanti e caratteristiche interessanti dei dati, mentre in questo caso si è più interessati a cicli di breve vita. Questo perché reti casuali o randomizzate mostrano generatori persistenti monodimensionali ad ogni scala, mentre generatori di breve vita testimoniano la presenza di organizzazione locale in scale diverse.[4]

1.3 L'algoritmo MAPPER

L'algoritmo Mapper permette di costruire un complesso simpliciale a partire da un dataset in modo da rivelare alcune delle caratteristiche topologiche dello spazio. Anche se non in maniera esatta, MAPPER è in grado di stimare importanti aspetti sulla connettività dello spazio sottostante ai dati così che possa essere esplorato in un formato visuale. Questa è una tecnica non supervisionata di generazione di una rappresentazione visuale dei dati che spesso rivela nuove intuizioni sui dati che altre tecniche non sono in grado di rivelare. Informalmente, l'algoritmo MAPPER agisce eseguendo un algoritmo di clustering locale guidato da una funzione di proiezione.[5] I passaggi sono i seguenti:

- Esegue una proiezione del dataset.

[4] `https://arxiv.org/pdf/1301.6498.pdf`

[5] `https://kepler-mapper.scikit-tda.org/theory.html`

- Ricopre la proiezione con intervalli o ipercubi sovrapposti. Kepler Mapper supporta solamente ipercubi n-dimensionali, ma in teoria si potrebbe usare qualsiasi forma.
- Clustering dei punti all'interno di un intervallo (clusterizzando sulla proiezione malgrado la perdita di precisione oppure sui dati originali) utilizzando un qualsiasi clusterer e metrica per le distanze tra i punti.
- I cluster diventano nodi in un grafo.
- A causa della sovrapposizione, un punto può apparire in più nodi. Quando ciò avviene un arco viene tracciato tra questi nodi.
- Come aiuto all'esplorazione visuale:
  - I nodi del grafo sono dimensionati tramite una funzione di interesse
  - I nodi del grafo sono colorati tramite una funzione di interesse
  - I nodi del grafo hanno una certa forma a seconda di una funzione di interesse
  - Gli archi del grafo sono dimensionati tramite una funzione di interesse
  - Gli archi del grafo sono colorati tramite una funzione di interesse
  - Gli archi dl grafo hanno una certa forma a seconda di una funzione di interesse
  - Sono mostrate statistiche descrittive sia sui nodi che sul grafo

2 | Metodologia di sviluppo

Pensare di avere un'idea ben definita dell'applicativo finale, soprattutto se trattasi di una web-app, è utopia. Tra l'altro, è importante considerare che i requisiti del software in questione sarebbero potuti cambiare da un momento all'altro. Proprio per queste motivazioni abbiamo deciso di gestire il processo di sviluppo tramite metodologia AGILE, nello specifico seguendo il modello iterativo/incrementale

2.1 Fase iniziale

Durante la fase iniziale, intercorsa dal 10 di Marzo fino alla fine dello stesso mese, sono state svolte le attività di analisi dei requisiti, selezione delle tecnologie e degli strumenti utili allo sviluppo, organizzazione delle iterazioni e definizione del workflow da utilizzare con Git.

Una volta terminato il periodo di studio degli argomenti alla base dell'applicazione (omologia, analisi topologica), il 25 Marzo è stata indetta una seconda riunione con il docente in modo da definire i requisiti funzional. E' emerso che il prodotto finale avrebbe dovuto permettere di gestire, tramite GUI, una pipeline per l'analisi topologica e la visualizzazione di datasets.

In sostanza, rendere user-friendly e automatizzato un processo che fino a quel momento era eseguibile solo tramite scrittura di codice.

Subito dopo la riunione con il docente, il gruppo si è riunito privatamente per l'attività di specifica dei requisiti.

2.1.1 Specifica dei requisiti

Secondo i requisiti funzionali richiesti dal docente, l'applicazione avrebbe dovuto permettere all'utente, tramite GUI, l'esecuzione di analisi su dataset da lui stesso caricati, permettendogli di specificare i vari parametri. In aggiunta, il software avrebbe dovuto gestire la visualizzazione interattiva dei dataset e dei risultati ottenuti.

2.1.2 Le tecnologie

La necessità dell'applicativo di avere un'interfaccia grafica era soddisfacibile in due modi:

- Sviluppare un software desktop
- Sviluppare una web app

La decisione di sviluppare una web app è stata presa considerando i seguenti fattori: Innanzitutto, lo sviluppo tramite HTML e CSS, oltre ad essere più semplice ed immediato rispetto alle librerie grafiche desktop (Qt, ad esempio), permette una personalizzazione maggiore.

Inoltre, poiché l'applicazione nasce per lavorare nell'ambito della ricerca e dell'analisi dati, le operazioni che ci aspettiamo effettui costano molto in termini di potenza computazionale, e andrebbe installata in macchine potenti.

Sviluppare un software desktop avrebbe portato a due possibilità:

- Un'applicazione installata in un server principale a cui è possibile accedervi una persona alla volta.
- Applicazione installata su i PC di tutti gli utilizzatori, che però richiede hardware prestante.

Sarebbe risultata scomoda, soprattutto se la si immagina in un contesto dove venga sfruttata da più di una persona.

L'idea dell'applicazione web nasce proprio da queste necessità. Un unico server, prestante, che fornisce il servizio. Un servizio utilizzabile da qualsiasi membro autorizzato e da qualsiasi dispositivo, smartphone compreso.

2.1.3 Project Management

Per la gestione dello sviluppo si è deciso di utilizzare Trello, una web app per il Project Management.

Dopo aver definito il layout della bacheca Trello, sono stati registrati nell'apposita scheda i requisiti funzionali e le specifiche dell'applicativo in modo da poter essere consultabili, in qualsiasi momento, da tutti i membri del gruppo.

Lo sviluppo del progetto è stato organizzato secondo le seguenti regole:

- Per ogni iterazione è stato impostato un time-boxing di una settimana e mezza.

- All'inizio di ogni iterazione è stata svolta una riunione, in modo da discutere di eventuali problematiche riscontrate nell'iterazione appena conclusa e di pianificare i tasks per il ciclo successivo.
- Ogni task pianificato è stato registrato in Trello, corredato di tutte le sue specifiche ed assegnato al membro del gruppo incaricato.

2.1.4 Git

Per fare in modo di lavorare in cooperazione si è scelto di utilizzare Git, sfruttando il workflow "Pull Request".

Esiste una repository principale, contenente due branch: master e develop.

Nel "master" è memorizzato l'applicativo alla versione stabile. Dalla branch master nasce il "develop", la branch utilizzata come "banco di lavoro". Sul develop però non è ammesso lavorare direttamente. Per implementare nuove funzionalità (o correggere bugs) è occorrente eseguire una fork del develop e lavorare su di essa separatamente (Utilizzando il workflow che si preferisce). Quando la modifica è pronta, viene chiamata una "Pull Request" alla repository principale. Questo workflow, oltre a rendere lo sviluppo più pulito ed organizzato, facilita il processo di rollback, nel caso dovesse essercene bisogno. Nel momento in cui il develop raggiunge un livello di stabilità appropriato, viene effettuata una Pull Request sul master.

2.2 Prima Iterazione
(01/04/2019 - 11/04/2019)

Durante la prima iterazione (01/04/2019 - 11/04/2019) sono state gettate le basi architetturali dell'applicativo.

Nella consueta riunione indetta all'avvio dell'iterazione è stata definita l'architettura Research-Dataset-Analysis che verrà spiegata dettagliatamente nella sezione `Backend`. La struttura è stata pensata in modo da poter offrire un'organizzazione coerente alle numerose analisi e ai numerosi dataset che la piattaforma si troverà a dover ospitare.

Come primo task abbiamo provveduto ad installare Django tramite cookiecutter, fornendogli una configurazione professionale e sicura. Successivamente sono state create le app necessarie ai fini della prima iterazione: `Research` e `Dataset`.

In questa iterazione è stata sviluppata l'intera app Research, sia per quanto riguarda il modello, sia per quanto riguarda le views e i templates. E' stato possibile sviluppare l'app interamente già dalla prima iterazione data la sua semplicità e indipendenza con il resto delle funzionalità dell'applicativo.

L'app dataset, invece, è stata solo improntata. E' stato creato il modello generico, senza specificarne nessuna tipologia.

Nel frattempo, grazie a cookiecutter, è stato anche implementato un template grafico di base per effettuare test di funzionamento.

2.3 Seconda Iterazione
(12/04/2019 - 23/04/2019)

Per la seconda iterazione (12/04/2019 - 23/04/2019) venne previsto di implementare la funzionalità per il caricamento di un dataset di tipo CSV e l'analisi di filtrazione. Il Dataset CSV è stato implementato sfruttando l'ereditarietà concreta fornita da Django (CSVDataset -> Dataset). Questa, seppur generalmente sconsigliata, è risultata essere la soluzione migliore per la gestione dei modelli dei vari tipi di dataset, considerando che ognuno possiede dei parametri differenti. Ogni tipologia di dataset eredita dalla classe "Dataset". A livello database, questo si concretizza con una relazione 1a1 tra le due tabelle. Le analisi sono state strutturate nella stessa maniera: Un modello "Analisi" generale ereditato da tutte le sotto-tipologie di analisi, le FiltrationAnalysis.

2.4 Terza Iterazione
(24/04/2019 - 1/05/2019)

Nella terza iterazione è stata aggiunta la compatibilità con i dataset EDF e la possibilità di effettuare le analisi Mapper.

2.5 Quarta Iterazione
(2/05/2019 - 11/05/2019)

Nella quarta iterazione è stata aggiunta la possibilità di eseguire analisi a partire da matrici di distanza pre-computate. Inoltre, i plotting dei dataset sono stati migliorati, permettendo interattività e zoom.

È stato inoltre implementato il calcolo dell'entropia.

2.6 Quinta Iterazione
(12/05/2019 - 22/05/2019)

Nella quinta iterazione, oltre all'implementazione front-end di una schermata di loading da mostrare durante le elaborazioni, è stata inserita la possibilità di effettuare analisi per "finestre". La feature è stata sviluppata tramite l'inserimento di un nuovo modello, "Window", agganciato ad "Analysis" tramite relazione 1 - *.

2.7 Sesta Iterazione
(1/06/2019 - 15/06/2019)

Nella sesta iterazione il lato frontend è stato notevolmente migliorato tramite l'inserimento di helptext nei forms, in modo da aiutare l'utente a capire meglio i campi. E' stata inserita la possibilità di scaricare i dati delle elaborazioni, e la possibilità di caricare più di una matrice di distanza precomputata alla volta in un'analisi.

2.8 Settima Iterazione
(15/06/2019 - 28/06/2019)

La settima iterazione, quella finale, ha visto l'implementazione della grafica definitiva e delle elaborazioni bottleneck.

3 | Tecnologie utilizzate

3.1 Python

È un linguaggio multi-paradigma che ha tra i principali obiettivi dinamicità, semplicità e flessibilità. E' molto utilizzato nel settore del machine learning e data science, data la sua sintassi semplice. Per lo sviluppo di questo progetto la preferenza è ricaduta su questo linguaggio e non su, ad esempio, R, che nell'ambito del data science è un prodotto validissimo, vista la maggiore conoscenza da parte di tutti i membri del gruppo, la popolarità maggiore (documentazione più chiara, supporto online più celere) e il numero di librerie e framework a supporto.[1]

3.2 Django

Django è un web-framework con licenza open source per lo sviluppo di applicazioni web, scritto in Python. Il framework segue il pattern architetturale "Model-Template-View", una rivisitazione del ben più noto "Model-View-Controller".[2]

Models

L'insieme delle classi utilizzate per la rappresentazione dei modelli dei dati, che poi andranno a costituire le tabelle del database, sono chiamate "models". Tutti i modelli derivano dalla classe di default "Model", che implementa una serie di funzionalità di default per la gestione delle operazioni CRUD. Questo è possibile grazie all'avanzatissimo ORM (Object Relational Mapping) integrato, considerato uno tra i migliori sul mercato. Tutti i modelli sono contenuti all'interno del modulo `models`.

[1] `https://www.python.org/`
[2] `https://www.djangoproject.com/`

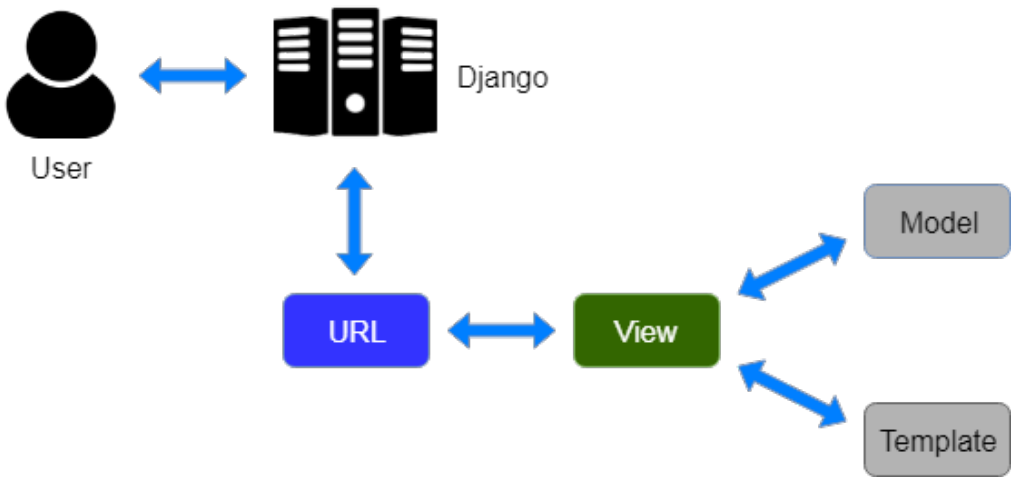

Figure 3.1: Il pattern Model-Template-View in Django.

Views

Layer potentissimo. Funge sia da controller che da elaboratore dei template. E' il router a cui tutte le richieste vengono inoltrate. Preleva i dati dai modelli, li elabora secondo le regole definite dallo sviluppatore e genera un template che restituisce all'utente.

Il punto di forza è la facilità d'uso, migliorata ulteriormente tramite l'implementazione, dalle ultime versioni di Django, delle "Class Based View", delle super classi da cui è possibile ereditare una serie di parametri e funzioni che permettono la creazioni di views comuni (list, detail, create, delete, redirect, form, etc) in pochi secondi.

Templates

Insieme di files HTML misti a codice Python. Vengono popolate dai dati tramite le views.

Un progetto Django è costituito da una o più app. Secondo le best practices, ogni app dovrebbe gestire una funzionalità del progetto, nel modo più indipendente possibile.

Data la necessità di sviluppare in breve tempo un servizio robusto, sicuro e basato sul Python, la scelta di questo framework è stata una semplice conseguenza.

Django, infatti, oltre ad offrire un'architettura ben progettata, incorpora diversi componenti built-in, come ad esempio: un pannello di amministrazione integrato, un sistema per la gestione degli utenti, un sistema per lo storaging di password tramite SECRET KEY [...], che facilitano e velocizzano di molto

lo sviluppo, limitando gli errori.

Nota: Importanti compagnie hanno deciso di adottare Django per sviluppare i loro servizi, ad esempio: Instagram, Spotify, Youtube, Dropbox e alcuni portali della NASA.

3.2.1 Channels

Channels è un progetto che estende le potenzialità di Django oltre l'HTTP, permettendo l'uso di WebSockets, protocolli di chat e IoT e altro ancora. Si basa sulla specifica Python ASGI (Asynchronous Server Gateway Interface) che provvede una interfaccia standard tra server e applicazioni Python capaci di operare in modo asincrono. Channels aggiunge uno strato asincrono all'architettura Django di base, lasciando eseguire Django in modalità sincrona ma gestendo connessioni e socket in modo asincrono e permettendo di utilizzare entrambe le modalità.[3]

Nel progetto utilizziamo Channels per sfruttare il protocollo WebSocket. Attraverso una WebSocket è stato possibile mostrare una barra di caricamento durante la creazione di una Analisi e del calcolo delle distanze di Bottleneck. Utilizzando questo protocollo molti altri miglioramenti dell'esperienza dell'utente finale possono essere resi possibili.

3.3 PostgreSQL

PostgreSQL è un completo DBMS ad oggetti rilasciato con licenza libera (stile Licenza BSD).[4]

PostgreSQL è una reale alternativa sia rispetto ad altri prodotti liberi come MySQL, Firebird SQL e MaxDB che a quelli a codice chiuso come Oracle, IBM Informix o DB2 ed offre caratteristiche uniche nel suo genere che lo pongono per alcuni aspetti all'avanguardia nel settore dei database.

PostgreSQL permette agli utenti di definire nuovi tipi basati sui normali tipi di dato SQL, permettendo al database stesso di comprendere dati complessi. Inoltre, permette l'ereditarietà dei tipi, uno dei principali concetti della programmazione orientata agli oggetti. Dalla versione 7.4 è diventato molto più semplice creare ed usare tipi personalizzati attraverso il comando "CREATE DOMAIN".

La scelta è ricaduta su questo DBMS data la sua forte integrazione con Django (L'unica libreria ufficialmente mantenuta per l'aggiunta di tipologie di

[3] `https://channels.readthedocs.io/en/latest/`

[4] `https://www.postgresql.org/`

campi all'ORM è proprio quella per postgres) e la quantità di campi "speciali" che offre, come quello che è stato usato maggiormente: il JSONField.

3.4 Ripser

Ripser è una libreria Python per il calcolo dell'omologia persistente basata sull'omonima libreria C++.[TSBO18] Ripser fornisce un'interfaccia intuitiva per:

- calcolo della coomologia persistente per dataset sparsi e densi;
- visualizzazione di diagrammi di persistenza;
- calcolo di lowerstar filtrations su immagini;
- calcolo di cocatene rappresentative.

Nel progetto utilizziamo Ripser per il calcolo della coomologia persistente e per visualizzare diagrammi di persistenza. Sia le analisi di tipo Vietoris-Rips Filtration che Clique Weight Rank Filtration sono effettuate utilizzando la funzione `ripser()` con una matrice di distanza o correlazione rispettivamente. I risultati sono poi visualizzati utilizzando il metodo di istanza `plot()` della classe `Rips`.

3.5 Kepler Mapper

Kepler Mapper è una libreria Python che implementa l'algoritmo MAPPER.[5] Kepler Mapper può essere utilizzato per la visualizzazione di dati ad alta dimensionalità e nuvole di punti tridimensionali. Inoltre è compatibile con algoritmi di scaling e clustering forniti dalla libreria Scikit-Learn.[vVS19]

Nel progetto utilizziamo Kepler Mapper per le analisi con l'algoritmo MAPPER, dalla loro esecuzione fino alla visualizzazione.

3.6 NumPy e SciPy

Numpy è la libreria fondamentale per il calcolo scientifico in Python. Aggiunge al linguaggio Python il supporto ad array multi-dimensionali e matrici di

[5] Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition(Singh et al.)

grandi dimensioni e una collezione di funzioni matematiche di alto livello per operare su questi array.[6]

Scipy è una libreria Python free e open-source per il calcolo tecnico e scientifico. Scipy contiene moduli per l'ottimizzazione, algebra lineare, calcolo integrale, interpolazione, Trasformata di Fourier Veloce, processamento di segnali e immagini, risolutori di equazioni differenziali ordinarie e altre attività comuni in scienza e ingegneria. Scipy espande gli array di Numpy, i quali assieme a Matplotlib, Pandas e SymPy fanno parte della pila Numpy o pila Scipy.[7]

Nel progetto facciamo ampio uso degli array Numpy e utilizziamo la funzione Numpy `corrcoef()` per calcolare matrici di correlazione. Altri usi della libreria sono per operazioni su matrici come riempire la diagonale con dei valori o produrre una trasposta. Scipy viene invece utilizzato per creare matrici di distanza quadrate combinando la funzione `pdist()` con `squareform()`.

3.7 Matplotlib e Pyplot

Matplotlib è una libreria Python open-source per disegnare diagrammi che utilizza una API di tipo orientato ad oggetti che fa affidamento su librerie grafiche general-purpose come Tkinter, wxPython, Qt o GTK+. Pyplot è un modulo di Matplotlib che fornisce un'interfaccia simile all'ambiente di sviluppo MATLAB.[Hun07]

Nel progetto utilizziamo Pyplot per disegnare ogni sorta di grafici, sia direttamente come nel caso della visualizzazione di dataset o dell'entropia che indirettamente nella visualizzazione di diagrammi di persistenza (Persim utilizza Pyplot).

3.8 Scikit-learn

Scikit-learn è una libreria Python free software per il machine learning, contenente algoritmi per la classificazione, regressione e clusterizzazione tra cui macchine a vettori di supporto, foreste casuali, potenziamento del gradiente, k-means e DBSCAN. È stata progettata per interoperare con le librerie NumPy e SciPy.[8]

[6] `https://www.numpy.org/`
[7] `https://www.scipy.org/`
[8] `https://scikit-learn.org/stable/`

Nel progetto utilizziamo Scikit-learn per le sue funzioni di clustering e scaling utilizzate come parametri delle funzioni di Kepler Mapper.

3.9 Persim

Persim è una libreria Python che fornisce vari strumenti per l'analisi di diagrammi di persistenza.[9] Le sue capacità sono:

- Immagini di persistenza;
- Distanza di bottleneck;
- Distanza di Gromov-Hausdorff;
- Sliced Wasserstein kernel;
- Heat Kernel;
- Visualizzazione dei diagrammi.

Nel progetto utilizziamo questa libreria indirettamente attraverso Ripser per la visualizzazione di diagrammi e direttamente per il calcolo e visualizzazione della distanza di bottleneck.

3.10 Pandas

Pandas è una libreria Python free e open source che fornisce strutture dati e strumenti di analisi di dati facili da utilizzare e ad alta performance. In particolare si specializza per dati di tipo numerico e serie temporali. La libreria è particolarmente ottimizzata per essere performante con sezioni critiche di codice scritte in Cython o C.[10]

Nel progetto questa libreria è utilizzata per la lettura, graficamento e scrittura di file testuali, in particolare file in formato CSV. Ciò è reso possibile utilizzato la classe `DataFrame` e i suoi metodi `to_csv()`, `plot()` e `read_csv()`.

[9] `https://persim.scikit-tda.org/`
[10] `https://pandas.pydata.org/`

3.11 PyEDFlib

PyEDFlib è una libreria Python open-source per la lettura e scrittura di file in formato EDF basta su EDFlib. La libreria supporta inoltre file in formato EDF+, BDF e BDF+.[11]

Nel progetto utilizziamo questa libreria per leggere file in formato EDF e, riutilizzando degli snippet di codice presente negli esempi, anche per visualizzarne il grafico, seppur facendo comunque affidamento su Pyplot.

3.12 Mpld3

La libreria Python Mpld3 unisce la nota libreria Python Matplotlib per disegnare grafici con la libreria JavaScript D3js per creare visualizzazioni interattive di dati. Il risultato è una semplice API per esportare grafici di Matplotlib in HTML per essere visualizzati da un browser, in pagine web, blog o notebook IPython.[12]

Nel progetto la utilizziamo per rendere interattivi i grafici relativi ai dataset e il grafico dell'entropia.

3.13 JavaScript

JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client.[13]

In questo progetto è stato utilizzato nel front-end, per inserire animazioni (interazione con i grafici), rendere dinamica la visualizzazione di alcuni forms e mostrare le schermate di caricamento.

3.14 JSON

JSON è un formato per l'interscambio dei dati e nell'ambito di questo applicativo è stato utilizzato per la memorizzazione di quelli ottenuti dall'analisi dei datasets.[14] L'alternativa principale al JSON è l'XML, ma la scelta è ricaduta sul primo per le seguenti motivazioni:

[11] `https://pyedflib.readthedocs.io/en/latest/`
[12] `https://mpld3.github.io`
[13] `https://www.javascript.com/`
[14] `https://www.json.org/`

- PostgreSQL possiede un tipo di campo specifico per la memorizzazione dei JSON
- Semplicità di lettura
- Orientato ai dati, a differenza dell'XML che è orientato al documento (comprende quindi tags per il layout, in questo caso non utili)
- Supporto arrays

3.15 CSV files

Il formato CSV (Comma Separated Values) è una specifica non standardizzata per file testuali per l'esportazione e importazione di dati. In questo formato ogni record è rappresentato da una riga, dove ogni valore è separato con un apposito carattere separatore, generalmente ',' o ';'. Ogni riga è terminata da un apposito carattere di fine riga che dipende dal sistema operativo.

Nel progetto utilizziamo i file CSV sia come file di input per l'aggiunta di nuovi dataset, sia come output per i valori di entropia e le distanze di bottleneck.

3.16 EDF files

Il formato EDF (European Data Format) è una specifica non proprietaria standardizzata per lo scambio e salvataggio di serie temporali in ambito medico. In questo formato i segnali sono divisi in canali che possono avere differenti frequenze di campionamento e può essere presente un header con informazioni generali e tecniche per ogni segnale.

Nel progetto utilizziamo file EDF come file di input per l'aggiunta di nuovi dataset.

3.17 BlockUI

BlockUI è un plugin jQuery che permette di simulare un comportamento sincrono senza bloccare il browser. Quando viene attivato, impedisce all'utente di interagire con la pagina ulteriormente finché non viene disattivato.[15]

Nel progetto questo plugin viene utilizzato per bloccare la pagina durante l'esecuzione di una analisi o durante il calcolo della distanza di bottleneck,

[15] `http://malsup.com/jquery/block/`

e durante il blocco mostrare una barra di caricamento e un bottone per annullare l'operazione.

3.18 Git

Git è un software di controllo di versione distribuito, creato da Linus Torvalds nel 2005. La principale motivazione che ha portato alla scelta di questo sistema di versionamento è nata studiando il suo sistema di branching che, a differenza dei sistemi di controllo di versione centralizzati (come SVN) è meno costoso e rende più facile il processo di merging (favorendo la cooperazione). Inoltre, Git crea copie dei file molto più leggere di SVN. Quest'ultimo infatti crea ben due copie di ciascun file: una per l'utilizzo da parte dell'utente e una copia nascosta per operazioni come status, differenze e commit.[16]

3.19 Bootstrap

Bootstrap è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript. I pregi derivanti dall'utilizzo di questo framework sono la velocizzazione del processo di sviluppo e di creazione del layout. Difetta, invece, nella personalizzazione. Visto che l'applicativo non richiedeva un layout grafico complesso Bootstrap si è rivelata essere la scelta migliore.[17]

[16] `https://git-scm.com/`
[17] `https://getbootstrap.com/`

4 | Strumenti utilizzati

4.1 GNU/Linux

L'applicativo è stato sviluppato da entrambi i componenti del gruppo poggiandosi su sistemi operativi basati su kernel Linux e il progetto GNU. In particolare: Debian (Verdolini) e Linux Mint Debian Edition 3 (Belenchia).

Le motivazioni che hanno portato a preferirlo a Windows sono le seguenti[1]

- Gratuito
- Terminale fortemente integrato nel sistema
  - Windows possiede powershell, che seppur migliore del prompt dei comandi di default, non regge il confronto con la shell messa a disposizione dalla controparte GNU/Linux, ricca di funzionalità e capace di svolgere qualsiasi mansione utilizzando una sintassi semplice (a differenza di windows che ne utilizza una molto più articolata).
- Pieno controllo del sistema.
- Strumenti per lo sviluppo reperibili e installabili più facilmente.
- Integrazione nativa dei database all'interno del sistema (GNU/Linux nasce come sistema sia lato server, che lato user, che lato database. Questa sua versatilità lo rende potentissimo per gli sviluppatori).

4.2 VSCode

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft per Windows, GNU/Linux e macOS. Esso include supporto per debugging,

[1] `https://www.linux.org/` e `https://www.fsf.org/`

un controllo per Git integrato, Syntax highlighting, IntelliSense, Snippet e refactoring del codice. È un software libero, anche se la versione ufficiale è sotto una licenza proprietaria.[2]

E' sviluppato in Electron, un framework open source sviluppato da GitHub che consente lo sviluppo di applicazioni GUI desktop che utilizzano tecnologie web. Combina il motore di rendering Chromium e il runtime NodeJS.

Nell'ultimo sondaggio di StackOverflow Visual Studio Code è risultato come l'ambiente di sviluppo più popolare (34.9%).

Le caratteristiche che hanno contribuito alla scelta di questo strumento sono principalmente due:

Versatilità

VSCode è un editor "neutro". Seppur ideato da Microsoft per lo sviluppo in TypeScript, grazie alla sua interfaccia grafica neutrale e al suo sistema di estensioni (di cui numerose ufficiali) è possibile renderlo adatto a qualsiasi linguaggio. Questa caratteristica è stata ritenuta fondamentale per lo sviluppo dell'applicativo, dato che fin dal principio era stato previsto l'utilizzo, oltre che del Python, anche dell'HTML, del CSS e del Javascript.

Immediatezza

L'editor mette a disposizione tutte le funzionalità necessarie agli sviluppatori senza complicarne l'intefaccia.

I numerosi snippets da tastiera, iil terminale integrato, il debugger e l'esecutore interno dei comandi permettono un processo di sviluppo rapido e senza fronzoli.

4.3 Pgadmin4

PGAdmin è un'applicazione libera sviluppata in C++ che permette di amministrare, in modo semplificato e tramite interfaccia grafica, database PostgreSQL. L'applicativo, oltre a permettere la creazione, rimozione e modifica di database e tabelle, offre un sistema per la gestione degli utenti e dei permessi. Il software in questione è stato scelto data la sua vasta popolarità nella community.[3]

[2] `https://code.visualstudio.com/`

[3] `https://www.pgadmin.org/`

4.4 Trello

Trello è uno strumento web di base gratuito, ma che prevede dei piani premium con funzionalità aggiuntive, per il project management. L'applicazione permette la creazione di "bacheche" virtuali. All'interno di ogni bacheca è possibile creare liste di task, organizzabili a piacimento.[4]

L'applicativo presenta tre elementi chiave:

- Board: la bacheca
- List: a seconda della natura della board, rappresenta i macro-step di un progetto; idee, persone in un team.
- Card: Unità di base della board. Rappresenta il singolo task o un'idea.

Le board possono essere sia private che pubbliche. E' possibile condividere una board con account specifici (scegliendo i permessi per ogni membro) in modo da favorire la cooperazione e il coordinamento. Le card, contenenti i task, si trovano all'interno delle liste e possono essere spostate tramite un drag and drop da una lista all'altra.

Ogni card può essere assegnata ad uno o più membri e possiede un "profilo", chiamato `card-back`, a cui è possibile aggiungere una descrizione, dei commenti, una checklist, un'etichetta, una scadenza e ulteriori campi installabile tramite estensioni.

Il punto di forza di Trello risiede nella sua estrema semplicità e personalizzazione. Non imponendo un formato specifico per l'organizzazione delle bacheche, è possibile scegliere quello che meglio si adatta al contenuto che si vuole andare a gestire.

In questo caso si è scelto di organizzarlo utilizzando le seguenti liste:

- Resources, Contenente tutte le risorse che sarebbero potute risultare utili ai fini del progetto. Requisiti funzionali, librerie, pipeline di configurazione varie, chiavi segrete, etc.
- Backlog, Contenente tutti i task ancora da svolgere
- Blocked, Contenente i task che andrebbero svolti, ma che per qualche motivo sono "congelati". (Esempio: Relazione)
- Sprint-Backlog, Contenente ii task prossimi all'esecuzione
- Sprint, Contenente i task in esecuzione

[4] `https://trello.com/en`

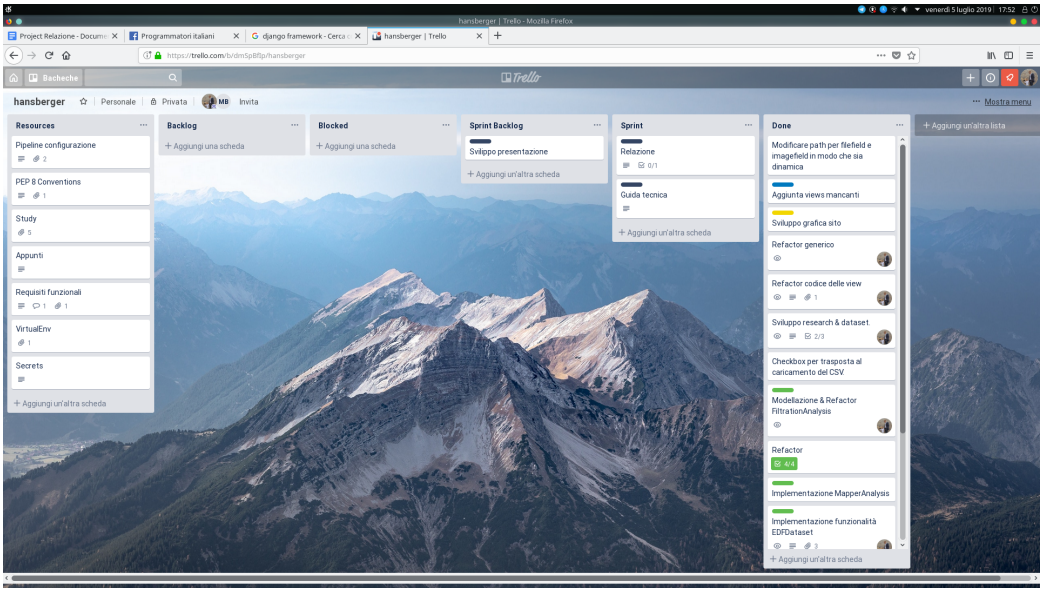

Figure 4.1: Una schermata di Trello.

- Done, Contenente i task svolti

Trello fin da subito si è rivelato essere uno strumento fondamentale per l'organizzazione dello sviluppo, la cooperazione tra i membri del gruppo, la gestione delle tempistiche e la condivisione delle informazioni.

4.5 CookieCutter

Cookiecutter è uno strumento nato per automatizzare il processo di layout di un progetto software. In questo caso è stato utilizzato cookiecutter/django, una fork dell'applicativo originale che permette di organizzare, personalizzando numerosi parametri tramite CLI, un progetto Django.[5]

In sintesi, cookiecutter/django si occupa di:

- Creare la struttura dei files e delle cartelle del progetto.
- Generare i file di configurazione per l'applicativo (sia per l'avvio in locale che in produzione).
- Generare i files necessari per dipendenze dell'applicativo, e già riempiti con alcuni pacchetti iniziali di default (Tra cui Django).

[5] `https://cookiecutter.readthedocs.io/en/latest/`

L'uso di CookieCutter è stato fondamentale per lo sviluppo dell'applicativo, dato che ha permesso di azzerare il tempo che era stato previsto per il layout e la configurazione dei numerosi parametri che Django richiede.

4.6 Github

Github è un servizio di hosting per progetti software. E' un'implementazione dello strumento di controllo di versione distribuito Git. La scelta è ricaduta su questo servizio data la maggiore conoscenza da parte di entrambi i membri del gruppo e la scelta di utilizzare un sistema di versionamento come Git.[6]

4.7 Pip

Pip è il sistema standard usato per la gestione di pacchetti Python (installazione, rimozione, aggiornamento).[7]

4.8 Virtualenv

Virtualenv è uno strumento utile per la creazione di ambienti Python virtuali. Il tool è stato integrato nella librerie standard da Python 3.3.[8]

Avere un ambiente virtuale è stato fortemente necessario per lo sviluppo.

Infatti, un applicativo Python può richiedere l'installazione di una o più librerie. Queste, se installate globalmente nel sistema, potrebbero creare conflitti con altre già presenti. Proprio per questo motivo è nato virtualenv. Tramite la creazione di environments virtuali è possibile gestire separatamente le librerie di ogni progetto. Il vantaggio è immediato: un'organizzazione coerente dei pacchetti si tramuta a livello pratico in una maggiore facilità nella registrazione delle dipendenze necessarie all'applicativo.

[6] `https://github.com/`
[7] `https://pip.pypa.io/en/stable/`
[8] `https://virtualenv.pypa.io/en/latest/`

5 Installazione

5.1 Pre-requisiti

I pre-requisiti necessari per l'installazione dell'applicativo sono i seguenti:

- PostgreSQL
- Python >= 3.7

Le modalità per l'installazione delle librerie utilizzate dall'applicativo sarà descritta nel paragrafo `Installazione`.

5.2 Installazione locale

Facoltativo E' consigliabile utilizzare un virtualenv. Per crearne uno è sufficiente spostarsi all'interno della cartella principale del progetto e digitare il seguente comando:

```
python -m venv .venv
```

Una volta terminata l'operazione digitare il seguente comando per abilitare l'environment:

```
source .venv/bin/activate
```

Si potrà notare, a lato della riga di comando, la stringa (.venv). Questa sta a significare che un virtual environment è attivo.

Per disattivare il virtual environment è sufficiente digitare:

```
deactivate
```

Installazione

1. Installare le dipendenze richieste dall'applicativo: `pip install -r requirements/local.txt`
2. Creare un file chiamato ".env" all'interno della root del progetto. In questo file specificare le seguenti variabili:

(a) DATABASE_URL, per esempio:
`postgres://myuser:mypassword@127.0.0.1:5432/hansberger`

(b) DJANGO_SECRET_KEY, generabile dal seguente servizio
`www.miniwebtool.com/django-secret-key-generator/`

3. Settare la variabile d'ambiente `DJANGO_READ_DOT_ENV_FILE=True` (Questa andrà ri-settata ad ogni ri-accensione dell'environment)

4. Aggiungere, nella lista `ALLOWED_HOST` all'interno del file config/settings/local.py l'indirizzo IP dell'host su cui si vuole eseguire il servizio

5. Se si vuole disabilitare le opzioni di debug, modificare il valore di `Debug` a `False` (nel medesimo file di configurazione)

6. Per avviare l'applicativo digitare:

   `python manage.py runserver <ip>:<port>`

6 Back-end

Il Back-end è suddiviso in 3 applicazioni (o app): `Research`, `Dataset` e `Analysis`. La suddivisione è solamente a livello logico, in quanto queste applicazioni sono strettamente collegate fra loro.

6.1 Research

L'app `Research` gestisce l'entità "Ricerca", che si può immaginare come una directory o cartella nella quale sono raccolti sia `Datasets` che `Analyses`. Una Ricerca può essere creata o eliminata (assieme a tutti i `Datasets` e `Analyses` contenuti) e si può visualizzare la lista di tutte le Ricerche inserite nel sistema. Nella fase di creazione di una ricerca, si può anche associare una descrizione come aiuto mnemonico riguardo il contesto di una ricerca (per esempio "Analisi effettuate sul dataset X" o "Analisi studenti"). Dopo aver creato una Ricerca, si possono aggiungere `Datasets` o `Analyses` che poi saranno accessibili esclusivamente da quella ricerca.

6.2 Dataset

L'app `Dataset` gestisce l'entità "Dataset", ovvero una generica collezione di dati. Un Dataset è caricabile nel contesto di una Ricerca e sarà utilizzabile solo all'interno della stessa, con un vincolo di unicità nel nome (tra tutti i Dataset nella stessa Ricerca) che gli viene assegnato in fase di caricamento. I tipi di file supportati correntemente sono 2, file testuali e file EDF.

6.2.1 Caricamento dei Datasets

Un file testuale viene generamente inteso come file in formato CSV, tuttavia anche normali file di testo nel formato adeguato sono supportati. Nella schermata di caricamento di un file testuale è presente l'opzione `transpose`

settata di default. Di norma i file CSV che rappresentano serie temporali hanno una colonna con gli indici temporali e una o più colonne per le misurazioni effettuate, quindi per convertire il dataset in una serie temporale che ha le misurazioni effettuate come righe e gli indici temporali come variabile indipendente è necessario trasporre la matrice associata al Dataset. Togliendo l'impostazione `transpose` questa operazione viene saltata. Le altre impostazioni peculiari del caricamento di file testuali sono `values separator character`, `identity column index` e `header row index`. Il `values separator character` specifica il carattere usato come separatore dei valori nel file, che generalmente è ',' o ';' ma in un file testuale potrebbe anche essere lo spazio ' '. L' `identity column index` è l'indice della colonna (partendo da 0) che identifica le righe, o in altre parole la colonna con gli indici temporali. Infine,l' `header row index` è l'indice della riga (partendo da 0) che contiene le etichette delle colonne, se presente.

Un file EDF invece non ha opzioni particolari durante il caricamento, eccezion fatta che il file deve appunto rispettare il formato EDF.

In entrambi i casi il limite correntemente impostato per il caricamento di un file è 100MB.

6.2.2 Salvataggio dei Datasets

Quando un Dataset viene aggiunto, il file caricato viene salvato fisicamente in una cartella locale della macchina sulla quale risiede il server, e tale file viene acceduto sia quando il Dataset viene visualizzato sia quando i suoi dati vengono acceduti da una Analisi: dato che alcuni Dataset (specialmente quelli provenienti da file EDF) possono avere milioni di record non abbiamo ritenuto plausibile salvarne sul database i contenuti. Il fatto che per ogni analisi bisogna prima caricare il Dataset da file non è stato considerato come un costo eccessivo dato che il tempo impiegato per ciò sarebbe di molto inferiore a quello per svolgere un'analisi. Anche durante la visualizzazione, il file viene letto e visualizzato sul momento, senza salvare sul database il grafico perché potrebbe essere molto grande. In alternativa si potrebbe salvare il grafico su disco, ma abbiamo preferito evitare il più possibile di salvare file localmente.

6.2.3 Visualizzazione dei Datasets

I Datasets vengono visualizzati utilizzando `pyplot` e quindi resi interattivi convertendoli in formato SVG utilizzando `mpld3`. Purtroppo la libreria, ormai abbandonata dagli sviluppatori, non supporta il dimensionamento responsive dei grafici, quindi la dimensione dei grafici è stata impostata staticamente ad una lunghezza di 10 pollici e una altezza di 5 pollici.

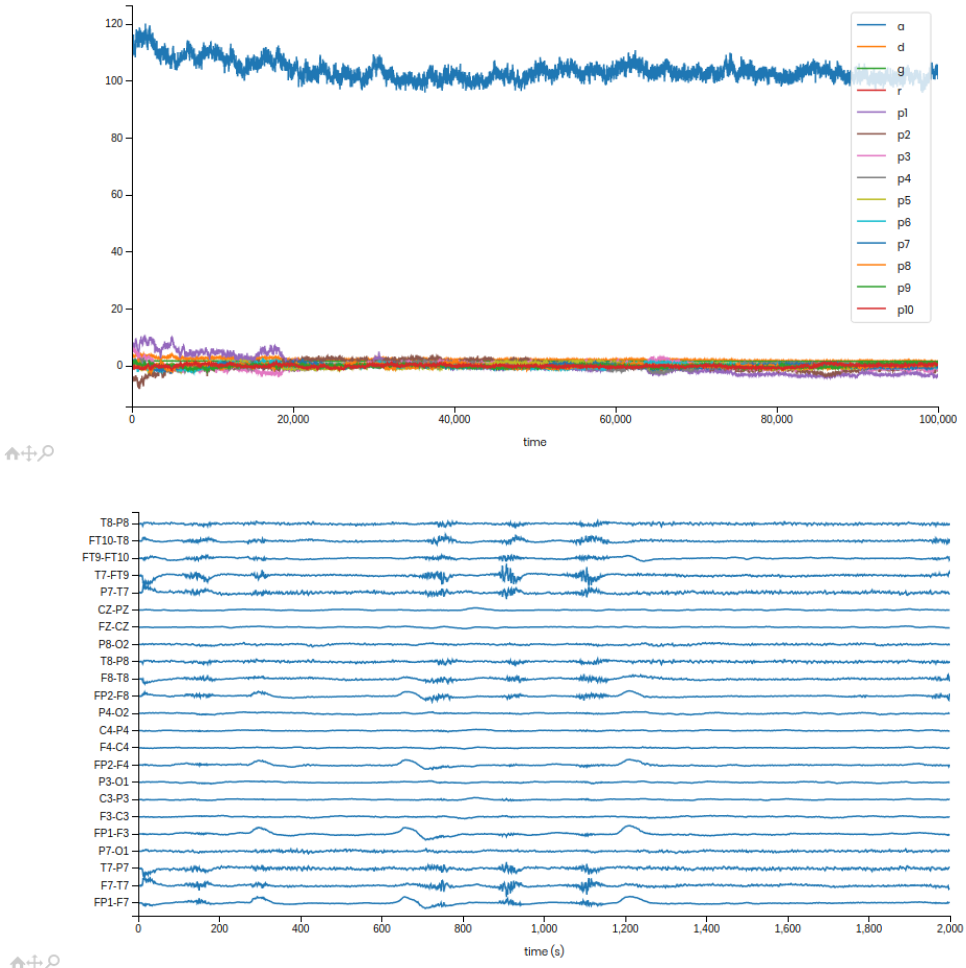

Figure 6.1: Visualizzazione di un dataset proveniente da un file CSV [Sopra] e da un file EDF [Sotto].

6.3 Analysis

L'app `Analysis` gestisce l'entità "Analisi", ovvero l'esecuzione di un certo algoritmo su un dato Dataset. Una Analisi è effettuata nel contesto di una Ricerca e sarà visualizzabile solo all'interno della stessa, con un vincolo di unicità nel nome (tra tutte le Analisi nella stessa Ricerca) che gli viene assegnato in fase di creazione. Le tipologie di Analisi implementate sono 3: Vietoris-Rips Filtration e Clique-Weighted Rank Filtration (da qui in avanti genericamente chiamate "Filtration Analysis") utilizzando `Ripser` e l'algoritmo Mapper con `Kepler Mapper` (che chiameremo "Mapper Analysis"). Una qualsiasi Analisi inoltre è effettuabile caricando direttamente le matrici di distanza che si vogliono utilizzare, senza selezionare alcun Dataset.

6.3.1 Creazione di una Analisi

Nella creazione di una nuova Analisi prima di tutto attraverso un `Form` si sceglie se si vuole utilizzare `Ripser` o `Kepler Mapper` e da dove prelevare i dati, ossia se da un Dataset caricato nella Ricerca in cui si sta creando l'analisi o utilizzando delle matrici di distanza precomputate salvate tra i file locali dell'utente. A seconda delle proprie scelte viene mostrato il `Form` appropriato.

Analisi utilizzando un Dataset

Nella creazione di una nuova Analisi utilizzando come fonte di dati un Dataset caricato precedentemente, il Form richiede di selezionare il Dataset e attraverso l'uso del Javascript è possibile, senza cambiare pagina, visualizzare il Dataset selezionato in una finestra a scomparsa all'interno del Form. Questo permette di poter rivedere cosa si è selezionato e in particolare di rivedere il numero di righe e colonne del Dataset. Ciò è necessario perchè le impostazioni successive richiedono di inserire, se si vuole, la dimensione delle finestre in cui suddividere il Dataset (`Window size`), e in tal caso anche di impostare quanto le finestre si sovrappongono in numero di colonne(`Window overlap`). Per esempio una Window size di 200 e una Window overlap di 10 vuole dire che il Dataset è suddiviso in finestre da 200 colonne e ogni finestra seguente alla prima inizia con le ultime 10 colonne della finestra precedente. Se non viene compilato il campo Window size, viene creata un' unica finestra comprendente tutto il dataset e l'opzione Window overlap non viene presentata. Nel caso invece che la combinazione di Window overlap e Window size lasci una certa quantità di colonne al termine del dataset non facenti parte di alcuna finestra (perché in numero minore rispetto alla dimensione di una finestra) queste vengono saltate

dalla computazione. È compito dell'utente di stabilire un dimensionamento delle finestre adatto per ricoprire l'intero Dataset. A questo punto è necessario selezionare la metrica da usare per convertire ogni finestra nella rispettiva matrice di distanza, impostando il parametro `Distance matrix metric`. Prima di applicare la conversione, i dati di ogni finestra sono trasposti.

Nel caso di una Filtration Analysis, il campo `Filtration Type` serve a scegliere tra Vietoris-Rips Filtration e Clique Weight Rank Filtration. Nel caso di quest'ultimo, la metrica selezionata in `Distance matrix metric` viene ignorata e viene invece calcolata la matrice di correlazione.

Analisi utilizzando matrici precomputate

Nella creazione di una nuova Analisi utilizzando come fonte di dati delle matrici precomputate, il Form richiede semplicemente di caricare i file necessari, dove ogni file corrisponde a una matrice. È possibile caricare un qualsiasi numero di file, a patto che la dimensione totale non superi il limite di 100MB e che il sistema operativo nel quale il server è in esecuzione supporti di avere aperti quel numero di file contemporaneamente (su GNU/Linux, si può modificare questa impostazione col comando `ulimit`). I file caricati devono essere in un formato comprensibile dalla funzione `numpy.loadtxt()`, e quindi generalmente può accettare matrici quadrate salvate con `numpy.savetxt()`. Ogni file caricato viene fatto corrispondere con una finestra durante l'esecuzione dell'Analisi, e le finestre sono ordinate seguendo l'ordine nel quale i file corrispondenti sono stati caricati, che corrisponde al tipo di ordinamento nella cartella dove i file risiedono i.e. se i file sono ordinati per nome nella cartella locale, allora vengono caricati e quindi analizzati in quell'ordine. È consigliabile dare dei nominativi numerici ai propri file e quindi ordinarli localmente per nome prima di caricarli. Imporre un ordinamento lessicografico non sarebbe stato pratico perché file con nomi numerici sarebbero ordinati in modo inaspettato (e.g. 1 11 12 111 2 21 22). Una volta caricati i file, essi vengono letti uno ad uno ed i loro contenuti concatenati ad una stringa JSON che verrà salvata sul database e che verrà poi usata come sorgente di dati per l'Analisi. I file quindi non risiederanno in alcuna cartella sul Server, e una volta che i loro contenuti sono stati copiati vengono rimossi dalla memoria volatile.

Filtration Analysis

I parametri per creare una Filtration Analysis ricalcano i parametri della funzione `ripser()` della libreria `Ripser` ad eccezione dei parametri `distance_matrix` e `metric` che non risultano necessari dato che la matrice di distanza viene calcolata nel passaggio precedente. La libreria Ripser offre alternativa-

mente le stesse funzionalità della funzione `ripser()` tramite la classe `Rips()` ma utilizzando la funzione è più semplice salvare i risultati: utilizzando la classe bisogna accedere a diversi attributi di istanza mentre la funzione ritorna direttamente un dizionario con tutti i risultati, semplificando quindi il salvataggio dei dati nel database. I parametri da impostare presenti nel Form sono:

- `Max homology dimension`, che corrisponde a `maxdim` e indica la massima dimensione di omologia che verrà calcolata;
- `Max distances considered`, che corrisponde a `thresh` e indica la massima distanza considerata durante la filtrazione;
- `Coeff` per specificare i coefficienti nel campo Z/pZ con $p = coeff$;
- `Do cocycles` per specificare se calcolare i cocicli;
- `N perm` per specificare il numero di punti da utilizzare in una 'permutazione golosa' . Questi punti sono utilizzati al posto del numero totale di punti per una computazione più rapida al costo di un po' di precisione.

Una volta impostati tutti i parametri e confermata la creazione, per ogni finestra viene chiamata la funzione `ripser()` con come primo parametro `X` la matrice di distanza della finestra e poi passando gli altri parametri appena elencati.

Mapper Analysis

I parametri per creare una Mapper Analysis si basano sui parametri delle funzioni `fit_transform()` e `map()` e sugli argomenti per la costruzione di una istanza di `Cover` e `GraphNerve`.

I parametri della funzione `fit_transform()` che sono richiesti dal Form sono `projection` e `scaler`. Per `projection` i valori selezionabili sono una delle stringhe disponibili nella documentazione di Kepler Mapper[1]. Se viene selezionato `knn_distance_n` appare un nuovo campo dove si può inserire il numero da inserire al posto di n. Per quantor riguarda `scaler` sono disponibili tutti gli scaler della libreria `Scikit-Learn`, seppur solo utilizzando i costruttori di default. Il parametro `distance_matrix` non viene utilizzato perchè la matrice di distanza viene già computata in un passaggio precedente mentre il parametro `X` sarà la matrice di distanza di una finestra.

I parametri dlla funzione `map()` che sono richiesti dal Form sono:

[1] `https://kepler-mapper.scikit-tda.org/reference/index.html`

- `Use original data`, che corrisponde a `X` e specifica se applicare l'algoritmo di clustering sulla matrice originale (ossia prima del calcolo della matrice di distanze) o sulla matrice risultato di `fit_transform`.

- `Clusterer` che specifica l'algoritmo di clustering da utilizzare. Sono presenti tutti gli algoritmi di clustering della libreria `Scikit-Learn`, seppur con i costruttori di default. L'unica eccezione è DBSCAN, che è presente anche nella versione con `min_samples = 1`. Questa impostazione può risultare utile perché l'algoritmo mapper può fallire per valori troppo elevati di questo parametro, che di default in Kepler Mapper è impostato a 3.

- `Cover n cubes` e `Cover perc overlap` servono per costruire un oggetto di tipo `Cover` e corrispondono a 2 dei 4 parametri del costruttore. Il terzo parametro non è stato possibile aggiungerlo al Form per via della sua complessità e il quarto non riguarda il contesto di una applicazione visuale. Il parametro `Cover n cubes` specifica il numero di ipercubi per ogni dimensione, mentre `Cover perc overlap` è la percentuale di sovrapposizione tra cubi adiacenti calcolata solo lungo un'unica dimensione.

- `Graph nerve min intersection` serve per costruire un oggetto di tipo `GraphNerve` e corrisponde al parametro `min_intersection` del costruttore. Il suo valore rappresenta la minima intersezione considerata quando viene calcolato il nervo. Un arco viene creato solo se l'intersezione tra due nodi è maggiore o uguale a questo valore.

- `Precomputed` se selezionato, indica che i dati su cui si sta eseguendo il clustering è una matrice precomputata.

- `Remove duplicate nodes` indica di rimuovere i nodi duplicati prima che gli archi vengano determinati. Un nodo è considerato un duplicato se ha sattamente lo stesso insieme di punti di un altro nodo.

I parametri `nr_cubes` e `overlap_perc` non sono stati aggiunti perché deprecati, mentre il parametro `lens` sarà l'output di `fit_transform()`.

Una volta impostati tutti i parametri e confermata la creazione, per ogni finestra viene chiamata la funzione `Keplermapper().fit_transform()` con come primo parametro `X` la matrice di distanza della finestra e poi passando gli altri parametri appena elencati. Quindi viene chiamata la funzione `KeplerMapper().map()` con come primo parametro `lens` l'output della precedente funzione e poi i parametri descritti sopra. Infine, chiamando

la funzione `KeplerMapper().visualize()` otteniamo una stringa HTML che mostra i risultati dell'Analisi.

6.3.2 Visualizzazione di una Analisi

Nella pagina di visualizzazione di una Analisi, viene mostrato un sommario che indica la Ricerca di cui fa parte, il Dataset, se presente, da cui sono stati prelevati i dati, ed i parametri impostati per l'esecuzione dell'Analisi. Da questa pagina è anche possibile eliminare l'Analisi e visualizzarne i risultati.

6.3.3 Risultati di una Analisi

Una Analisi è eseguita su ogni finestra, quindi per visualizzare i risultati di una Analisi bisogna prima scegliere di quale finestra visualizzarli. Ogni finestra contiene i risultati dell'Analisi su di essa e corrisponde ad una entità "Finestra" salvata nel database. Ad ogni Finestra è associato un valore numerico che la identifica univocamente nel contesto di una Analisi; tale valore numerico è il valore di sequenza, partendo da 0, nella successione delle finestre, dove la prima Finestra ha valore 0, la seconda valore 1 e così via.

Nella visualizzazione dei risultati è quindi riportato l'elenco delle finestre dell'Analisi in questione, identificate dal loro valore numerico, e si deve selezionare quale finestra visualizzare.

Finestre di Filtration Analysis

Nella visualizzazione di una Finestra di Filtration Analysis vengono riportati gli indici di inizio e di fine della Finestra nel Dataset, se è stato utilizzato un Dataset. Inoltre viene mostrato il grafico dei diagrammi di persistenza, mostrando in un unica immagine (scaricabile in formato PNG) i valori per ogni omologia. Infine, sono mostrati i valori di entropia, sia normalizzati che non, per ogni omologia.

L'output completo dell'Analisi è scaricabile, e ciò include, oltre ai diagrammi di persistenza, una serie di altri valori come descritto nella documentazione di Ripser.[2] Il formato in cui il file viene scaricato è `.dat` ed è interpretabile da Python come un dizionario. Per fare ciò, è consigliabile utilizzare la funzione `literal_eval()` del modulo Python `ast`.

[2] `https://ripser.scikit-tda.org/reference/stubs/ripser.ripser.html`

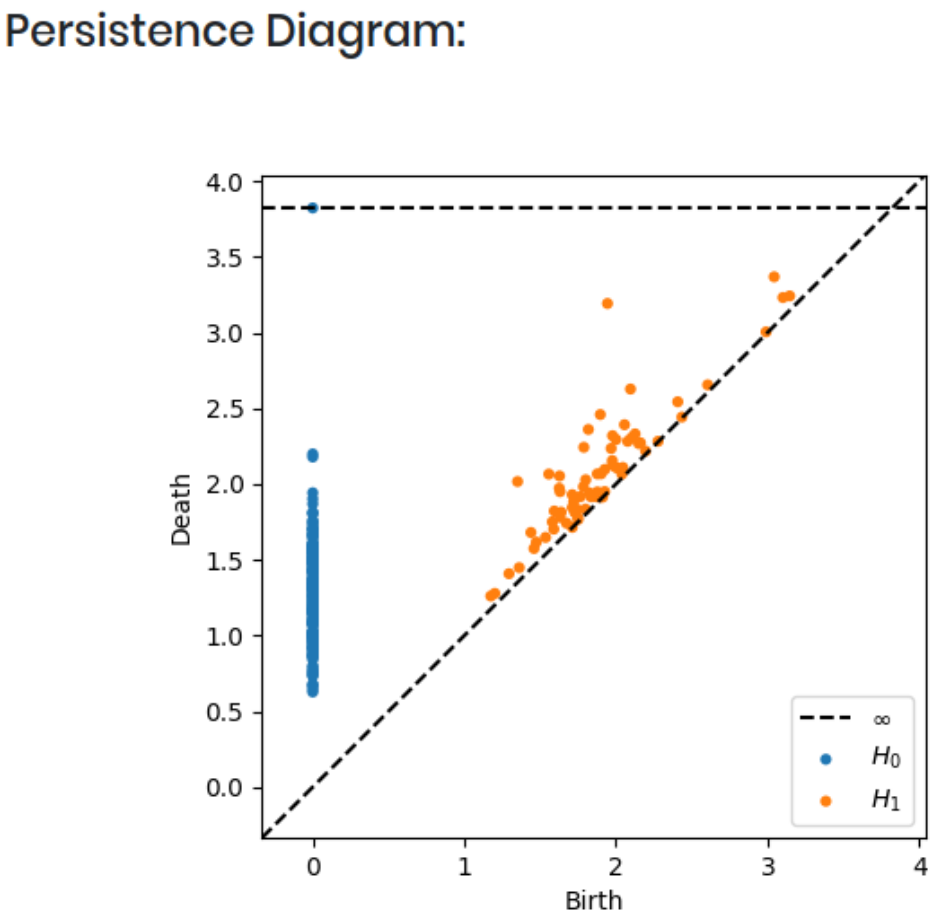

Figure 6.2: Diagramma di persistenza di omologia 0 e 1.

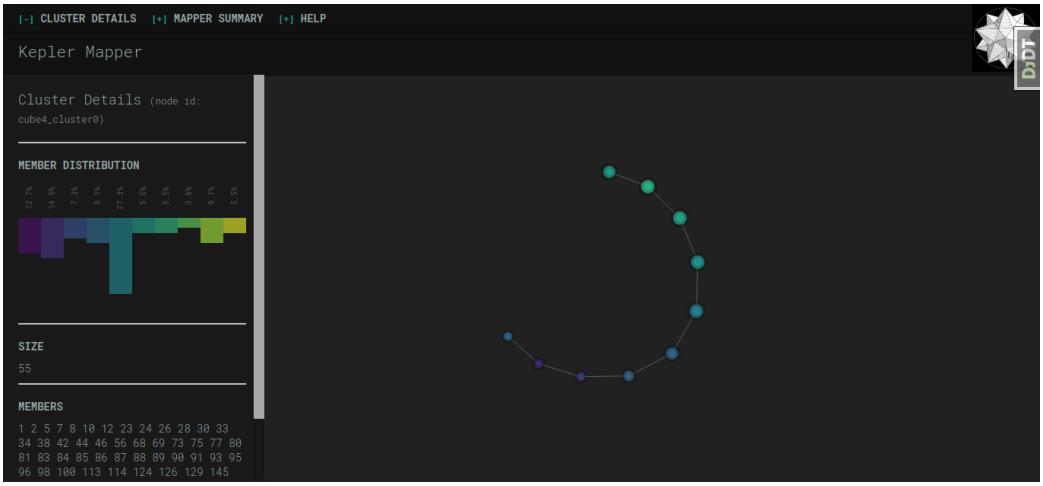

Figure 6.3: Pagina HTML col risultato dell'Analisi di Kepler Mapper

Finestre di Mapper Analysis

Nella visualizzazione di una Finestra di Mapper Analysis vengono riportati gli indici di inizio e di fine della Finestra nel Dataset, se è stato utilizzato un Dataset. Il risultato dell'Analisi viene invece mostrato in una nuova scheda dato che è a tutti gli effetti una pagina HTML a sé stante. In questa pagina viene mostrato interattivamente il grafo costruito dall'algoritmo MAPPER e sono riportati dettagli riguardo il grafo, ogni nodo e un resoconto sui parametri dell'Analisi così come sono anche riportati nella `Visualizzazione di una Analisi` descritta precedentemente.

6.3.4 Entropia

Nel caso di una Filtraion Analysis, nella pagina di visualizzazione dell'Analisi viene anche mostrato un grafico dell'entropia. L'entropia viene calcolata su ogni finestra facente parte di una data Filtration Analysis e per ogni valore di omologia. I valori di entropia sono calcolati durante l'esecuzione di una Analisi, subito dopo che una finestra è stata valutata.

Tra gli output della funzione `ripser()`, il più importante sono i diagrammi di persistenza (uno per ogni omologia), ognuno dei quali è espresso come lista di coppie di valori. Ogni coppia di valori rappresenta il livello di filtrazione nel quale l'elemento appare (nascita) e il livello nel quale scompare(morte). Per calcolare l'entropia di un diagramma di persistenza calcoliamo per ogni coppia (nascita, morte) un valore li nel modo seguente:

$$li = morte - nascita$$

dove nel caso il valore di *morte* sia infinito, esso è sostituito col massimo valore non infinito di *morte* nel diagramma di persistenza sommato ad 1. Successivamente calcoliamo un valore *ltot* come:

$$ltot = \sum li$$

Infine calcoliamo l'entropia con la formula:

$$entropia = -\sum \frac{li}{ltot} \cdot \log_{10} \frac{li}{ltot}$$

Questo valore di entropia è tuttavia non normalizzato. Oltre a questo valore di entropia, viene quindi anche calcolato un valore normalizzato con la seguente formula:

$$entropia\_normalizzata = \frac{1}{\log_{10}(\#li)} \cdot -\sum \frac{li}{ltot} \cdot \log_{10} \frac{li}{ltot}$$

dove con $\#li$ si intende il numero di coppie presenti nel diagramma di persistenza e al valore $\log_{10}(\#li)$ viene sostituito 1 qualora il valore del logaritmo sia 0.

Per visualizzare il grafico dell'entropia, vengono prelevate tutte le finestre dell'Analisi in ordine dal database, quindi da ogni finestra sono presi i valori di entropia normalizzati e non per ogni omologia e visualizzati in due grafici: uno per l'entropia non normalizzata e uno per quella normalizzata. I grafici presentano punti per ogni valore e linee per connettere i punti, così che siano visualizzabili anche nel caso ci sia una sola finestra nell'Analisi.

I valori di entropia sono scaricabili in formato CSV in un unico file contenente sia la versione normalizzata che non per ogni omologia.

6.3.5 Distanza di bottleneck

Nel caso di una Filtration Analysis è possibile calcolare la cosiddetta distanza di bottleneck tra due diagrammi di persistenza. La distanza di bottleneck misura la somiglianza tra due diagrammi di persistenza. La somiglianza è calcolata come la più breve distanza b per la quale esista una corrispondenza esatta tra i punti dei due diagrammi (considerando anche tutti i punti sulla diagonale) così che per ogni coppia di punti messi in corrispondenza la distanza è al massimo b.[3]

Sono disponibili 3 tipi di calcolo di distanza di bottleneck: si può calcolare tra ogni finestra e il suo successore, tra una data finestra e tutte le altre,

[3] `https://rdrr.io/cran/TDA/man/bottleneck.html`

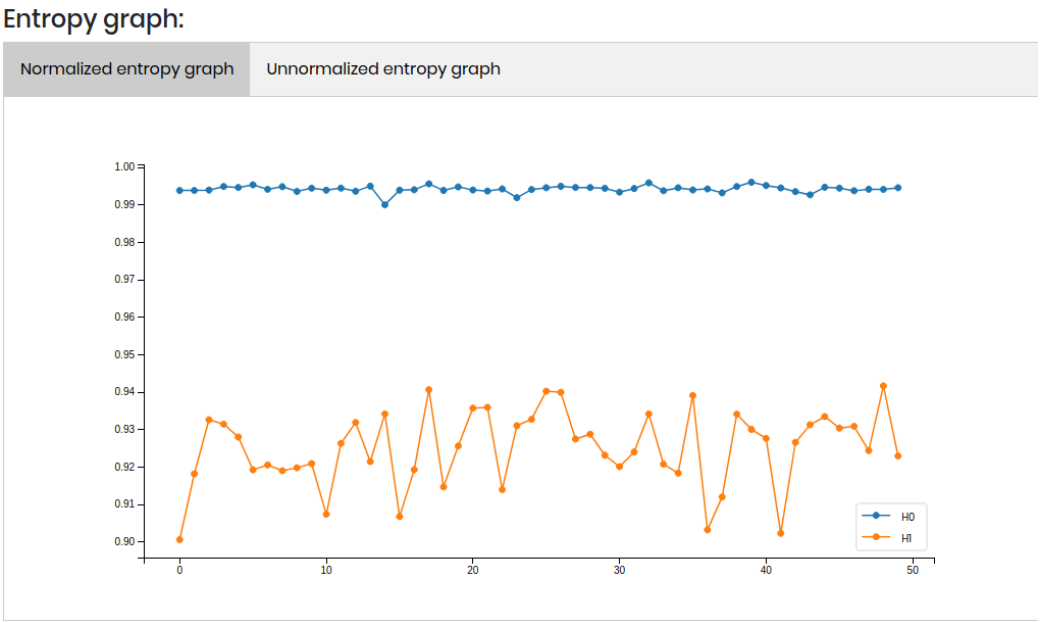

Figure 6.4: Il grafico di entropia.

e tra ogni finestra verso ogni finestra. Ovviamente per ognuna di queste computazioni si deve scegliere l'omologia dei diagrammi di persistenza da usare. Per il calcolo della distanza di bottleneck tra una data finestra e tutte le altre, si deve prima navigare fino alla finestra in questione e da lì accedere al Form della distanza di bottleneck, il quale chiede semplicemente di selezionare il numero di omologia. Negli altri due casi invece il al Form si accede attraverso la pagina di Analisi, e nel Form oltre all'omologia viene richiesto quale dei due tipi di calcolo della distanza di bottleneck eseguire.

Ogni calcolo della distanza di bottleneck viene salvato sul database come entità "Bottleneck" contenente informazioni sul tipo di calcolo effettuato, omologia e a quale Analisi o Finestra si riferisce. Prima di effettuare qualsiasi calcolo viene verificato nel database se quello stesso calcolo sia già stato eseguito, e in questo caso viene direttamente ritornato il risultato senza ripetere il calcolo. Per ogni coppia di diagrammi di persistenza sui quali viene eseguito il calcolo, la funzione di Persim `bottleneck()` ritorna il valore di distanza e informazioni aggiuntive da passare alla funzione `plot_bottleneck()` per disegnare un grafico relativo alla distanza di bottleneck. Per ogni coppia di diagrammi che vengono confrontati il valore di distanza e il grafico sono salvati nel database su un entità chiamata "Diagramma" collegata via chiave esterna all'entità Bottleneck che l'ha generata.

Nel caso del calcolo della distanza di bottleneck tra ogni finestra verso

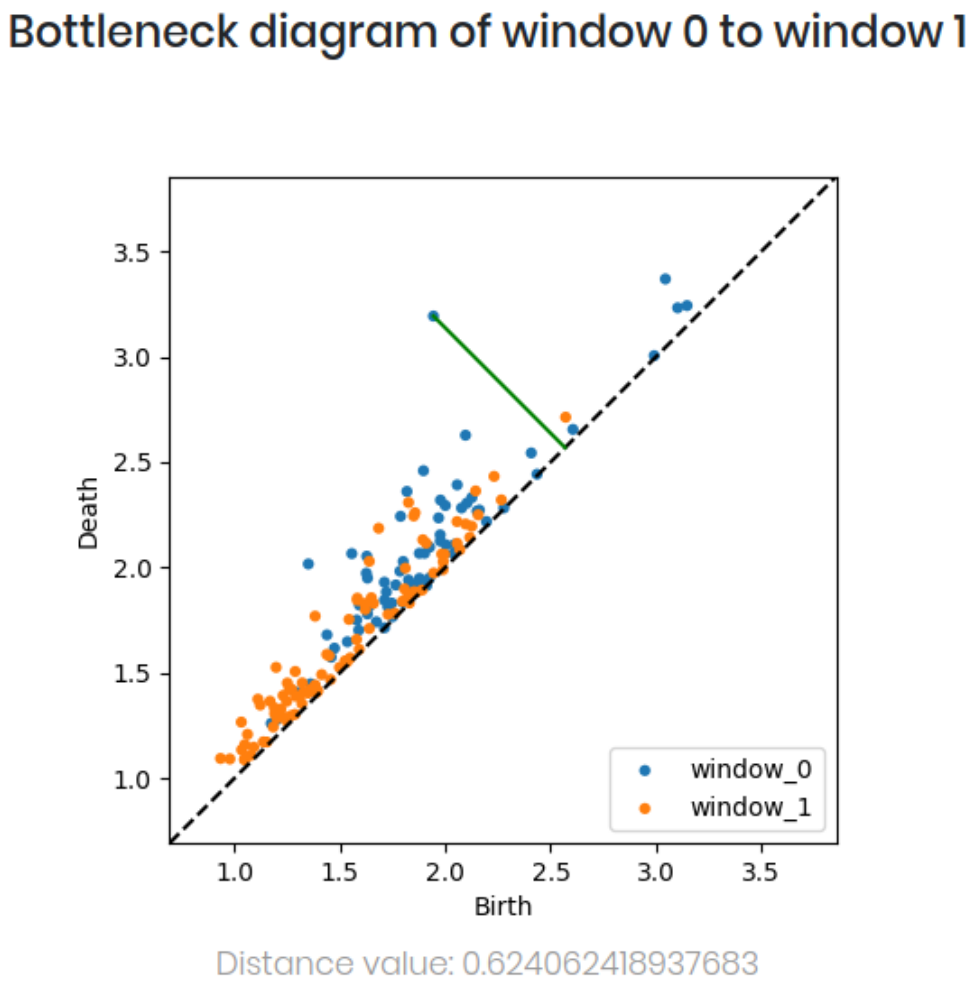

Figure 6.5: Un grafico che mostra la distanza di bottleneck nell'omologia 1.

ogni finestra, non è necessario effettuare tutti gli n^2 confronti, dato che il valore di distanza è invariante rispetto all'ordine con cui si confrontano le finestre. In realtà in questo caso è necessario solamente fare $\frac{n \cdot (n+1)}{2}$ confronti, ossia la finestra ennesima viene confrontata solamente con finestre di valore numerico pari(ossia con se stessa) o superiore.

Nella schermata di visualizzazione dei risultati, sono riportate tutte le distanze calcolate e tutti i grafici di confronto generati. È possibile scaricare le distanze calcolate in formato CSV. Nel caso del calcolo tra ogni finestra verso ogni finestra il file CSV contiene l'intera matrice quadrata, duplicando il triangolo superiore (quello che è stato effettivamente calcolato) nel triangolo inferiore. Inoltre è anche possibile cancellare il risultato, con l'effetto di

rimuovere dal database l'entità Bottleneck e tutti i Diagrammi associati.

Gestione degli errori nella libreria Persim

Il calcolo della distanza di bottleneck avviene utilizzando funzioni della libreria Persim. Tuttavia la funzione `bottleneck()` presenta dei problemi in alcuni casi particolari e lancia una eccezione. I casi problematici che fin'ora sono stati individuati sono:

- Confronto di un diagramma di persistenza con se stesso
- Confronto tra due diagrammi di persistenza dove almeno uno dei due è vuoto

Il primo caso non è sempre fonte di errore, ma lo è spesso. Solitamente il problema avviene quando il diagramma ha un solo punto ma il problema può presentarsi anche in altre occasioni e non è chiaro cosa provoca il problema. In ogni caso gestire questo problema è triviale, impostando manualmente il valore di distanza di bottleneck a 0 e visualizzando come grafico il diagramma di persistenza immutato. Nel secondo caso il confronto viene saltato.

7 | Front-end

Il layout dell'applicazione si può generalmente suddividere in 3 parti: una barra laterale, una barra superiore e la sezione dei contenuti.

La barra laterale, oltre a contenere il nome dell'applicazione, presenta 3 bottoni: `Navigation`, `About` e `Contact`. Cliccando su Navigation, si apre un menù a tendina che mostra in sequenza la navigazione sul sito: come primo link è sempre presente un bottone per ritornare alla Home, poi, a seconda di dove ci si trova, possono essere presenti link alla pagina principale di una Ricerca, di un Dataset, di una Analisi o di una Finestra. Per esempio, se si sta visualizzando la Finestra 0 dell'Analisi Test nella Ricerca Prova, saranno presenti, in sequenza, i link Home, Research: Prova, Analysis: Test e Window: 0. Il bottone About mostra una pagina che presenta una breve descrizione dell'applicazione e un collegamento ipertestuale alla presente relazione. Il bottone Contact mostra una pagina con gli indirizzi email istituzionali degli sviluppatori.

La barra superiore viene utilizzata come barra delle attività: in un dato momento presenta bottoni corrispondenti alle operazioni che si possono svolgere nel contesto del contenuto che si sta visualizzando. Nella parte più a sinistra è presente un bottone `Toggle` che ha come unico utilizzo contrarre o espandere la barra laterale descritta precedentemente. Invece, a partire da destra, è presente un bottone `Back` che porta l'utente nella pagina immediatamente precedente e a seguire le attività di disponibili. Per esempio nella pagina principale l'unica attività disponibile è visualizzare la lista delle ricerche, mentre nella pagina di una Ricerca è possibile visualizzare la lista dei Dataset, la lista delle Analisi e cancellare la Ricerca.

La sezione dei contenuti presenta il contenuto vero e proprio della pagina che si sta visualizzando, che sia una lista di Ricerche, Dataset o Analisi, i dettagli di una Analisi o di un Dataset, oppure il risultato di una Analisi su una certa Finestra. Sempre in questa sezione è possibile compilare Form di creazione di una nuova Ricerca, aggiunta di un Dataset, o creazione di una nuova Analisi.

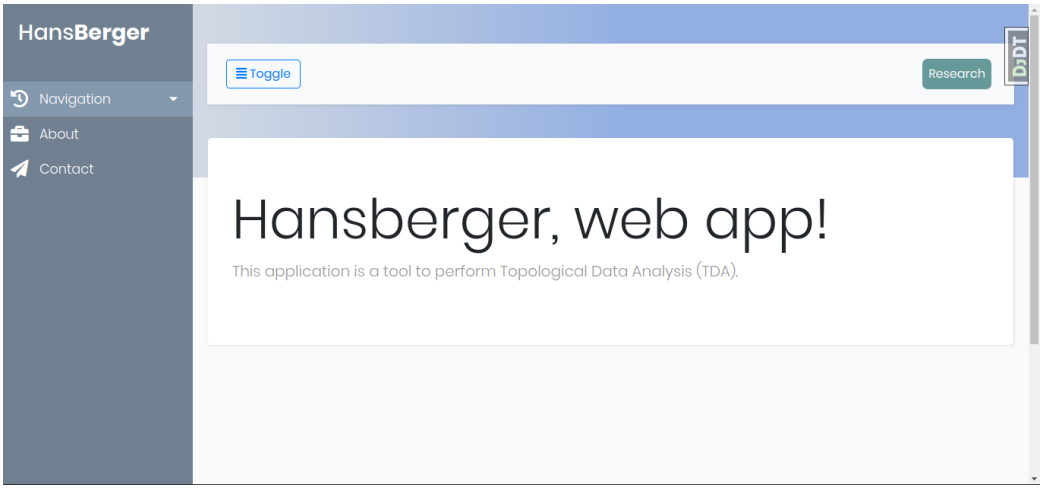

Figure 7.1: La pagina principale dell'applicazione.

7.1 Schermata di caricamento

Nella pagina del Form di creazione di una nuova Analisi o di calcolo della distanza di bottleneck è presente uno script che blocca la pagina mentre il back-end sta eseguendo l'operazione (che può richiedere da qualche millesimo di secondo a svariate ore). Mentre la pagina è bloccata viene mostrato un messaggio di caricamento che mostra la progressione dell'esecuzione. Questa progressione è mostrata in numero di finestre valutate sul numero totale di finestre su cui operare, ossia appare nella forma `Processing window n/N` dove n sta per il numero di finestra corrente e N il numero totale di finestre. Inoltre sotto al messaggio di caricamento è presente un bottone `Abort`. Cliccando su questo bottone l'esecuzione viene interrotta ma i risultati fino ad ora ottenuti sono comunque salvati sul database e visualizzabili.

Questa funzionalità è stata implementata mediante il protocollo Web-Socket. Quando una di queste operazioni potenzialmente onerose viene lanciata, il front-end invia ad intervalli regolari di 1s una richiesta di status al back-end utilizzando il formato JSON. Il back-end risponde nello stesso formato riportando il numero corrente di finestra e il numero totale di finestre. Quando e se poi l'utente clicca il bottone Abort, un messaggio JSON che codifica l'intenzione di interrompere l'esecuzione viene mandato al back-end, che provvede di interrompere come richiesto semplicemente ritornando dalla funzione che sta eseguendo il calcolo. Il pezzo mancante in questo meccanismo è come far comunicare la parte di back-end che riceve i segnali dal front-end con la parte back-end che sta eseguendo le operazioni, visto che sono in esecuzione su thread diversi. La soluzione pragmatica a ciò è stato

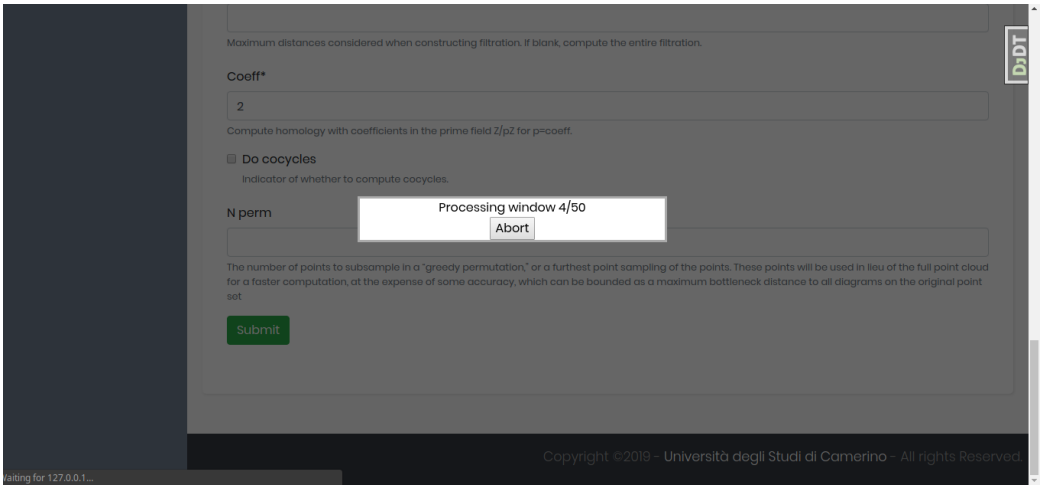

Figure 7.2: La schermata di caricamento.

implementare una classe che istanzia il pattern Singleton e permettere a entrambi i thread di accedervi. Il thread di esecuzione di operazioni scrive il suo status ed eventualmente legge se deve interrompere, mentre l'alto legge lo status e scrive se l'altro thread deve interrompersi. L'inconveniente di questo approccio è che per una corretta esecuzione in un dato momento deve esserci al massimo una di queste operazioni in corso, perché dato che questo meccanismo di comunicazione si basa su un solo oggetto, multiple operazioni finirebbero col modificare concorrentemente i suoi contenuti. La soluzione a ciò sarebbe implementare un meccanismo dove per ogni operazione in corso è presente un differente oggetto che ne conserva lo stato, identificato da un codice.

8 Limitazioni

8.1 Parametri di Kepler Mapper

Le funzioni offerte da `Kepler Mapper` accettano parametri in input che vanno oltre a quanto offerto dalla nostra applicazione. Ad esempio il parametro `projection` può accettare una qualsiasi classe di `Scikit-learn` con un metodo `fit_transform`, cosa che al momento la nostra applicazione non permette. Un altro esempio è che alcuni parametri, come `clusterer`, accettano una istanza di classe che può essere impostata con degli specifici parametri nel costruttore, mentre nella nostra applicazione usiamo i costruttori di default. Non sarebbe possibile permettere tutta la varietà di scelta possibile senza complicare eccessivamente l'interfaccia di creazione di una Analisi quindi ci siamo accontentati di ciò.

8.2 Memoria

Le Analisi svolte su Dataset di grosse dimensioni, in particolare in formato EDF, hanno grossi requisiti di memoria dinamica. I problemi sono stati riscontrati nelle fasi di creazione dell'Analisi, creazione del grafico di entropia e calcolo delle distanze di bottleneck.

In fase di creazione di Analisi, se non si utilizzano finestre o le finestre sono eccessivamente grandi è molto facile imbattersi nel limite della memoria e il sistema va in crash. Utilizzando finestre di dimensione consona sono comunque stati riscontrati problemi, ossia il freeze del computer, causato dal progressivo aumento della memoria utilizzata fino a saturazione della stessa. Questo problema teoricamente non dovrebbe avvenire, perché ogni finestra, dopo essere stata creata e scritta nel database, non viene più referenziata e quindi la memoria dovrebbe essere stata liberata dal garbage collector. Ciò non avviene perché Django utilizza il meccanismo delle transazioni database per ogni richiesta HTTP che riceve, in altre parole durante la creazione delle Analisi e contestualmente la creazione di tutte le finestre dell'Analisi, la

scrittura su database avviene solo al termine e se non ci sono stati errori, come è tipico dell'architettura commit/rollback delle transazioni. La soluzione a questo problema è data disattivando questo meccanismo per quanto concerne la creazione di Analisi. Malgrado ciò, il consumo di memoria si vede comunque aumentare ma i problemi riscontrati precedentemente vengono evitati richiamando esplicitamente il garbage collector.

I problemi dati nella fase di creazione del grafico di entropia e calcolo di distanze di bottleneck sono dati dal fatto che queste computazioni richiedono di caricare in memoria tutte le finestre dell'Analisi, che da come si può intuire in certi casi la memoria volatile non è sufficiente e ciò provoca nuovamente il freeze del computer. La soluzione in questo caso è data dal batching delle finestre, ovvero suddividere le finestre in lotti ed operare su un lotto alla volta. Questo in Python è possibile utilizzando il costrutto dei generatori, che possono pigramente restituire un lotto di finestre alla volta. La dimensione di un lotto è stata euristicamente impostata a 1/16 del numero di finestre totali.

Con queste accortezze è stato possibile analizzare (con i dovuti tempi) un Dataset in formato EDF di 23 righe e 921 000 colonne con finestre da 100 (9 216 finestre) o 200 (4 608 finestre) colonne.

8.3 Connessione ad Internet

Una connessione ad Internet non è strettamente necessaria per il funzionamento dell'applicazione dato che il server è in esecuzione localmente, tuttavia la libreria `mpld3` utilizza una libreria JavaScript non locale per la funzione `fig_to_html()`[1] quindi per la visualizzazione dei Dataset e dell'entropia una connessione è necessaria.

[1] `https://mpld3.github.io/faq.html`

9 Estensioni

9.1 Analisi in parallelo

Una possibile estensione del sistema è lo sviluppo di un meccanismo per eseguire più Analisi contemporaneamente in modo asincrono. Attualmente il parallelismo dello svolgimento delle Analisi è già presente e insito nell'architettura Client-Server, tuttavia le Analisi sono comunque eseguite immediatamente in risposta a una richiesta del Client, bloccando la pagina corrente e senza scheduling o prioritizzazione. Peraltro, dato che tutti i grafici si basano su `Pyplot` e che in alcuni casi il suo utilizzo avviene indirettamente attraverso altre funzioni di altre librerie utilizzate, è una attività non banale rendere possibile disegnare grafici diversi contemporaneamente. Il modulo `Pyplot` offre un meccanismo per farlo, ma implementarlo in tutti i grafici richiede anche di andare a modificare la funzione di `Persim` che disegna i diagrammi di persistenza e `Pandas` che visualizza i file CSV per utilizzare tale meccanismo. Infine, come descritto precedentemente, la barra di caricamento e il bottone di annullamento si basano su un unica istanza e quindi andrebbe implementato un meccanismo per avere una istanza per ogni analisi che viene eseguita parallelamente. Superata questa difficoltà, si può implementare questo tipo di parallelismo utilizzando la libreria python `Celery` accoppiata con lo storage `Redis` per il passaggio di messaggi.

9.1.1 Celery

Celery è una libreria Python che implementa una coda di attività asincrona basata sul passaggio distribuito di messaggi. Le attività vengono eseguite in modo concorrente su uno o più `worker` servers utilizzando multiprocessing, `Eventlet` o `getevent`. Le attività possono essere eseguite in background in modo asincrono o parallelamente in modo sincrono, attendendo il loro termine.[1]

[1] `http://www.celeryproject.org/`

9.1.2 Redis

Redis è uno store di strutture dati chiave-valore in memoria rapido e open source. Redis offre una serie di strutture dati in memoria molto versatili, che permettono di creare un'ampia gamma di applicazioni personalizzate. I principali casi d'uso per Redis sono il caching, la gestione di sessioni, servizi pub/sub e graduatorie. Si tratta dello store chiave-valore più usato. Dispone di licenza BSD, è scritto in C, il suo codice è ottimizzato e supporta diverse sintassi di sviluppo. Redis è un acronimo e sta per REmote DIctionary Server.[2] I suoi punti di forza sono:

- estrema velocità: Redis conserva i dati in memoria RAM, salvandoli in maniera persistente solo in un secondo momento. Ciò permette di ottenere ottime prestazioni in scrittura e lettura;
- dispone di una grande varietà di tipi di dato. Quindi nonostante l'architettura dell'archivio sia basata su una struttura a dizionario, i valori possono assumere varie forme: liste, dizionari stessi, e molto altro. Da questo punto di vista, Redis può essere visto come un gestore di strutture dati persistenti;
- tutte le operazioni sono atomiche, quindi in caso di accessi concorrenti da parte di più client, i dati forniti risulteranno sempre aggiornati;
- possibilità di implementare configurazioni multi-nodo, cluster e replication.

9.2 Deploy su Server remoto

Per effettuare il deployment dell'applicativo è necessario, per questioni di sicurezza, l'utilizzo di AWS (Amazon Web Service).

Mentre nella configurazione locale tutti i media vengono salvati localmente, in quella "production" si richiede obbligatoriamente l'uso di un servizio esterno, da configurare secondo le proprie esigenze.

9.3 Sistema Utenti

E' possibile integrare facilmente un sistema utenti all'interno dell'applicativo, essendo già predisposto nel codice. Un suo utilizzo naturale risiederebbe nella suddivisione delle ricerche. Ogni utente potrebbe avere il suo set di ricerche

[2] `https://redis.io/`

privato, ed, eventualmente, accedere a datasets pubblici per effettuare le proprie analisi.

9.4 Documenti

Le ricerche sono contenitori con infinite possibilità di espansione. Allo stato attuale dell'applicativo queste contengono unicamente datasets e analisi, ma in futuro potrebbe presentarsi la necessità di associarvi della documentazione. Mettendo in relazione il modello ricerca con un eventuale nuovo modello "Documento" sarà possibile aggiungere questa feature, permettendo agli utenti di caricare varie tipologie di file e visualizzarli online.

Bibliography

[ELZ02] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28(4):511–533, 2002.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[TSBO18] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018.

[vVS19] Hendrik Jacob van Veen and Nathaniel Saul. Keplermapper. http://doi.org/10.5281/zenodo.1054444, Jan 2019.