Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

SRAE: Sustainable Regressive Auto Encoder

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Riccardo Preite

Correlatore:
Dott. Andrea Zanellati

Sessione I
Anno accademico 2021/2022

Abstract

Al giorno d'oggi viviamo in una realtà dove lo *sviluppo economico*, l'*innovazione tecnologica*, la *qualità della vita* e l'*impatto ambientale* sono i protagonisti assoluti. Tutti, persino gli Stati del mondo, si trovano a fare i conti con varie problematiche riguardanti i quattro aspetti sopracitati e qui possiamo dire che la *sostenibilità* ne è il punto chiave e che al momento non sembra esistere ancora una metrica riconosciuta e approvata per consigliare, a chi di interesse, come modificare certi aspetti per crescere in modo *sostenibile*. Le Nazioni Unite hanno deciso, di comune accordo, di stilare una lista di obiettivi da raggiungere entro il 2030 dove è possibile trovare argomenti in linea con quanto descritto finora. Questa raccolta è principalmente divisa in aspetti *economici*, *sociali* e *ambientali* che sono le stesse categorie di dati impiegate per il calcolo del *Sustainable Development Index*[1]. In questo elaborato ci si propone di progettare e sviluppare una rete neurale predittiva da affiancare a un sistema di feedback per realizzare un prodotto che sia abile di: descrivere il contesto di partenza tramite l'SDI e/o consigliare comportamenti per migliorare la situazione in modo sostenibile. Il codice dell'applicazione è disponibile al seguente link: SRAE.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo elaborato viene descritto il lavoro svolto riguardo alla combinazione degli SDG[10] (Sustainable Development Goals) e tecniche di Machine Learning per creare un sistema innovativo che possa essere uno strumento ausiliare per pianificare lo sviluppo economico, sociale e/o ambientale in modo sostenibile. Le Nazioni Unite hanno deciso, di comune accordo, di stilare una lista di obiettivi da raggiungere entro il 2030 principalmente divisa in aspetti *economici*, *sociali* e *ambientali* dove purtroppo i dataset forniti non sono completi e presentano dei valori nulli. Un aspetto interessante di questi SDG è che rappresentano obiettivi che potrebbero essere collegati fra loro, ad esempio un maggiore accesso ai *servizi sanitari/acqua pulita* dovrebbe ridurre la percentuale di *malattie*. Qualsiasi azienda potrebbe sottopagare le persone o usare materiali inquinanti per poter *crescere economicamente* ma ovviamente andrebbe a discapito dell'*innovazione tecnologica*, in quanto si dovrebbero usare tecniche antiquate, e l'*impronta* che lasceremmo sul pianeta aumenterebbe soltanto, infine la *qualità della vita* ne verrebbe ridotta sia per il peggioramento dell'ambiente sia per la riduzione della paga.

Quanto si vuole raggiungere con questo lavoro di tesi è la creazione di un sistema di raccomandazioni che sia in grado di dare consigli su come crescere economicamente, socialmente o ambientalmente in modo sostenibile sfruttando le suddette correlazioni. Si vuole dare la possibilità agli utenti di creare un

sistema personalizzato, scegliendo quali aspetti dell'Agenda2030 considerare, dove possano richiedere al sistema che comportamenti adottare per raggiungere un determinato obiettivo. Ad esempio *Quanto dovrebbe cambiare il PIL di uno Stato per aumentare la qualità di vita dei cittadini?* Per ottenere quanto detto si vuole progettare un'Intelligenza Artificiale seguendo l'idea delle Deep Neural Network per definire gli strati della rete. Ci si propone di realizzare un modello capace di assimilare o scartare eventuali feature tra i vari obiettivi dividendo il flow in due parti: nella prima sezione i vari strati si chiudono in un unico nodo, un cosiddetto indice, il quale rappresenta la *sostenibilità della situazione*; nel secondo settore invece si vuole aumentare la dimensionalità della rete fino a ricostruire l'input inserito. Questo è possibile grazie alla disponibilità di un valore target e dell'input stesso che coprono il ruolo di "ground truth" per il sistema durante la fase di allenamento: per la prima sezione si usa l'SDI[1], in inglese Sustainable Development Index, trattasi di un indice che rappresenta quanto lo sviluppo umano sia sostenibile usando dati di carattere economico, sociale ed ambientale (come quelli nell'Agenda2030); per la seconda parte, l'Auto Encoder, si usa l'input per calcolare l'errore del modello. La rete nella sua forma finale dovrà essere in grado di calcolare l'indice con nuovi dati inseriti e fare previsioni di uno o più obiettivi. Inizieremo presentando prima l'SDI e poi gli SDG, spiegando più nel dettaglio la sfida che pongono verso l'umanità tutta. Il codice dell'applicazione è disponibile al seguente link: SRAE.

1.1 SDI

Il Sustainable Development Index è il risultato di una formula che misura quanto è efficace a livello ecologico lo sviluppo umano coinvolgendo due fattori principali: HDI ed EII, rispettivamente Human development Index ed

Ecological Impact Index in inglese.

$$\text{SDI} = \frac{\text{HDI}}{\text{EII}}$$

L'Indice di Sviluppo Umano viene calcolato a sua volta sfruttando dati a livello economico e sociale: **Indice di aspettativa di vita**, **Indice di educazione** e **Indice di reddito** calcolati a comparti stagni per ogni Stato in base alle misurazioni che hanno avuto riguardo ad: **aspettativa di vita**(LE), **media di anni di istruzione**(MYS), **anni di istruzione previsti**(EYS) e **PIL**(GNI).

$$\text{Human Development Index} = \sqrt[3]{\text{Life Expectancy Index} * \text{Education Index} * \text{Income Index}}$$

$$\text{Life Expectancy Index} = \frac{\text{LE} - 20}{85 - 20}$$

$$\text{Education Index} = \frac{\text{MYSI} - \text{EYSI}}{2}$$

$$\text{Life Expectancy Index} = \frac{\ln(\text{GNIpc}) - \ln(100)}{\ln(20000) - \ln(100)}$$

Per l'Indice di Impatto Ecologico invece vengono coinvolti: il Material Footprint(quantità totale di risorse minerali e fossili impiegate per la produzione di bene e/o servizi) e le emissioni di Co2; a questi si aggiungono due valori "limite" che variano ogni anno in base alla popolazione e rappresentano il limite planetario di di Co2 e Material footprint per persona. Utilizzando questi "boundary" all'interno della formula della media geometrica è possibile impedire che uno Stato possa avvalersi di compensazioni nel caso in cui uno dei due indici sia estremamente alto e l'altro estremamente basso. Di seguito sono riportate le formule per il calcolo di tutti gli indici sopracitati e si rimanda alla pagina dell'SDI[1] per una descrizione più approfondita di quanto spiegato.

$$\text{Ecological Impact Index} = 1 + \frac{e^{\text{AO}} - e^{1}}{e^{4} - e^{1}}$$

$$if\ \text{AO} > 4, then\ \text{EII} = \text{AO} - 2$$

$$\text{AO} = \sqrt[2]{(\frac{\text{MF}}{\text{boundary}} \geq 1) * (\frac{\text{Co2}}{\text{boundary}} \geq 1)}$$

1.2 SDG challenge

La sfida lanciata dagli SDG è ancora in gioco. Attualmente chiunque potrebbe migliorare nei principali aspetti inerenti allo sviluppo sostenibile, infatti nel report delle nazioni unite del 2020[12] viene riportato che: più di 800 milioni di persone nel mondo soffrono la fame[12], circa 2 miliardi di persone non hanno accesso a servizi di acqua potabile[12]; nel 2018 invece il 10% delle donne, dai 15 anni in su, ha affermato di aver subito una molestia fisica o sessuale da una persona vicina senza contare il fatto che in media meno del 40% delle donne che hanno subito violenze cercano aiuto[12]. La maggior parte dell'energia utilizzata nel mondo non proviene da fonti rinnovabili, nel corso del 2017 poco più del 17% del consumo energetico mondiale proviene da energia pulita[12]. Inoltre, il nostro pianeta negli ultimi anni ha subito un processo di inquinamento talmente intenso da quasi non sembrare vero, tra la California e le Hawaii si estende un'enorme isola di spazzatura di circa 1.6 milioni di km quadrati[5]. Questa lista potrebbe continuare all'infinito se dovessimo contare tutte le problematiche che abbiamo nel pianeta: surriscaldamento, CO2, caccia e pesca abusiva e abuso dei diritti umani sono solo alcuni aspetti che verrebbero eliminati nel caso si riesca a vincere il problema posto dagli SDG[1].

[1]Nel caso si volesse analizzare più a fondo i dati o i vari indicatori presenti è possibile visualizzarli tramite l'apposito portale messo a disposizione dalle Nazioni Unite[11].

Proposta di ricerca Nel corso dei decenni nonostante l'incredibile sviluppo tecnologico non vediamo diminuire la quantità di notizie che riportano quanto cittadini del mondo si trovano a fronte di situazioni per niente sostenibili, un esempio è il The New Humanitarian[3] che pubblica notizie di carattere umanitario ed ambientale. Alcuni degli articoli più recenti raccontano che: l'80% delle persone che ha subito uno sfollamento a causa del cambiamento climatico sono donne[21] con conseguenti problemi a livello di salute o gravidanze, in Africa 27 milioni di persone soffrono la fame a causa della siccità, dei conflitti e delle ricadute economiche della pandemia[16] e addirittura negli ultimi due anni in Madagascar le persone che soffrono la fame estrema hanno raggiunto la metà della popolazione, 1.5 milioni di individui, a causa della siccità[22]. Il contributo che questa tesi si propone di apportare è quello di utilizzare le informazioni contenute all'interno dei report, inerenti ai vari Goal dell'Agenda2030, per progettare e sviluppare un modello predittivo che possa lavorare affianco a un sistema di feedback per consigliare agli utenti comportamenti più sostenibili per il futuro. Oltre a questi report si vuole utilizzare un indice che esprima quanto sostenibile è lo sviluppo umano e in questo caso viene in nostro aiuto l'SDI[1]. Possiamo dire che gli SDG e l'SDI sembrano fatti l'uno per l'altro e da qui nasce la proposta di questa ricerca: trovare un nuovo modo di calcolare l'SDI tramite gli SDG allenando un modello che sia in grado di esprimere quanto "sostenibilie" sia la situazione corrente calcolando l'SDI tramite gli SDGs che gli vengono dati in input.

Intelligenza Artificiale e SDG Si propone quindi di creare un sistema di mash-up tra IA e SDG cioè un modello predittivo/regressivo capace di:

- Calcolare un indice che possa essere considerato una misura valida per esprimere la sostenibilità della situazione;
- Fare previsioni riguardo a certi obiettivi a richiesta dell'utilizzatore.

Per combinare questi due aspetti si è pensato ad una Deep Neural Network composta da strati densamente connessi. L'idea alla base è che ognuno di

questi indicatori abbia almeno una correlazione con qualcuno dei restanti e quindi una rete fully connected sarebbe in grado di trovarle, in quanto cerca le relazioni tra una qualsiasi caratteristica dell'input con tutte le altre, andando a manipolare i dati in ingresso e la loro dimensionalità. Il flusso pensato per la rete è quello di partire dagli **indicatori** (scelti dall'utente) e scendere a cascata passando per i rispettivi **goal** e le **macro categorie** per poi arrivare all'**indice** e ripercorrere il percorso a ritroso per ricostruire gli indicatori passati in input. Una spiegazione più dettagliata del sistema è presente nella sezione 4.1. Quanto descritto può essere interfacciato all'utente tramite una dashboard con la quale interagire col sistema: creare i template, allenare la rete e fare le richieste al modello. Ovviamente questo applicativo deve essere supportato da un backend il quale si occupa di: svolgere la parte di costruzione del modello (manipolazione dati, creazione e allenamento della rete) e di query: gestire le richieste dell'utente e spedire il messaggio al client con il risultato. L'utente destinatario di questo sistema viene immaginato come una figura che deve usare questi dati per prendere decisioni di tipo collettivo, come ad esempio un *police maker*, che si deve occupare di adottare una strategia in merito a questioni sociali o politiche all'interno di uno Stato.

Capitolo 2

Stato dell'Arte

Un'introduzione all'Agenda 2030 e alle reti neurali si sente obbligatoria per proseguire avendo gli elementi necessari a comprendere lo scopo del progetto. L'Agenda 2030 si posiziona centralmente in questo lavoro di tesi. Come spiegato nell'introduzione si tratta di una collezione di obiettivi, creata di comune accordo dalle Nazioni Unite, che devono essere raggiunti entro appunto il 2030. La parte importante di questa agenda sono proprio i dati che la compongono i quali permettono di avere un'idea di come, diversi Stati o raggruppamenti di aeree geografiche, affrontino determinate situazioni. Una rete neurale invece può essere definita come il mezzo tramite il quale l'Agenda2030 viene trasformata in un nuovo canale che ricopre un ruolo predittivo (il sistema di raccomandazioni proposto nell'introduzione) invece che una mera raccolta descrittiva (il ruolo di partenza dell'agenda).

2.1 Agenda2030

Le Nazioni Unite mettono a disposizione una serie di dataset che coprono varie tematiche ed esprimono valori sotto diverse unità di misura. Un classico esempio è un dataset in numeri percentuali e un altro che contiene gli stessi valori ma espressi per milioni di persone. Al giorno d'oggi purtroppo non sono presenti molti dati e specialmente prima del nuovo millenio la

quantità di dataset che non sono composti principalmente da valori nulli è veramente basso. Quello su cui possiamo contare sono i dati raccolti dal 2000 (circa) in poi in quanto altri Stati hanno iniziato a fornire le loro misurazioni diminuendo il numero di valori mancanti. Purtroppo non sono presenti dei dataset che descrivano la disponibilità di informazioni e quindi si deve rimandare alla dashboard dei global trends [11] delle Nazioni Unite per scegliere gli indicatori di interesse e controllarne la disponibilità.

2.1.1 SDG

Gli SDG, in inglese **Sustainable Development Goals** ed in italiano **Obiettivi di Sviluppo Sostenibile**, sono una collezione di 17 task che presentano collegamenti fra loro definiti dalle Nazioni Unite con l'intento di essere raggiunti entro il 2030, da qui il nome con il quale vengono indicati: Agenda2030. Questi task, divisi in 231 unici sotto obiettivi (248 totali), appartengono senza alcuna distinzione a tutti i paesi sviluppati e in via di sviluppo e poggiano le loro basi sui tre pilastri della crescita sostenibile: economico, sociale e ambientale. L'*Inter Agency Expert Group on SDGs (IAEG-SDGs)* è un gruppo formato dalla Commissione Statistica delle Nazioni Unite e composto da osservatori internazionali e regionali con il compito di implementare un framework per raccogliere gli indicatori globali. Questo è la principale fonte di dati per gli SDG, riportati qui di seguito e in grassetto quelli ritenuti più interessanti:

- **Obiettivo 1 (Società): Porre fine ad ogni forma di povertà nel mondo;**
- **Obiettivo 2 (Società): Porre fine alla fame, raggiungere la sicurezza alimentare, migliorare la nutrizione e promuovere un'agricoltura sostenibile;**
- **Obiettivo 3 (Società): Assicurare la salute e il benessere per tutti e per tutte le età;**

- **Obiettivo 4 (Società): Fornire un'educazione di qualità, equa ed inclusiva, e opportunità di apprendimento per tutti;**
- Obiettivo 5 (Società): Raggiungere l'uguaglianza di genere ed emancipare tutte le donne e le ragazze;
- **Obiettivo 6 (Società): Garantire a tutti la disponibilità e la gestione sostenibile dell'acqua e delle strutture igienico-sanitarie;**
- Obiettivo 7 (Ambiente): Assicurare a tutti l'accesso a sistemi di energia economici, affidabili, sostenibili e moderni;
- **Obiettivo 8 (Economia): Incentivare una crescita economica duratura, inclusiva e sostenibile, un'occupazione piena e produttiva ed un lavoro dignitoso per tutti;**
- **Obiettivo 9 (Ambiente ed Economia): Costruire un'infrastruttura resiliente e promuovere l'innovazione ed una industrializzazione equa, responsabile e sostenibile;**
- Obiettivo 10 (Società): Ridurre l'ineguaglianza all'interno di e fra le nazioni;
- Obiettivo 11 (Ambiente ed Economia): Rendere le città e gli insediamenti umani inclusivi, sicuri, duraturi e sostenibili;
- **Obiettivo 12 (Ambiente ed Economia): Garantire modelli sostenibili di produzione e di consumo;**
- Obiettivo 13 (Ambiente): Promuovere azioni, a tutti i livelli, per combattere il cambiamento climatico;
- **Obiettivo 14 (Ambiente): Conservare e utilizzare in modo durevole gli oceani, i mari e le risorse marine per uno sviluppo sostenibile;**

- **Obiettivo 15 (Ambiente): Proteggere, ripristinare e favorire un uso sostenibile dell'ecosistema terrestre;**
- Obiettivo 16 (Società): Promuovere società pacifiche e inclusive per uno sviluppo sostenibile;
- Obiettivo 17 (Società): Rafforzare i mezzi di attuazione e rinnovare il partenariato mondiale per lo sviluppo sostenibile;

Indicatori Il framework per gli indicatori è stato approvato nella 48esima sessione della Commissione statistica delle Nazioni Unite, marzo 2017. Come è possibile vedere dall'elenco puntato gli obiettivi presenti nell'Agenda2030 coprono argomentazioni troppo ampie per essere fini a se stessi. L'IAEG-SDGs ha stilato una lista di target e di indicatori per portare l'attenzione su più piccoli aspetti. Il Goal rappresenta un obiettivo dell'Agenda2030 ed ogni target può essere visto come una delle sue possibili descrizioni. Ad esempio per il Goal 1 ***Povertà Zero*** due target sono:

- *Entro il 2030, sradicare la povertà estrema per tutte le persone in qualsiasi luogo, attualmente misurata come persone che vivono con meno di $1,25 al giorno*
- *Entro il 2030, ridurre almeno della metà la percentuale di uomini, donne e bambini di tutte le età che vivono in povertà in tutte le sue dimensioni in accordo ai termini nazionali*

Ogni indicatore è invece una rappresentazione tabellare dei dati che possono descrivere questi target, ad esempio per il primo target un indicatore presente è la raccolta di quante persone vivono sotto la linea internazionale di povertà. Quindi la logica dietro al sistema dell'Agenda2030 è riassumibile nel seguente modo:

- Goal: Obiettivo generale da raggiungere entro il 2030;
- Target: Una delle possibili descrizioni che un Goal può avere;

- Indicatore: Una raccolta dati che fa capo al target.

Non esiste un dataset per un task ma bensì sono presenti varie raccolte di dati su un particolare punto ma che fanno riferimento ad un task più generico. È possibile prelevare tutti gli indicatori, per task, in unico file dove per ogni riga sono riportate informazioni sufficienti per discriminare la provenienza di quel dato, i.e. a quale indicatore fa fronte e per quale anno e Stato vale quella misurazione.

2.2 SDI

Con SDI indichiamo un indice di sviluppo sostenibile, compreso in un intervallo tra [0,1], che punta a misurare l'efficienza ecologica dello sviluppo umano in quanto quest'ultimo non può superare i confini del pianeta. Nasce come evoluzione dell'HDI[13], in inglese Human Development Index, a seguito dell'avvento dell'epoca geologica Antropocene[20]. Per calcolare l'SDI si parte dal valore HDI di ogni nazione (aspettativa di vita, istruzione e reddito) il quale viene diviso per il rispettivo superamento ecologico ovvero la misura in cui le emissioni di CO2 basate sul consumo e l'impronta materiale superano le quote dei limiti del pianeta. Nel 2019 sono stati ottenuti buoni risultati anche se nessuno ha superato lo 0.9. È presente anche una time table[1] che riporta i dati storici dal 1990 al 2019.

2.3 Reti neurali

Le reti neurali artificiali sono un modello computazionale che mira a replicare il funzionamento di una rete neurale biologica. Nel 1958 viene proposto da Frank Rosenblatt[18] il primo modello di rete neurale: il **Percettrone**[19]. Questo rappresenta una svolta in quanto i pesi sinaptici che lo compongono sono variabili permettendo al percettrone di apprendere. Nel 1986 viene proposto un metodo di addestramento delle reti MLP (Multi Layer Perceptron) tramite retropropagazione dell'errore[17] (error backpropagation) ovvero una

tecnica che permette di aggiustare i pesi della rete in base all'errore prodotto dalla comparazione dell'output con il risultato atteso. Al giorno d'oggi l'Intelligenza Artificiale ha fatto passi da gigante dando vita a diversi meccanismi di apprendimento: supervised, unsupervised e reinforcement learning; varie metodologie di architettura delle reti e anche diverse applicazioni come funzioni di regressione, classificazione o clustering dei dati. Ad esempio in questo elaborato viene proposta una rete *regressiva* e un *autoencoder* allenati rispettivamente tramite supervised e unsupervised learning. Questo ci porta a presentare le Deep Neural Network che pongono le base per la costruzione del modello proposto in questa tesi.

2.3.1 Deep Neural Network

Per Deep Neuarl Network intendiamo un modello di rete neurale artificiale che si compone di vari livelli dove ognuno calcola un valore che verrà usato dallo strato successivo. L'idea alla base è quella di sfruttare diversi livelli di rappresentazione non lineare dei dati per estrarre caratteristiche dal'input in modo da aumentare la precisione dell'informazione propagata. Inoltre grazie alla possibilità di manipolare la dimensionalità degli strati della rete è possibile cercare di imprimere un filo logico alla rete nel caso si voglia ottenere un'elaborazione del'input tramite trasformazione di esso in altri valori. Nel nostro caso dove si vuole ottenere un flusso del tipo:
$indicatori \rightarrow task \rightarrow macro \rightarrow indice \rightarrow macro \rightarrow task \rightarrow indicatori$.

2.3.2 Python e Pandas

Python e Pandas si presentano rispettivamente come ottimi strumenti per la manipolazione di reti neurali e di dati. Con python abbiamo accesso a librerie come *Pytorch*[14] che danno la possibilità di creare e manipolare reti neurali e *Flask*[7] per la creazione di un backend. Pandas d'altro canto fornisce librerie ausiliare per la manipolazione di collezioni di dati, come i

file csv (Comma Separated Value), semplificando le varie fasi di elaborazione dei dati come ad esempio rimozione/individuazione dei valori nulli.

2.3.3 Imputer

Con *Imputer* definiamo un meccanismo che si occupa di sostituire eventuali valori mancanti utilizzando una data tecnica. Un semplice esempio di Imputer è la sostituzione tramite media o valore mediano ma ci sono meccanismi iterativi che permettono di trovare i valori secondo l'andamento che questi hanno. Scikit-learn[15] fornisce un algoritmo simile chiamato *Iterative Imputer*[15] il quale imputa questi dati assenti modellando ogni caratteristica che presenta valori nulli in funzione di altre caratteristiche in modalità round robin.

Capitolo 3

Dataset

Come spiegato nel capitolo precedente i dataset sono rappresentati dai vari indicatori, target o goal. Per questo studio sono stati utilizzati dei dataset organizzati per *Goal* (SDG), ad esempio nel file *Goal1.csv* è possibile accedere a tutti gli indicatori del Goal 1.

Descrizione Va fatta una prima analisi sui dataset di partenza: questi sono agglomerati di vari indicatori che fanno capo ad un particolare task dal quale prendono il nome. Ogni riga del dataset riporta informazioni volte a discriminare quel particolare dato da tutto il resto usando le seguenti caratteristiche (colonne):

- Target;
- Indicatore;
- Codice di serie dell'indicatore;
- Descrizione dell'indicatore;
- Nome dell'area geografica;
- Anno;
- Valore.

Goal	Target	Indicator	SeriesCode	SeriesDescription	GeoAreaCode	GeoAreaName	TimePeriod	Value
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1981	42.7
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1982	42.3
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1983	41.4
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1984	39.8
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1985	38.2
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1986	36.8
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1987	35.8
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1988	33.8
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1989	36.9
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1990	36.2
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1991	36
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1992	35.1
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1993	34.3
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1994	33.2
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1995	31.3
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1996	29.7
1	1.1	1.1.1	SI_POV_DAY1	Proportion of population below international poverty line (%)	1	World	1997	29.6

Figura 3.1: Esempio del dataset Goal1.csv

Nella Figura 3.1 è presente uno screenshot di una porzione del dataset *Goal1.csv*.

L'unica caratteristica che è interessante per il modello è il *Valore* invece per costruire dei dataset che siano un semplice input per la rete abbiamo bisogno di effettuare un processo di pre elaborazione ed in questo caso verranno coinvolti: il *Codice di serie dell'indicatore*, l'*Anno* di rilevazione e i riferimenti all'*area geografica* per avere una chiave comune con l'SDI[1]. In breve quello che si vuole ottenere sono dei nuovi dataset, a partire dai *Goal*, dove ogni file contiene i dati inerenti ad un singolo indicatore: le colonne sono gli anni delle misurazioni e le righe contengono tutte le misurazioni disponibili per un certo Stato.

Feature Come detto nella sottosezione 2.1.1 gli indicatori univoci presenti sono 231, 248 totali, tra questi è possibile trovare tutti i dati che l'IAEG-SDGs ha raccolto riguardo i vari Goal. Alcuni task interessanti per creare una rete sarebbero relative ai Goal 1, 2, 3, 4, 6, 7, 8, 9, 12, 13 che coprono i tre cardini dello sviluppo sostenibile e specialmente riguardano gli aspetti più di tendenza negli ultimi anni. I temi più delicati nell'ultimo periodo

[1]Per poter utilizzare i nomi delle aree geografiche come chiave comune è stato effettuato un passaggio di ridenominazione di alcune aree geografiche all'interno del dataset dell'SDI.

sono proprio la *povertà*, insieme alla *fame* e alla *salute*, l'*accesso a servizi igienico-sanitari* e anche la *qualità dello studio*. Quest'ultima sarebbe utile in qualsiasi contesto come ad esempio le *energie rinnovabili, produzione e consumo responsabile, cambiamento climatico* ed infine la *crescita economica* che possiamo dire essere il punto centrale in quanto senza ricavi non è possibile investire in nessuno dei precedenti punti. I dati generalmente si presentano come numeri decimali in quanto esprimono per lo più *percentuale di persone*, *consumi e produzioni* o *accesso ai servizi*.

Problematiche Apparentemente i dataset nel formato della Figura 3.1 sembrerebbero "pieni", cioè non riportano alcun valore nullo. In realtà analizzando il file in modo più approfondito si noterà che per alcuni indicatori, o aree geografiche, sono presenti meno righe e questo di conseguenza non permette di avere delle misurazioni per un insieme comune di anni. Ad esempio, prendendo un indicatore qualsiasi, se per l'Italia abbiamo 10 righe con dati dal 2000 al 2009 e per la Francia abbiamo misurazioni dal 2000 al 2004 quando andremo a costruire il dataset per quell'indicatore avremo 5 valori nulli per la Francia, dal 2005 al 2009. Per far fronte a questa situazione si propongono due diversi approcci: il primo è una soluzione "naive" dove si sostituiscono i valori mancanti con il valore medio, la seconda soluzione implica l'utilizzo di un Imputer iterativo[15] che permette di modellare tutte le caratteristiche con valori nulli in funzione delle altre caratteristiche in modalità round robin. Alla fine verranno presentati i risultati della rete con tutti e due i metodi proposti.

3.1 Preelaborazione

I dataset subiscono quindi due fasi di preelaborazione:

Divisione dei *big goal* In questa parte i dataset vengono spezzettati in file più piccoli dove ognuno di essi rappresenta univocamente un indicatore;

Riempimento dei valori nulli Una volta ottenuti i dataset degli indicatori si applicano i processi di riempimento dei dati come spiegato nel paragrafo precedente con le due tecniche: *media e IterativeImputer*.

Quello che si vuole arrivare ad ottenere è una collezione di dataset dove in ogni file le **righe** rappresentano le varie aree geografiche e le **colonne** gli anni di rilevazione dei dati, un esempio è la Tabella 3.1.

Country/Year	2010	...	2019	2020
Sub-Saharan Africa	0.0	...	0.2	-4.5
Middle Africa	-2.6	...	-1.8	-5.2
Southern Africa	-0.1	...	-1.0	-8.2
...	...	...	...	...
Northern Africa	3.4	...	-0.9	-5.8
Western Asia	0.8	...	0.0	-4.9

Tabella 3.1: Formato del dataset per indicatore

Target Anche per quanto riguarda l'SDI ci troviamo di fronte ad una situazione simile a quella precedente in quanto mancano alcune misurazioni per degli anni/aree geografiche. Verranno applicati nuovamente gli stessi procedimenti utilizzati per i dataset degli SDG (media e Imputer). Il ruolo

Country/Indicator	2.1.1	3.1.1	4.1.1	5.6.1	...	15.1.1
Sub-Saharan Africa	0.0	197.3	3.0	4.2	...	264.2
Middle Africa	-2.6	46.5	5.0	3.8	...	57.1
Southern Africa	-0.1	4.7	1.0	9.5	...	6.8
...	...	...	...	...	...	...
Northern Africa	3.4	15.0	12.0	16.0	...	17.4
Western Asia	0.8	38.6	20.0	34.5	...	42.3

Tabella 3.2: Formato di un dataset bidimensionale per anno

coperto dall'SDI è quello di "ground truth" per una rete regressiva (o approssimativa) in quanto vogliamo che apprenda come calcolare questo indice e il target serve per effettuare un passo di backpropagation e modificare i pesi della rete in base all'errore che ha prodotto.

3.2 Input della rete

L'ultima parte di questo capitolo tratta come i dati preelaborati vengono manipolati nuovamente per costruire un plain input per la rete. Lo scopo di questo progetto è calcolare l'SDI partendo dai vari indicatori che l'utente decide di utilizzare. Dato che l'indice viene calcolato con cadenza annuale dovremmo fare in modo che anche i dati degli indicatori vengano forniti alla rete con lo stesso periodo. Si vuole mantenere come formato per le righe quello del dataset dell'SDI dove si posizionano le aree geografiche, sulle colonne invece si vogliono ordinare gli indicatori coinvolti. Nella Tabella 3.2 è presente un esempio di quello che sarà il dataset per un anno di misurazioni. Durante il processo di allenamento tutti i dataset prodotti verranno uniti in un'unica strttura che verrà mescolata e separata in train, validation e test set.

Capitolo 4

Metodo

L'architettura del sistema proposto viene sostenuta da tre punti fondamentali:

Dataset Collezione di dati pubblicata dalle Nazioni Unite;

Rete Neurale Un sistema che riceve in input i dati ed elabora le relazioni tra essi;

Indice di sostenibilità Un valore numerico, tra 0 e 1, che esprime l'indice di sviluppo sostenibile dell'umanità.

Il flusso inizia dai dataset che subiscono un accurato processo di pre elaborazione (vedi sezione 3.1) per essere usati come plain input per la rete che effettua due livelli di elaborazione dei dati:

Regressiva/Encoder Riceve gli indicatori in input e ne riduce la dimensionalità;

Decoder Punta a ricostruire l'input;

La loss, ovvero l'errore che la rete ha prodotto, viene calcolata sia sulla sezione regressiva che sul decoder per effettuare un passo di backpropagation.

4.1 Architettura

Il sistema proposto quindi si articola in due sotto reti: la prima rete si occupa di scoprire le correlazioni fra i vari indicatori in modo da trovare una formula per calcolare l'indice. La seconda parte è invece un Decoder, trattasi di una rete che mira a ricostruire l'input fornito. In pratica si vuole creare un modello che sia specializzato nel calcolo dell'indice, questa parte servirebbe solamente per un mero scopo descrittivo nel quale si vuole conoscere la qualità della situazione rappresentata dai dati inseriti, e un'altra rete che sia capace di predire input mancanti nel caso in cui l'utente volesse fare previsioni.

4.1.1 Rete regressiva

L'obiettivo posto è quello di scoprire eventuali correlazioni tra i vari indicatori/task, quindi una Deep Neural Network (DNN) composta da strati densamente connessi sembra fare al caso nostro: usando input bidimensionali dai quali scoprire relazioni tra qualsiasi caratteristica a qualunque altra caratteristica nei dati. Si utilizza questo procedimento perchè si vogliono trovare tre diversi tipi di correlazioni, seguendo appunto la struttura della rete nella Figura 4.1. Il primo layer mira a inferire le correlazioni che gli indicatori hanno sui task cercando di capire come ognuno di questi può essere formato a partire dagli indicatori. Nel secondo strato viene ridotta ulteriormente la dimensionalità dell'input per passare dai task alle macro categorie (Società, Economia e Ambiente). A questo punto la rete calcola il suo output al quale viene applicata la funzione di attivazione Sigmoide che normalizza i valori in un intervallo compreso tra [0,1]. Infine questo risultato verrà confrontato con la "ground truth" per eseguire un passo di backpropagation dell'errore.

Si propone quindi di costruire un'architettura composta come qui di seguito:

Primo strato Fully connected dove:

Input = #Indicatori

Output = #Task

Attivazione = Tanh

Secondo strato Fully connected dove:

Input = #Task

Output = #Macro categorie

Attivazione = Tanh

Terzo strato Fully connected dove:

Input = #Macro categorie

Output = Indice

Funzione di attivazione output Sigmoide per normalizzare l'output.

Optimizer SDG (Stochastic Descendent Gradient)

Loss function Mean Squeared Error

Relazioni

Come accennato nelle sezioni precedenti possiamo dire che indubbiamente ci sono relazioni fra i vari task. Se l'economia di un paese (task 8) cresce allora la percentuale di poveri (task 1) tende a diminuire, allo stesso tempo potrebbe capitare che l'ambiente soffra (task 13, 14, 15) a causa di questa crescita economica se non vengono applicate tecnologie sostenibili (task 9, 11).
Nella Figura 4.2 possiamo notare come al diminuire del *PIL* corrisponde un aumento della percentuale di *persone che soffrono la fame*. Questo andamento dovrebbe essere dovuto al fatto che con la diminuzione del PIL ci sarà

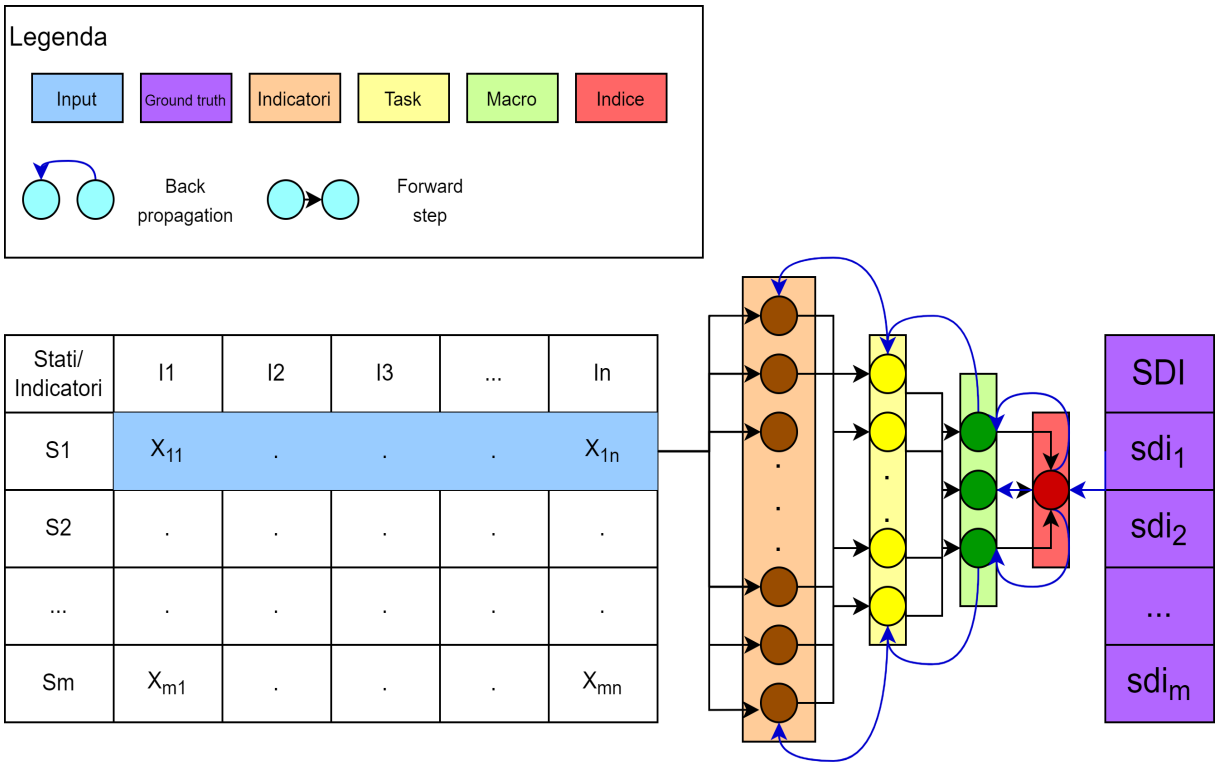

Figura 4.1: Struttura della rete predittiva

meno facilità di accesso al cibo, o comunque un aumento dei costi, e questo porta ad un incremento delle persone che soffrono la fame.

Comunque possiamo dire che ogni anno tutti gli Stati affrontano tematiche simili a quelle dell'esempio sopra e avere accesso ad un sistema di pianificazione delle risorse interrogabile in base ai propri bisogni potrebbe essere utile per diminuire gli sprechi di risorse e ridurre i tempi di compimento di questi obiettivi. Per poter dare vita ad un sistema come quello appena

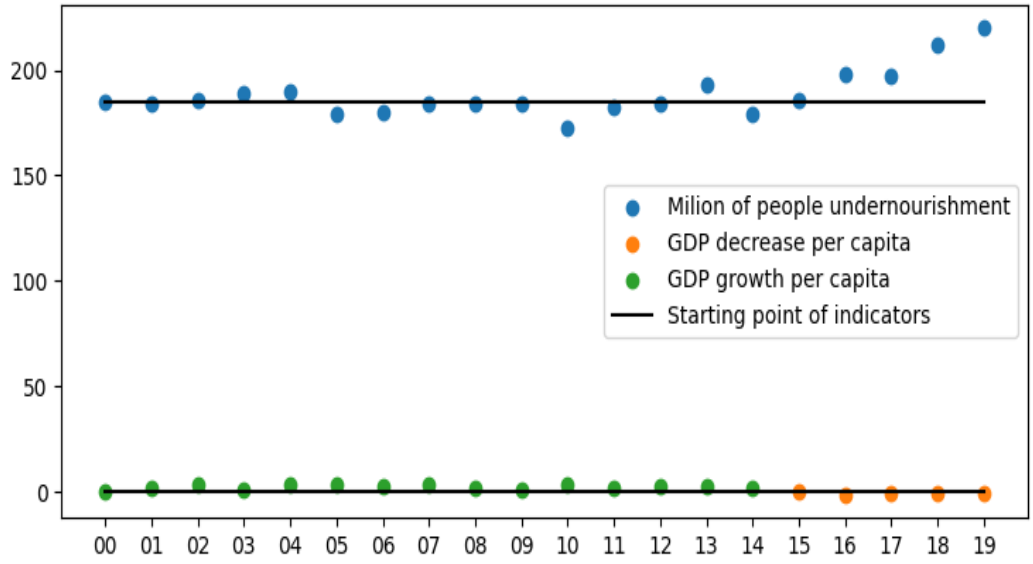

Figura 4.2: Grafico della crescita del PIL e % di persone che soffre la fame nell'Africa Sub Sahariana dal 2000 al 2019

descritto è necessario inferire quali relazioni vigono fra le variabili messe in gioco e qui acquisiscono un ruolo fondamentale gli strati densamente connessi. Quello che avviene all'interno della rete è semplicemente una moltiplicazione scalare tra una matrice e un vettore, la prima è composta dai parametri della rete, che vengono modificati (allenati) tramite backpropagation, il vettore invece è l'input ricevuto dallo strato precedente. Il risultato di questa operazione viene dato in pasto alla funzione di attivazione di quello strato e crea quello che sarà l'input per il layer successivo. Il passaggio tra il primo e il secondo strato è un'operazione mirata a scoprire che influenza ogni

indicatore, sotto parte di un task, ha su un qualsiasi altro task (di quelli presi in considerazione nel sistema). Il collegamento successivo invece vuole ridurre la dimensionalità del sistema per passare dai task dell'Agenda2030 ad un massimo di tre macro categorie: **Società**, **Ambiente** e **Economia**. Quanto detto è motivato dal fatto che l'SDI (Indice di sviluppo sostenibile[1]) è calcolato a sua volta da quattro aspetti principali che sono: **Aspettativa di vita e qualità dell'educazione**. In questo caso entrambi fanno parte di task presenti nella macro categoria società, **Reddito**, riferito quindi alla sezione economica, e infine un **indice di impatto ecologico** che fa capo alla categoria ambiente. Quando si raggiunge l'ultimo livello viene calcolato l'effettivo output della rete tramite la funzione Sigmoide che come detto in precedenza serve per normalizzare un valore nell'intervallo [0,1]. Una volta prodotto il risultato finale questo viene dato in input, insieme alla ground truth, alla funzione MSE (Mean Squared Error) che si occupa di calcolare l'errore quadratico medio usato poi per eseguire un passo di backpropagation con il quale verranno aggiustati i parametri della rete ed infine effettuato uno step per l'optimizer.

4.1.2 Auto Encoder

Nel caso dell'Auto Encoder quello che si vuole ottenere è una ricostruzione dell'input fornito, anche in questa situazione una DNN fa al caso nostro. L'obiettivo finale di questa rete è quello di dare in output una ricostruzione dell'input in modo da coprire eventuali dati mancanti. Si propone quindi di predire questi valori usando le correlazioni inferite fra le caratteristiche date in ingresso alla rete e sui vari strati densamente connessi che la compongono. Qui vediamo il punto di forza delle reti Fully Connected per questo particolare tipo di problema, come detto in precedenza questo tipo di reti mira a scoprire qualsiasi possibile rapporto presente fra le caratteristiche date in input e grazie a queste vogliamo sfruttare i valori disponibili per trovare quelli mancanti sfruttando le suddette relazioni.

Di seguito l'architettura proposta per l'Auto Encoder:

Primo strato Fully connected dove:

Input = #Indicatori

Output = #Task

Attivazione = Tanh

Secondo strato Fully connected dove:

Input = #Task

Output = #Macro categorie

Attivazione = Tanh

Terzo strato Fully connected dove:

Input = #Macro categorie

Output = #Task

Attivazione = Tanh

Quarto strato Fully connected dove:

Input = #Task

Output = #Indicatori

Funzione di attivazione output Sigmoide per normalizzare l'output.

Optimizer Adam with Look-ahead

Loss function Mean Squeared Error

Il flusso di questa rete è molto simile a quello esposto nella sottosezione 4.1.1, infatti, fino al terzo strato il modello rimane uguale per poi scindersi in una sotto rete specializzata nella ricostruzione dell'input manipolando i dati stessi, creando quindi un decoder. La dimensione degli strati della rete è stata pensata come un ibrido tra l'Agenda2030 e il SDI: prima si vuole creare un tramite tra gli indicatori (input) e i task dei quali fanno parte, come la classificazione dell'Agenda, dopo si vuole trovare i valori delle categorie dove questi task sono collocati (Economia, Società e Ambiente). L'ultimo strato

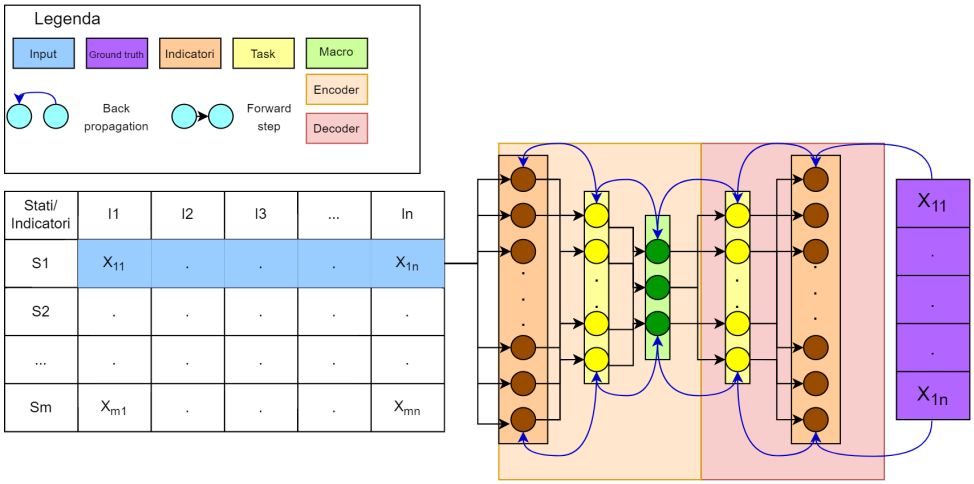

Figura 4.3: Struttura dell'Auto Encoder

dell'Encoder si chiude nei nodi delle macro categorie i quali vengono utilizzati come input per il decoder, da qui inizia il processo di decodificazione nel quale si cerca prima di ritrovare i valori dei task e poi degli indicatori raggiungendo l'ultimo dove viene utilizzata nuovamente la Sigmoide e ricostruito l'input tramite il processo inverso di normalizzazione. Anche in questo caso viene effettuato un passo di propagazione all'indietro dell'errore tramite la funzione dell'errore quadratico medio. La rete si divide quindi in due parti principali un **Encoder** e un **Decoder** che si occupano rispettivamente di codificare i dati in ingresso e di decodificarli per ricostruire l'input.

4.2 Input della rete (Allenamento)

L'input può essere rappresentato con un vettore dove ogni elemento contiene la misurazione per un preciso indicatore, di un certo Stato per un certo anno, come ad esempio nella Tabella 3.2. La rete riceve quindi in input gli indicatori e ne modifica la dimensionalità, tramite i vari layer densamente

connessi, cercando quindi di passare da indicatori a task a macro categorie fino all'indice. Immaginando questa situazione al contrario possiamo dire che lo stesso varrebbe muovendosi a ritroso, se siamo a conoscenza delle macro categorie possiamo intuire quali erano i valori dei task e così via fino ad arrivare nuovamente all'input inserito. Inoltre è possibile calcolare l'errore della rete riutilizzando semplicemente il batch che viene fornito: una riga di una matrice come nella Tabella 3.2 e l'indice che fa riferimento a quelle righe. Il vettore della matrice dato in pasto al modello viene usato sia per i calcoli interni sia per computare la perdita che il modello ha avuto per quel dato input (AutoEncoder), l'SDI passato viene usato per ricavare l'errore relativo al calcolo dell'indice da parte della rete (Regressiva). Questo approccio permette di effettuare una backpropagation su due livelli: il calcolo dell'indice e la ricostruzione dei dati ricevuti oltre ad un ulteriore step per l'optimizer, aumentando così la stabilità del modello.

4.3 Interfaccia web

Per rendere accessibile la fruizione del modello ad un'utenza non propriamente tecnica si è iniziata a sviluppare un'interfaccia web. In questa modalità si dovrebbe dare la possibilità all'utente di creare un profilo tramite il quale tenere traccia dei modelli creati ed utilizzarli per effettuare richieste al sistema. La home page è stata pensata con una GUI molto semplice, vedere Figura 4.4, con un logo e una barra in alto dove è possibile gestire l'account e entrare nelle due principali pagine del sito: la creazione di un template e l'accesso alla dashboard.

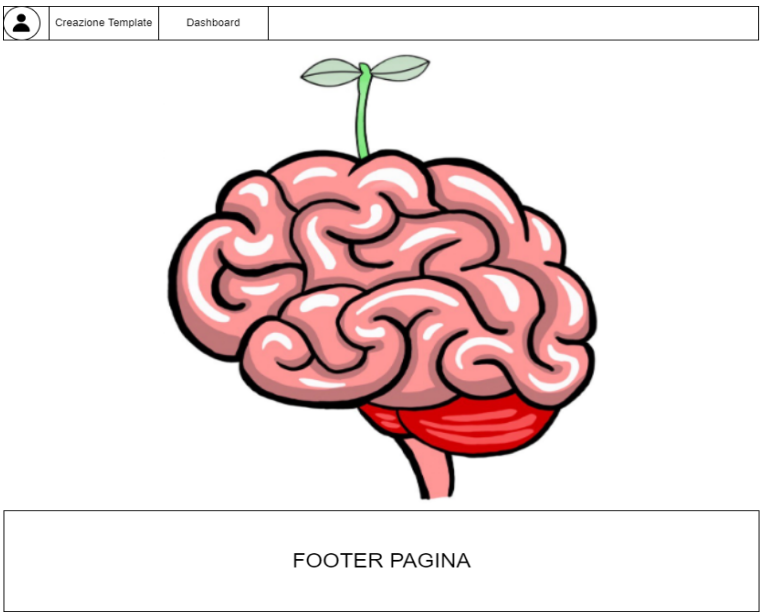

Figura 4.4: Home page del servizio web

4.3.1 Template

L'interfaccia di creazione di un template si compone di tre contenitori dove solo quello di sinistra viene reso modificabile per l'utente. Questo riporta due informazioni principali:

- Codice di serie dell'indicatore;

- Descrizione in linguaggio naturale dell'indicatore;

Selezionando dalla lista di indicatori si aggiungono o rimuovono quelli che l'user vorrebbe usare per creare il modello e questo aggiorna automaticamente i restanti due quadranti che rispettivamente rappresentano: i task (Agenda2030) selezionati tramite gli indicatori e l'anteprima degli strati della rete. Quando l'utente si ritiene soddisfatto dei dati inseriti può inviare la richiesta al backend tramite apposito tasto e avviare il processo di creazione della rete.

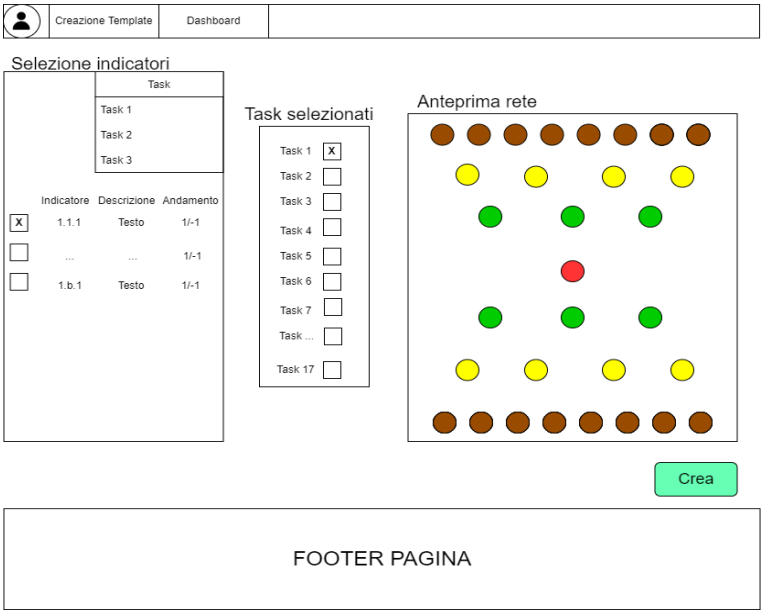

Figura 4.5: Sezione di creazione della rete

4.3.2 Query Interface

Si propone anche una versione di quella che dovrebbe essere una dashboard con la quale interagire con i modelli allenati. Nelle due successive figure possiamo vedere come questa si dovrebbe presentare: la prima è rivolta al calcolo dell'SDI e la seconda adibita alla previsione di obiettivi.

SDI Se si vuole semplicemente calcolare l'SDI si propone un'interfaccia come quella nella Figura 4.6 dove viene richiesto come input i valori per tutti gli indicatori presenti in quel template.

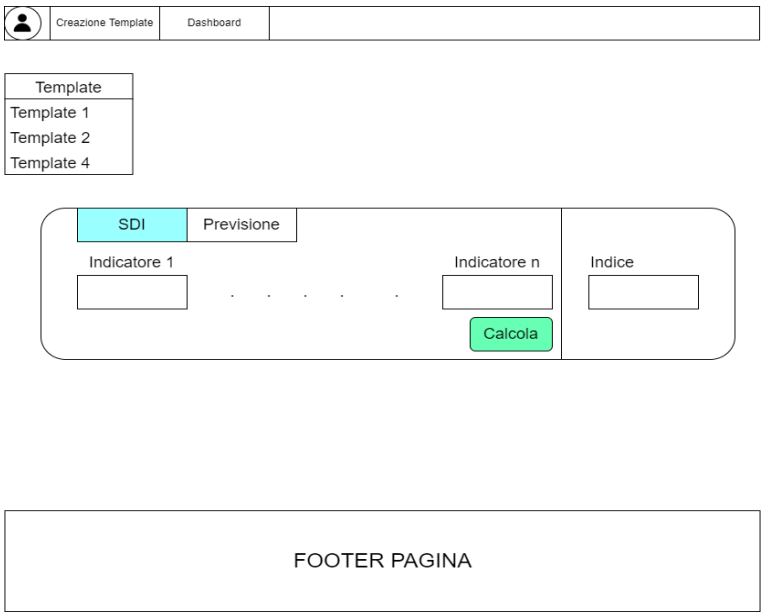

Figura 4.6: Dashboard durante la richiesta di calcolo dell'indice

Previsioni Per quanto riguarda le previsioni si vuole aggiungere una finestra dove viene mostrato l'output una volta calcolato. La forma dell'input rimane uguale a quella per l'SDI ma vengono omessi gli indicatori che si vogliono predire e viene richiesto l'indice che si vuole ottenere, di default 0.5.

Input della rete (Query)

Ricapitolando, una volta che il sistema è stato allenato è possibile interrogarlo in diversi modi in base al tipo di richiesta che vogliamo fare. Il primo modello di input riguarda la richiesta più semplice che si può effettuare al sistema, ovvero il calcolo dell'indice. In questo caso l'input della rete ha lo stesso formato di quello mostrato nella Tabella 3.2 ma viene inserita

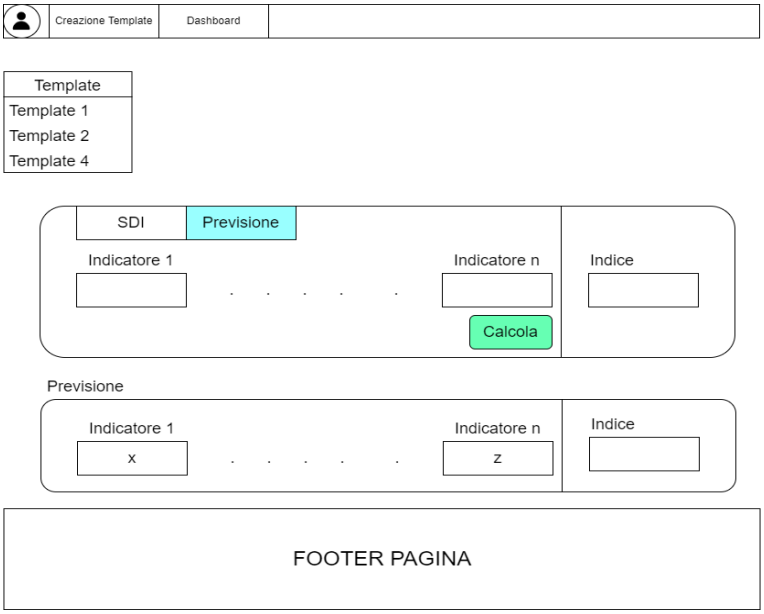

Figura 4.7: Dashboard durante la query di previsione di obiettivi

solamente una riga (l'input per il quale si vuole calcolare l'indice). Invece se si vuole fare una richiesta all'AutoEncoder va inserito un input simile a quello appena descritto ma dove viene omesso ciò che vogliamo predire insieme ad un indice, conosciuto o che si vorrebbe raggiungere/mantenere. Una descrizione più dettagliata è presente nel Capitolo 6.

4.4 Pipeline

L'esecuzione di questo modello, come la sua costruzione, segue certi precisi passaggi che possiamo dire ne compongono la pipeline. Per riassumere, i principali step sono:

- Scelta degli indicatori;
- Costruzione modello e dataset;
- Allenamento della rete;
- Rilascio del modello allenato.

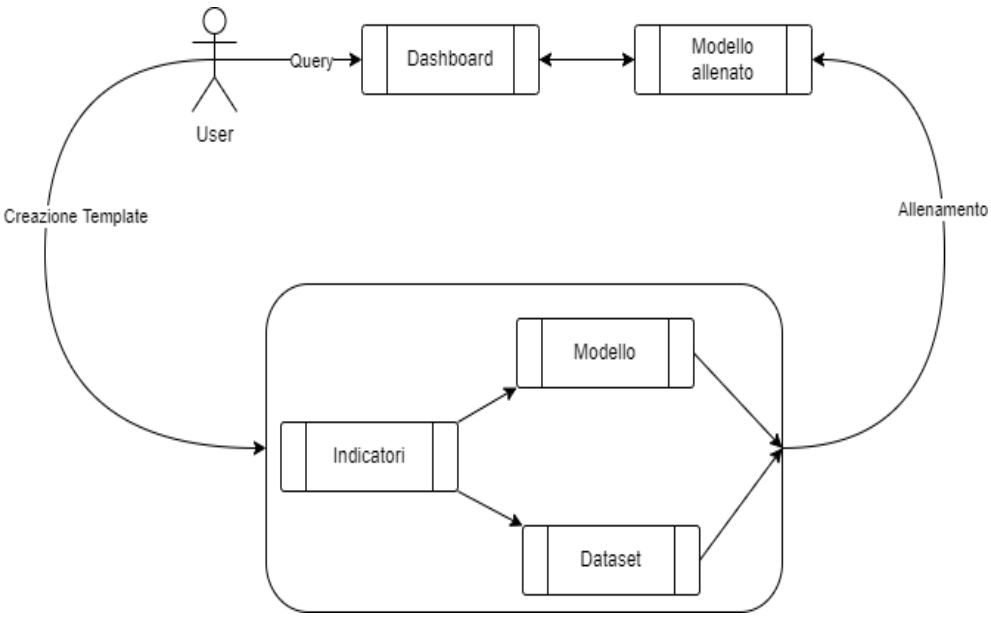

Figura 4.8: Schema della pipeline di SRAE (Sustainalable Regressive Auto Encoder)

La pipeline inizia nel momento in cui l'utente decide di creare un modello dall'interfaccia, nella Figura 4.5 infatti possiamo vedere che viene mostrato all'utente un menù dal quale è possibile selezionare gli indicatori, e di conseguenza i task, ai quali è interessato vedendo l'anteprima della rete nella stessa pagina. Quando l'utente ha finito di costruire la rete il backend ne riceve i dati e inizializza il modello. Il prossimo step è quello di creazione del dataset, il lavoro da svolgere consiste nel prelevare tutti i dati sugli indicatori scelti, dai corrispettivi file, e ordinarli come nella Tabella 3.2. A questo punto i dati sono ormai pronti per essere dati in pasto alla rete per il processo di allenamento. L'ultimo accorgimento da tenere è quello di normalizzare i valori, utilizzando il massimo e minimo presenti nei dataset, per semplificare e velocizzare il lavoro della rete. Una volta che il modello è stato allenato può essere utilizzato tramite la dashboard presentata nella Figura 4.6 e Figura 4.7.

Capitolo 5

Implementazione

In questa sezione verrà ripresa la pipeline, esposta alla fine del capitolo precedente, spiegandone in modo dettagliato ogni punto e la sua implementazione. Lo stato iniziale di questa routine parte dal *download dei dataset* fino ad arrivare al rilascio effettivo del modello allenato. Il linguaggio di programmazione principalmente coinvolto è Python per:

- Server;
- Oggetti dediti alla creazione, manipolazione, allenamento e interrogazione della rete;
- Creazione, manipolazione dei dati.

In breve: il backend si occupa di gestire le richieste dell'utente ed interfacciarlo con la rete neurale nonchè di gestire tutto quello che le riguarda, il frontend copre un ruolo interattivo dove è possibile visualizzare i risultati e interrogare o creare la rete. Per quanto riguarda la pre elaborazione dei dati invece questa si trova al di fuori del sistema finale in quanto riguarda un passaggio svolto una volta soltanto e che genera un insieme di dataset messi a disposizione dell'utente.

5.1 Dataset

Le Nazioni Unite hanno rilasciato vari formati di dataset. Attraverso il portale degli indicatori globali[11] è possibile accedere a una lista dei goal dell'Agenda2030 con i rispettivi target ed indicatori. Selezionando tutte le voci è possibile scaricare un file zip contenente 17 file ("Goal+Task+.csv") e ognuno di questi presenta le feature elencate nella sezione 3.

5.1.1 Da Goal a Indicatori

Una volta ottenuti questi dataset viene eseguito una e una volta soltanto l'algoritmo di pre elaborazione che viene parallelizzato su tutti i core della CPU per velocizzarne i calcoli. Questa routine si presenta come un semplice metodo di filtraggio dove si raccolgono gli indicatori comuni, si itera su tutti gli indicatori del dataset attuale e mano mano si prende soltanto la porzione del *dataframe* con quel codice di serie. Prima di procedere con l'iterazione si controlla se sono presenti almeno il 60% delle colonne sull'arco 1990-2019 questo perchè si vuole tenere soltanto quella parte di indicatori che hanno dati per almeno più della metà del range massimo proposto[1]. Quello che succede dopo è la semplice creazione di un vettore di dizionari dove ogni elemento dell'array corrisponde ad una diversa area geografica ed ogni dizionario contiene i valori di ogni anno, per quella determinata area, o "NaN" nel caso non sia presente all'interno del dataset. Una volta che finisce l'iterazione, sulla porzione del dataframe relativa ad un determinato indicatore, il vettore descritto precedentemente viene salvato in formato csv. L'array e il dizionario vengono reinizializzati nuovamente per la successiva iterazione fino al termine del dataset del Goal.

[1] Alcuni dataset hanno valori anche a partire dagli anni 70 ma sono veramente pochi quindi si è scelto come range massimo di dati dal 1990-2019 in quanto è lo stesso del dataset sull'SDI.

5.1.2 Riempimento valori nulli

In questa sottosezione verrà descritto come è stato implementato il processo di riempimento dei valori nulli dei dataset creati nella sottosezione precedente. Come accennato nella sezione 3 vengono proposte due differenti tecniche per colmare i valori nulli: IterativeImputer e Media. Quello che avviene è una semplice iterazione su tutti i dataset disponibili e per ognuno di questi viene applicata una delle due tecniche sopra, si vogliono creare appunto due versioni del dataset per comparare i risultati della rete per ognuno di essi. Inoltre prima di proseguire con una delle tecniche vengono effettuati ulteriori controlli:

- Rimozione di dataset con meno di sette righe;
- Rimozione di dataset con più del 90% di valori nulli complessivi;
- Rimozione di colonne o righe completamente vuote.

Se un dataset supera queste condizioni inizia il processo di riempimento di valori nulli come presentato nella seguente funzione: Listing 5.1. Per quanto concerne l'SDI il processo è identico al meno dei controlli che vengono fatti sugli indicatori.

5.1.3 Automazione

Quest'ultima sotto sezione relativa ai dataset si propone di esporre come vengono costruiti i data frame che verranno utilizzati per l'allenamento della rete in base agli indicatori scelti dall'utente. Anche in questo caso si tratta di una semplice iterazione ma in questo caso viene fatta sul range di anni a disposizione. Per ogni anno si itera su tutte le aree geografiche presenti negli indicatori selezionati e se almeno una di queste è presente nel dataset dell'SDI si prosegue, questo perchè non tutte le aree geografiche dell'Agenda2030 hanno un corrispettivo valore SDI. Se la precedente condizione è rispettata allora si itera su tutti i dataframe che l'utente ha selezionato e si preleva il corrispettivo dato per l'anno e lo Stato in questione, per ogni

iterazione viene salvato il corrispettivo file. Per la precisione il set di anni viene calcolato tramite intersezione tra tutti gli anni dei vari indicatori, per le aree geografiche invece si selezionano tutte quelle presenti. Si procede in questo modo in quanto i dataset con meno di 18 anni di misurazioni sono già stati rimossi per evitare un'intersezione troppo piccola. Per le aree geografiche invece non è stato applicato un controllo così ristretto per non dover rinunciare a troppi dataset e si propone la soluzione esposta prima inserendo come valore mancante la media per quell'anno.

```
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.impute import SimpleImputer
def FillNA(goal_df_list):
    for tmp_df in goal_df_list:
        nan_occ = tmp_df.isna().sum().sum()
        threshold = (tmp_df.shape[0] * tmp_df.shape[1])*0.9
        if len(tmp_df) > 7 and nan_occ < threshold:
            tmp_df.dropna(axis=0, how="all", inplace=True)
            tmp_df.dropna(axis=1, how="all", inplace=True)
            imp = IterativeImputer(
                max_iter=4, random_state=0, tol=0.1,
                min_value=tmp_df.min().min(),
                max_value=tmp_df.max().max()
            )
            # Caso Mean:
            #imp = SimpleImputer(
            #    missing_values=np.nan,
            #   strategy="mean"
            #)
            # Caso Median:
            #imp = SimpleImputer(
            #   missing_values=np.nan,
            #   strategy="median"
            #)
            imp.fit(tmp_df)
            filled_df = imp.transform(tmp_df)
            filled_df.to_csv(nomeFile)
```

Listing 5.1: Funzione di riempimento dei valori nulli

5.2 Rete neurale

Per lo sviluppo della rete neurale è stato utilizzato *Pytorch*[14] come framework. Il modello, come descritto nella sezione 4.1, si compone di due parti principali: Auto Encoder Regressivo e un Decoder. I vari strati vengono eseguiti in modo sequenziale passando sempre per una funzione di attivazione e producendo effettivamente due diversi output: un indice (AE Regressivo) e una lista di valori della stessa dimensionalità dell'input (Decoder). In questa sezione verrà presentato come la rete neurale viene costruita passo per passo e il processo di allenamento e test che ne consegue tramite i dati costruiti con le tecniche discusse nella sezione precedente.

5.2.1 Algoritmo creazione automatica rete

La rete si compone principalmente di tre gruppi di nodi:

Strato degli indicatori In questo livello ogni nodo rappresenta uno degli indicatori che l'utente ha scelto

Strato dei task Nel secondo (e penultimo) layer vogliamo rappresentare l'associazione nodo → task.

Strato delle macro In questo livello si cerca di portare l'attenzione verso le macro categoria delle quali i task, e i valori per calcolare l'SDI, fanno parte.

Quindi l'utente seleziona gli indicatori di interesse dall'interfaccia grafica e quando è soddisfatto della selezione manda la richiesta di creazione della rete. Quando viene ricevuta vengono inferiti in modo automatico quali task e macro categorie sono coinvolte con gli indicatori passati e viene creata la rete con questi tre valori come parametri. All'interno della rete vengono creati tutti gli strati necessari all'Encoder e al Decoder usando via via come dimensionalità di input e output i parametri del costruttore.

Optimizer & Loss function Per quanto riguarda l'Optimizer e la funzione per calcolare la loss abbiamo delle differenze per le due parti del modello. Gli optimizer utilizzati sono un semplice *SGD*, Stochastic Gradient Descendent, da stato dell'arte per la rete regressiva e un *ADAM con lookahead* di 5 step per tutta la rete (Auto Encoder). Gli autoencoders che utilizzano Adam con Lookahead come ottimizzatore hanno presentato risultati leggermente migliori rispetto a quelli con il semplice Adam, come riportato nel paper *Lookahead optimizer improves the performance of Convolutional Autoencoders for reconstruction of natural images*[9]. La scelta di due optimizer separati è stata guidata dall'idea di ottimizzare la rete in due punti diversi: con la tecnica della discesa del gradiente si vuole ottimizzare la rete regressiva per il calcolo dell'indice, tramite Lookahead ADAM invece si vuole ottimizzare il lavoro di Encoding e Decoding per la ricostruzione dell'input. Si propongono due diverse loss function organizzate come segue: una versione della rete utilizza come funzione di errore la *Mean Squared Error* l'altra utilizza la MSE per la rete regressiva e la *Binary Cross Entropy* per l'autoencoders. Per quanto riguarda l'autoencoders è stato preso spunto dal paper *On denoising autoencoders trained to minimise binary cross-entropy*[2] in cui viene citato che MSE è una funzione comunemente usata per l'allenamento di autoencoders e che nel caso dei pixel di un'immagine si può usare anche come funzione di loss la BCE visto che gli input sono compresi tra [0,1]. Anche nel nostro caso gli input sono compresi nel medesimo intervallo in quanto vengono normalizzati tramite massimi e minimi del dataset. Per quanto riguarda il calcolo dell'indice si utilizza semplicemente la MSE in quanto ha raccolto molto successo negli ultimi anni per i problemi di regressione[4].

Hyperparameters Per la scelta degli iper parametri, come *learning rate*, *batch size* e *epoche*, sono stati effettuati diversi test tramite un meccanismo di *EarlyStopping*. Per il learning rate sono state proposte due opzioni:

1e-3 in quanto valore di default degli ottimizzatori utilizzati;

3e-4 per avere a disposizione un valore più piccolo.

Come batch size sono stati utilizzati tre valori: 16, 32, 64. Come valore di default è stato fissato 32 per poi poter aggiustare il learning rate di conseguenza al cambiamento del batch size: la teoria suggerisce che quando si moltiplica il batch per un fattore k va anche moltiplicato il learning rate per $\sqrt{k}$.[8]. Invece riguardo al numero di epoche si è affidata la scelta al meccanismo di *EarlyStopping* incaricato di terminare l'allenamento quando la perdita calcolata dal modello, nel validation set, non diminuisce per cinque epoche consecutive. Ovviamente il numero di epoche che viene trovato da questi allenamenti non può essere utilizzato per tutti i tipi di rete che l'utente potrebbe costruire, su *gretel*[6] viene riportato che una buona regola è quella di moltiplicare il numero di colonne, che nel nostro caso sono gli indicatori, per tre, quindi si propone come moltiplicatore il numero di epoche migliore, risultato dagli allenamenti, diviso per gli indicatori utilizzati. Quanto esposto finora si concretizza tramite una rete che utilizza 7 indicatori appartenenti a 6 task e a tutte le macro categorie. Inoltre il dataset di allenamento viene diviso in 20% test set, 80% train set e validation set, rispettivamente 80 e 20%. Qui di seguito la lista degli indicatori coinvolti:

SI_POV_DAY1 Task 1(.1.1): Percentuale di popolazione al di sotto della soglia di povertà internazionale di 1,90 USD al giorno (%);

SI_COV_UEMP Task 1(.3.1): Percentuale di disoccupati che percepiscono l'indennità di disoccupazione, per sesso (%);

AG_PRD_AGVAS Task 2(.a.1): L'indice di orientamento agricolo per la spesa pubblica;

SH_DYN_MORT Task 3(.2.1): Tasso di mortalità sotto i 5 anni;

EG_EGY_CLEAN Task 7(.1.2): Percentuale di popolazione che fa affidamento principalmente su combustibili e tecnologie pulite;

SL_EMP_EARN Task 8(.5.1): Guadagno orario medio;

EN_EWT_GENPCAP Task 12(.4.2): Rifiuti elettronici prodotti pro capite (Kg).

5.2.2 Allenamento & Test

Il set di dati utilizzato per l'allenamento della rete cambia a seconda degli indicatori che l'utente seleziona, si presenta come un insieme di matrici bidimensionali dove le colonne rappresentano gli indicatori scelti e le righe gli stati per i quali sono disponibili le misurazioni, l'area geografica viene usata come chiave comune per selezionare l'SDI di quello Stato per quell'anno. Al momento della creazione della rete vengono prelevati tutti i dataset con gli indicatori di interesse e da questi si crea l'intersezione di tutti gli anni comuni. Il procedimento precedente serve ad evitare di avere dataset con valori nulli in quanto alcuni indicatori potrebbero non avere misurazione per gli stessi anni e ovviamente questo potrebbe portare a dataset nulli (nessun anno in comune) o comunque con poche colonne. Il processo di allenamento si divide in epoche: si tratta di un lasso di "tempo" in cui il modello vede tutto il dataset di allenamento. Il numero di epoche viene scelto in modo dinamico tramite il fattore trovato grazie al meccanismo di early stopping.

```
class SRAE(torch.nn.Module):
    def __init__(self,  indicator=1, task=1, macro=1, lr):
        super().__init__()
        self.sigmoid = torch.nn.Sigmoid()
        self.tanh = torch.nn.Tanh()
        self.activation = self.tanh
        self.fc0 = torch.nn.Linear( indicator, task)
        self.fc1 = torch.nn.Linear(task, macro)
        self.fc_regressive = torch.nn.Linear(macro, 1)
        self.fc0_decoder = torch.nn.Linear(macro, task)
        self.fc1_decoder = torch.nn.Linear(task, indicator)
        self.encoder = torch.nn.Sequential
            (self.fc0, self.activation,
             self.fc1, self.activation)
        self.regressive = torch.nn.Sequential
            (self.fc_regressive, self.sigmoid)
        self.decoder = torch.nn.Sequential
            (self.fc0_decoder, self.activation,
             self.fc1_decoder, self.sigmoid)
        default_batch = 32
        self.lr = math.sqrt((batch_size/default_batch)*lr)
        self.regressive_loss = torch.nn.MSELoss()
        self.autoencoder_loss = torch.nn.MSELoss()
        e_params = list(self.encoder.parameters())
        d_params = list(self.decoder.parameters())
        self.ae_params = e_params + d_params
        self.ae_optimizer = torch.optim.Adam(self.ae_params,
            lr=self.lr)
        self.ae_optimizer = Lookahead(optimizer=ae_optimizer,
            k=5,alpha=0.5)
        self.regressive_optimizer = torch.optim.SGD(
            self.regressive.parameters(),
            lr=self.lr, momentum=0.9)
        self.optimizer = MultipleOptimizer(
            self.regressive_optimizer, self.ae_optimizer)
}
```

Listing 5.2: Costruttore di SRAE

Capitolo 6

Casi d'uso

Per poter dare validità a questo studio sono stati pensati alcuni casi d'uso che potrebbero effettivamente apportare dei risvolti utili nella realtà, il primo riguarda le richieste che possono essere effettuate al modello ovvero quali informazioni utili questo può dare all'utente. Come secondo caso d'uso si è pensato di dare la possibilità di creare il proprio modello di rete tramite template semplici per l'utente dove può scegliere quali indicatori, tra quelli a disposizione, preferisce usare per la sua rete.

6.1 Query

Sono tre le query che possono essere effettuate al modello e sono:

- Calcolare un **Indice** che descriva quanto **sostenibili** sono i dati inseriti (riguardo ad uno Stato);
- Calcolare un indice prefissato (o pari a 0.5) con alcune feature sconosciute;
- Calcolare un indice prefissato (o pari a 0.5) con tutte le feature sconosciute;

La prima query, come detto all'inizio del capitolo, ha un mero valore descrittivo in quanto esprime la sostenibilità della situazione in funzione dei

dati inseriti. Le altre due query invece servono nel caso si voglia sapere come si deve modificare un certo indicatore in modo da raggiungere un determinato punto: ad esempio si vuole migliorare la situazione generica portandola ad un indice di 0.5 (tra 0 e 1) e abbiamo le rilevazioni riguardo a tutti gli indicatori per l'anno corrente, quello che ci interessa scoprire è come cambiare un obiettivo, diciamo il PIL. A questo punto si inseriscono tutti i dati incluso l'indice di 0.5, meno quello che si vuole predire, e la rete cercherà di ricostruire l'input con una modifica del valore del PIL in modo da calcolare l'output desiderato. Questo discorso è applicabile a più variabili contemporaneamente (ovviamente riducendone la precisione) o anche a tutte le variabili sconosciute usando valori randomici come input iniziale.

Template Un'altra proposta interessante sarebbe quella di poter creare dei template, tramite un'interfaccia grafica, dove scegliere quali indicatori utilizzare per costruire una rete. Inoltre seguendo questa strada sarebbe possibile costruire mini sistemi personalizzati, per indicatori di interesse, dove un soggetto come un police maker possa fare previsioni riguardo alla sua visuale. Ad esempio per lo sviluppo di uno Stato molto probabilmente i task importanti sono:

- Obiettivo 4: Fornire un'educazione di qualità;
- Obiettivo 8: Incentivare una crescita economica duratura;
- Obiettivo 9: Costruire un'infrastruttura resiliente e promuovere l'innovazione;
- Obiettivo 10: Ridurre l'ineguaglianza;
- Obiettivo 11: Rendere gli insediamenti umani sostenibili;
- Obiettivo 12: Garantire modelli sostenibili di produzione e di consumo;
- Obiettivo 13: Cambiamento climatico;

- Obiettivo 14: Conservare e utilizzare in modo durevole le risorse marine; (per aziende marittime)
- Obiettivo 15: Protezione e uso sostenibile dell'ecosistema terrestre; (per aziende terrene)

Il modello in questi casi verrebbe comunque allenato con i dataset costruiti in base agli indicatori selezionati e in seguito verrebbe interrogato utilizzando i dati forniti dall'utente. Poniamo caso ci interessi una versione ridotta del sistema dove consideriamo gli *Obiettivi 4, 8 e 10*, molto probabilmente la qualità dell'istruzione porta a dei ricavi maggiori, come anche la parità fra gli individui crea un ambiente più armonioso. A questo punto vorremmo sapere quanto cambierebbero gli introiti (Obiettivo 8) se si aumentasse il livello di istruzione dei cittadini (o ridurre il numero di persone prive di un'istruzione di base), in seguito la rete riceve in input i dati relativi agli obiettivi 4 e 10 e calcola il valore del task 8 in funzione degli altri due.

Capitolo 7

Analisi dei risultati

In questo capitolo vengono raccolti i risultati riguardo questo elaborato: prima di tutto verranno elencati le variabili testate e in seguito presentati i risultati riguardo agli allenamenti effettuati con lo scopo di trovare i migliori iper parametri per la rete oltre che il tipo di architettura neurale da utilizzare tra le due diverse proposte. In seguito verranno comparate due versioni del modello finale allenate rispettivamente con i dataset riempiti tramite Iterative Imputer e Media.

Sustenaibility Regressive Auto Encoder semplificato Trattasi di una versione della rete dove lo strato di encoder è comune sia per la parte regressiva che per l'autoencoder:
indicatori→*task*→*macro*→*indice*;

Sustenaibility Regressive Auto Encoder diviso In questa versione Encoder e Regressione condividono gli strati fino alle macro categorie dove si dividono rispettivamente in:
macro→*task* e *macro*→*indice*;

Batch size Indica quanti sample la rete utilizza per effettuare un passo di backpropagation. Tra le soluzioni proposte abbiamo: 16, 32, 64;

Learning rate Rappresenta la grandezza dello step mentre si minimizza la loss function. In questo progetto viene effettuato l'allenamento usando due learning rate: 1e-3, valore di default degli ottimizzatori, 3e-4;

Loss function Per loss function si intende quella funzione che calcola la distanza, o meglio l'errore, tra l'output della rete e l'output effettivo ("ground truth"). Nell'esperimento vengono usate la MSE, Mean Squared Error, e la BCE, Binary Cross Entropy;

Optimizer Trattasi di un algoritmo che si occupa di modificare i parametri della rete, pesi e bias, per ridurre la loss prodotta dal modello. Viene utilizzato SDG (Stochastic Gradient Descendent) per la rete regressiva e Lookahead Adam per l'AutoEncoder;

Data type Nel caso di questo progetto si utilizzano due versioni dei dati: una dove i valori nulli sono riempiti tramite **valore medio** ed una tramite **iterative imputer**[15].

Epoche Con un epoca si intende una completa esaminazione del dataset di allenamento da parte del modello. Si è posto come limite 100 epoche insieme ad un meccanismo di EarlyStopping.

7.1 Iper parametri

Per la scelta dei migliori iper parametri è stato eseguito un processo di allenamento per ogni possibile combinazione dei fattori elencati precedentemente. Tutte le possibili combinazioni dei parametri producono 48 diverse reti neurali. Il train delle reti avviene per un massimo di 100 epoche con un meccanismo di Early Stopping. In questa sezione verranno inclusi alcuni dei grafici più significativi ottenuti. Con "**tanh-srae**" si intende il modello dove encoder e regressione sono uguali fino allo strato con un nodo (indicatori→task→macro→indice→macro→task→indicatori) invece con **"srae"** indichiamo la rete che divide la regressione dall'encoder:

- indicatori→task→macro:
  - →indice;
  - →task→indicatori.

Una prima osservazione riguarda la funzione di loss BCE in quanto produce un errore decisamente più grande rispetto alla MSE, circa 0.3 contro 0.01. Nella Figura A.1 vengono presentati i risultati ottenuti dall'allenamento con Binary Cross Entropy come loss function per tutti i batch size e learning rate, per puro scopo informativo in quanto i risultati ottenuti non si ritengono interessanti. Al contrario con la funzione Mean Squared Error si sono ottenuti buoni risultati per tutti gli allenamenti, anche con valori leggermente inferiori allo 0.01 nel test set. Una prima visione è reperibile nella Figura A.2. Confrontando i dati degli allenamenti effettuati con MSE possiamo notare che nella maggior parte dei casi i risultati migliori sono stati ottenuti con un batch pari a 16 ed alcune volte con 32, escludendo così 64 come batch size. Infatti facendo sempre riferimento alla Figura A.2 possiamo notare che nella sezione relativa a 64 l'errore prodotto è leggermente più alto. Un ulteriore paragone usando learning rate come discrimante di questi due batch porta in vantaggio 0.001, vedere Figura 7.1, ottenendo dei risultati migliori anche se per una piccola differenza.

Gli iper parametri che hanno performato meglio fino a questo punto sono:

- Batch size: 16 e 32;
- Architettura rete: tanh-srae e srae;
- Dataset: sia Imputer che Media.

Prima di portare a paragone i modelli migliori dobbiamo fissare l'ultimo iper parametro, il batch size. Nella Figura 7.1 si nota che l'errore ottenuto dalle reti con batch size 16 è minore rispetto a 32. Nel caso di "srae" con batch 16 abbiamo una loss minore di 0.0085 sia usando i dataset Imputer che media. Se volgiamo lo sguardo ai risultati ottenuti con batch 32 invece solo in un

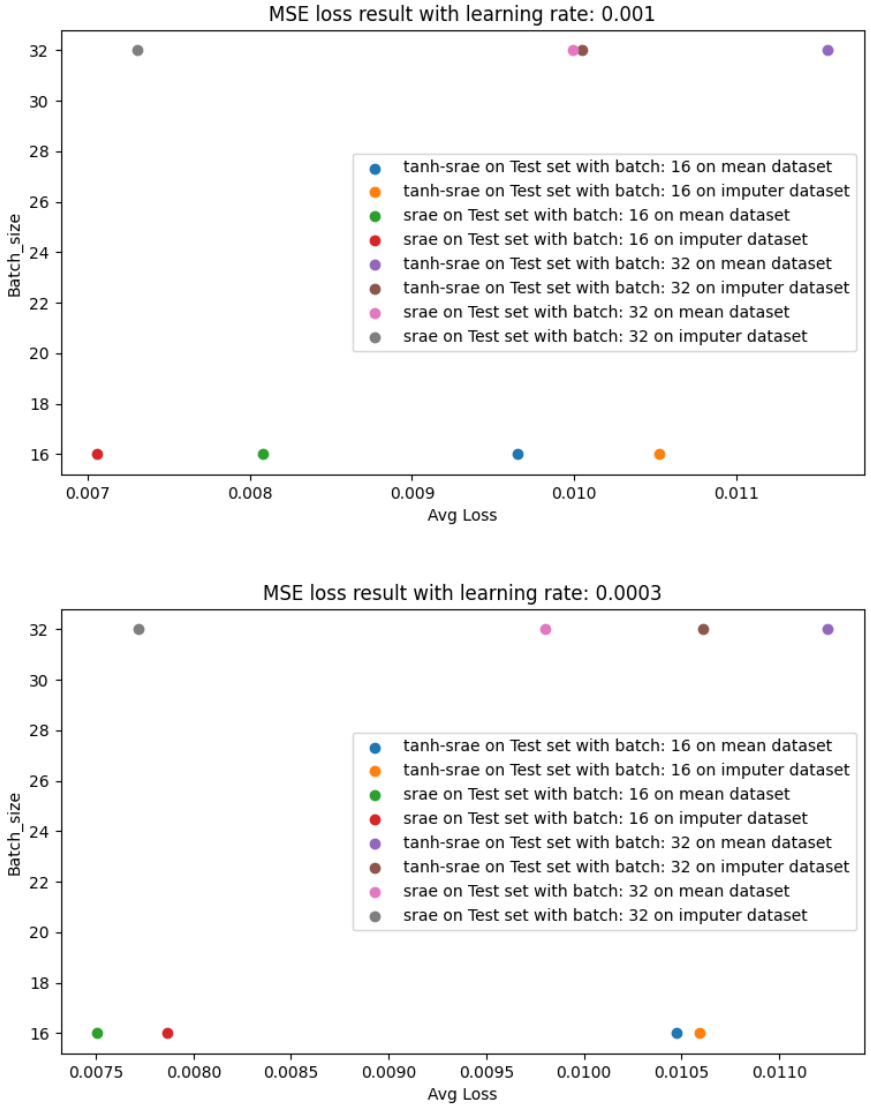

Figura 7.1: Media dell'errore sul test set per i modelli allenati con MSE, discriminati per learning rate

caso abbiamo una loss sotto lo 0.01 ("srae" imputer). Si propone quindi di utilizzare 16 come dimensione del batch in quanto 3 dei 4 esperimenti della Figura 7.1 hanno un errore inferiore all'1%.

7.1.1 Risultati iper parametri

L'ultimo paragone viene fatto fra le 4 combinazioni di parametri rimaste dopo aver fissato: batch size pari 16, learning rate a 0.001 e come loss function la MSE. I modelli sono:

- srae con Imputer;
- srae con Media;
- srae-tanh con Imputer;
- srae-tanh con Media;

Nella Figura A.3 possiamo notare immediatamente che "srae" è migliore rispetto a "srae-tanh" anche se si tratta di un distacco di solo 0.002 punti che comunque è un buon risultato considerando che il range dell'errore varia da 0 a 1. In tutti gli esperimenti "srae" ha sempre performato meglio rispetto all'avversario "srae-tanh". Inoltre, gli errori prodotti dagli allenamenti con Imputer sono leggermente più bassi probabilmente perchè l'Iterative Imputer fornisce una generalità maggiore ai dati riempiendo i valori nulli con numeri diversi al contrario della media che va ad inserire sempre lo stesso elemento. Oltretutto "srae" riesce ad ottenere più informazioni dall'input grazie alla divisione dei compiti all'interno della rete tramite appositi layer che ne permettono la specializzazione.

	srae-imputer	srae-mean	srae-tanh-imputer	srae-tanh-mean
Train	**0.011**	0.012	0.012	0.012
Validation	**0.010**	0.011	0.012	0.012
Test	**0.007**	0.008	0.010	0.009

Tabella 7.1: Tabella con la media dell'errore per Train, Validation e Test set su 4 versioni di SRAE.

7.2 Risultati finali SRAE: Sustainable Regressive Auto Encoder

Adesso verranno mostrati i grafici del modello che si propone come lavoro di ricerca. In queste immagini sono riportati i dati ottenuti dal train, validation & test set della rete **"srae"** con dataset **Imputer**, learning rate **0.001**[1] e batch size **16**. La rete riesce ad abbassare drasticamente la loss già durante le prime 10 epoche di allenamento. Partendo da un errore iniziale di circa 0.02 per poi dimezzare e scendere sotto la soglia dello 0.01 già alla tredicesima epoca. La rete comunque continua ad apprendere e riesce a ridurre ulteriormente l'errore che produce fino ad arrivare ad una loss media pari a 0.007 per il test set.

Test Sono stati effettuati alcuni test per controllare se effettivamente la rete ha imparato un andamento sensato tra gli indicatori e l'indice di sostenibilità. Per quando riguarda l'SDI sono state fatte due richieste cambiando il valore relativo all'indicatore "EG_EGY_CLEAN", percentuale di popolazione che fa affidamento principalmente su combustibili e tecnologie pulite, ed effettivamente all'aumentare di quel dato corrisponde anche un aumento dell'indice calcolato, vedere Figura 7.3, indicando un andamento sostenibile. È stato chiesto alla rete di predire un possibile valore di "EG_EGY_CLEAN",

[1] Si precisa inoltre che il lr: 0.001 è in realtà pari a 0.022 per il fatto che al moltiplicare di un fattore k il batch si deve moltiplicare il learning rate per $\sqrt{k}$ [8].

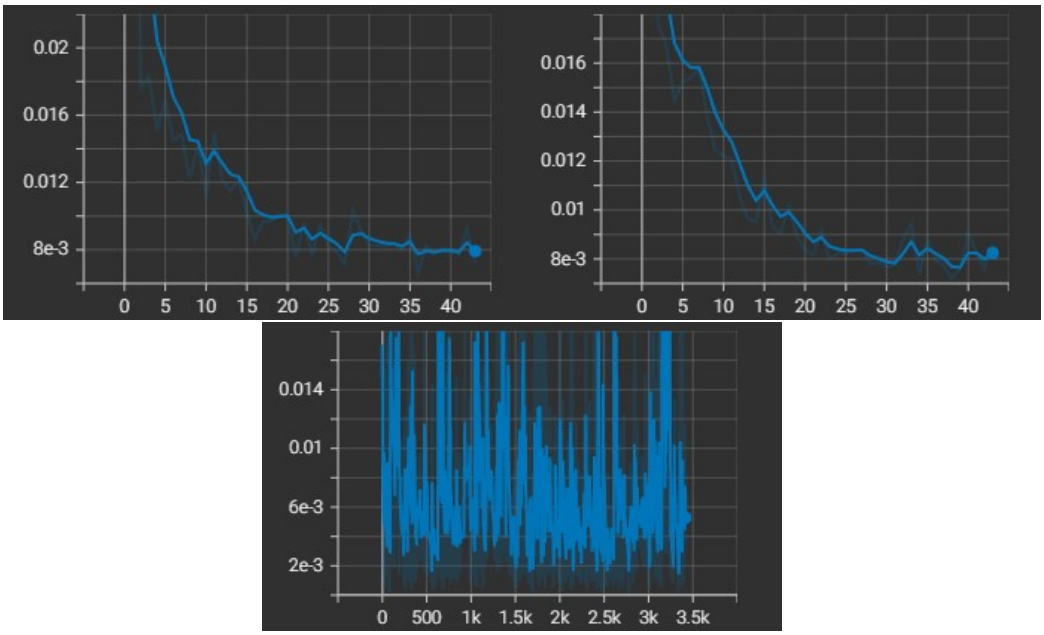

Figura 7.2: Plot dell'errore prodotto dalla versione finale di SRAE in train, validation e test set.

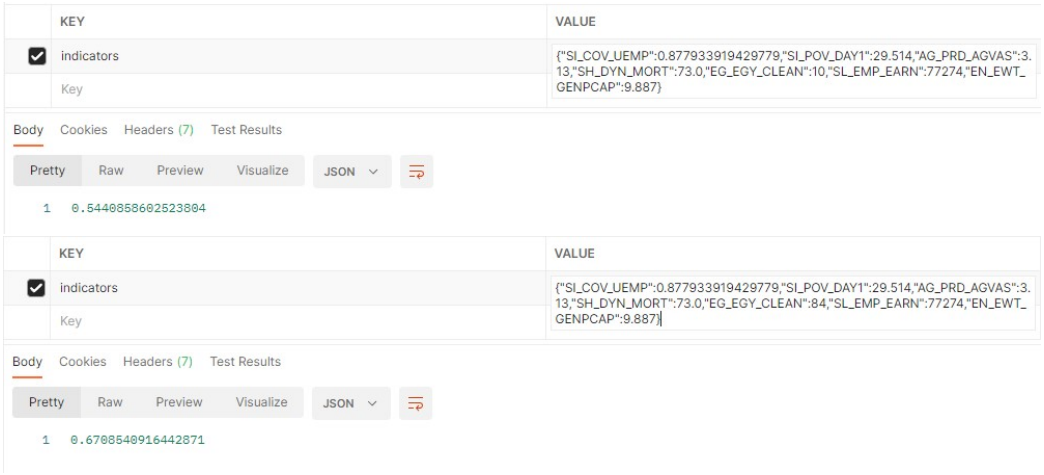

Figura 7.3: Confronto dell'indice calcolato con diversi valori di "EG_EGY_CLEAN"

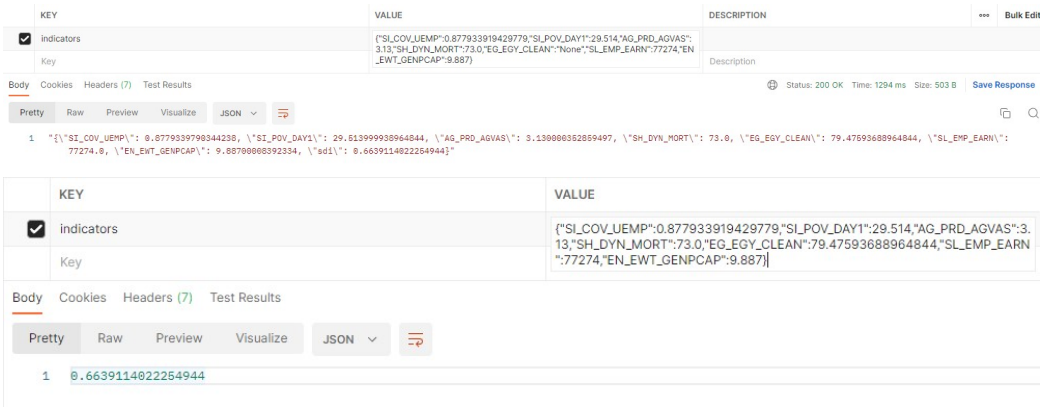

Figura 7.4: Nella prima foto: output della query di previsioni con i valori degli indicatori calcolati e l'indice corrispettivo. Nella seconda foto: sdi calcolato con l'output della previsione.

usando gli stessi dati della Figura 7.3, corrispondente ad un indice di 0.67. La rete ha trovato un output soddisfacente e questo se viene ridato in input alla rete calcola lo stesso SDI dimostrando che la rete rimane fedele alle computazioni passate. Nella Figura 7.4 è possibile vedere prima l'output della query di previsione e poi l'indice ottenuto da questi indicatori.

Capitolo 8

Conclusioni

In questo capitolo si presentano le ultime considerazioni riguardo questo elaborato e quelli che potrebbero essere alcuni sviluppi futuri.

Loss Function Nel capitolo precedente abbiamo visto che i modelli con **Binary Cross Entropy** come funzione di loss performano molto peggio rispetto a quelli che utilizzano la **Mean Squared Error** escludendo quindi la prima dell'architettura del modello, vedere Figura A.1 e Figura A.2.

Data type Inoltre, in tutti gli esperimenti i dataset costruiti tramite **Imputer** performano leggermente meglio rispetto a quelli con la **Media** rendendolo la soluzione più adatta per la mancanza di dati, riferimento a Figura A.2.

Batch size & Learning rate Per quanto riguarda il batch **64** è decisamente troppo alto e non permette al modello di apprendere bene. Invece **32** e **16** rendono decisamente meglio. **16** ha portato i migliori risultati in generale insieme ad un **learning rate** di $0.001 * \sqrt{0.5}$[8], vedere sezione 7.1.

Architettura Tra le due **architetture** presentate quella che performa meglio è senza dubbio **"srae"**. Questo dimostra come una separazione dei doveri all'interno della rete porti quest'ultima a ottenere risultati migliori. La divisione dei layer e dell'optimizer permette al modello di specializzare i suoi pesi

in base al punto di esecuzione. Possiamo dire che fino al livello delle macro categorie eseguiamo le stesse manipolazioni dell'input, sia per per la parte di **Encoding** che per la parte di **Regressione**. Una volta raggiunto quel layer si vuole portare la rete a generare due diversi output in base alla richiesta dell'utente: un semplice indice per indicare il livello di sostenibilità dello sviluppo e una ricostruzione dell'input per fare previsioni di dati. Questi due output dovrebbero essere generati da diversi calcoli perchè: rappresentano due risultati separati, hanno dimensionalità ben distinte e sono calcolati da due tipi di reti neurali diversi.

Epoche In media il numero di **epoche** impiegato dai migliori modelli è di 42 per 7 colonne di dati utilizzate (indicatori). Per scegliere il numero di epoche dell'allenamento si propone di moltiplicare il numero di feature selezionate dall'utente per 6, $\frac{42}{7}$. Si propone di lasciare comunque il meccanismo di Early Stopping, a discrezione dell'utente finale quando costruirà la rete personalizzata in modo da evitare overfitting del modello.

Considerazioni finali Il modello costruito è in grado di calcolare un indice di sostenibilità in base agli indicatori scelti, sa codificare e decodificare l'input anche se non completo di tutti i valori. Per dare in output una previsione la rete continua a manipolare l'input finchè questo non combacia con l'SDI che si voleva ottenere, da questo punto di vista ci sono ancora dei margini di miglioramento. La rete al momento è capace di costruire una serie di indicatori per raggiungere un determinato indice ma non è detto che questi siano quelli che l'utente si aspettava. Il modello modifica i pesi in base ai dati con i quali viene allenato e questi sappiamo benissimo non essere precisi in quanto contenenti valori mancanti in partenza. Inoltre, quando si mettono insieme vari indicatori si può creare un gap ancora maggiore perchè non è scontato avere dati per tutte le aree geografiche coinvolte. Comunque il modello sembra essere in linea con quanto si voleva ottenere avendo dimostrato che ha imparato che impatto determinati indicatori hanno sul calcolo dell'SDI. Questo si prospetta come uno strumento interessante in quanto si possono

selezionare le caratteristiche che ci interessano e sfruttarle per progettare il proprio sviluppo sostenibile. La rete estrapola l'influenza che ogni indicatore ha sull'indice in base a come questi cambiano tra gli anni ed in funzione delle altre feature selezionate. Quando la rete è allenata può essere usata per:

- Conoscere il grado di sostenibilità dei dati inseriti;
- Generare un input che combaci con un determinato indice di sostenibilità.

8.1 Sviluppi futuri

Certamente uno dei primi sviluppi futuri sarebbe quello di terminare l'implementazione dell'interfaccia grafica ed integrarla col sistema. Altre possibilità riguardanti i dati e la rete potrebbero essere: **ricercare dati del passato** per riempire i valori mancanti in modo da avere delle misurazioni reali invece che utilizzare un Imputer o **attendere** i prossimi anni per **nuove pubblicazioni di dati** sia da parte delle Nazioni Unite che nuovi indici SDI. Un altro possibile sviluppo teorico potrebbe coinvolgere **economisti, matematici e sociologi per effettuare uno studio più approfondito delle correlazioni** che i vari indicatori potrebbero avere e creare una funzione di forward della rete più specifica in modo da migliorarne stabilità e precisione. Sarebbe molto interessante anche studiare una diversa struttura della rete, sia come numero che per tipo di layer, che sia in grado di mettere in relazione feature vicine come l'influenza che Stati limitrofi hanno per alcune caratteristiche: uno Stato potrebbe fornire le risorse necessarie al Paese confinante ed in questa caso si influenzerebbero a vicenda, il venditore aumenterebbe il PIL e il compratore aumenterebbe la produzione.

Appendice A

Grafici dei risultati

Il codice è disponibile al seguente link: SRAE.

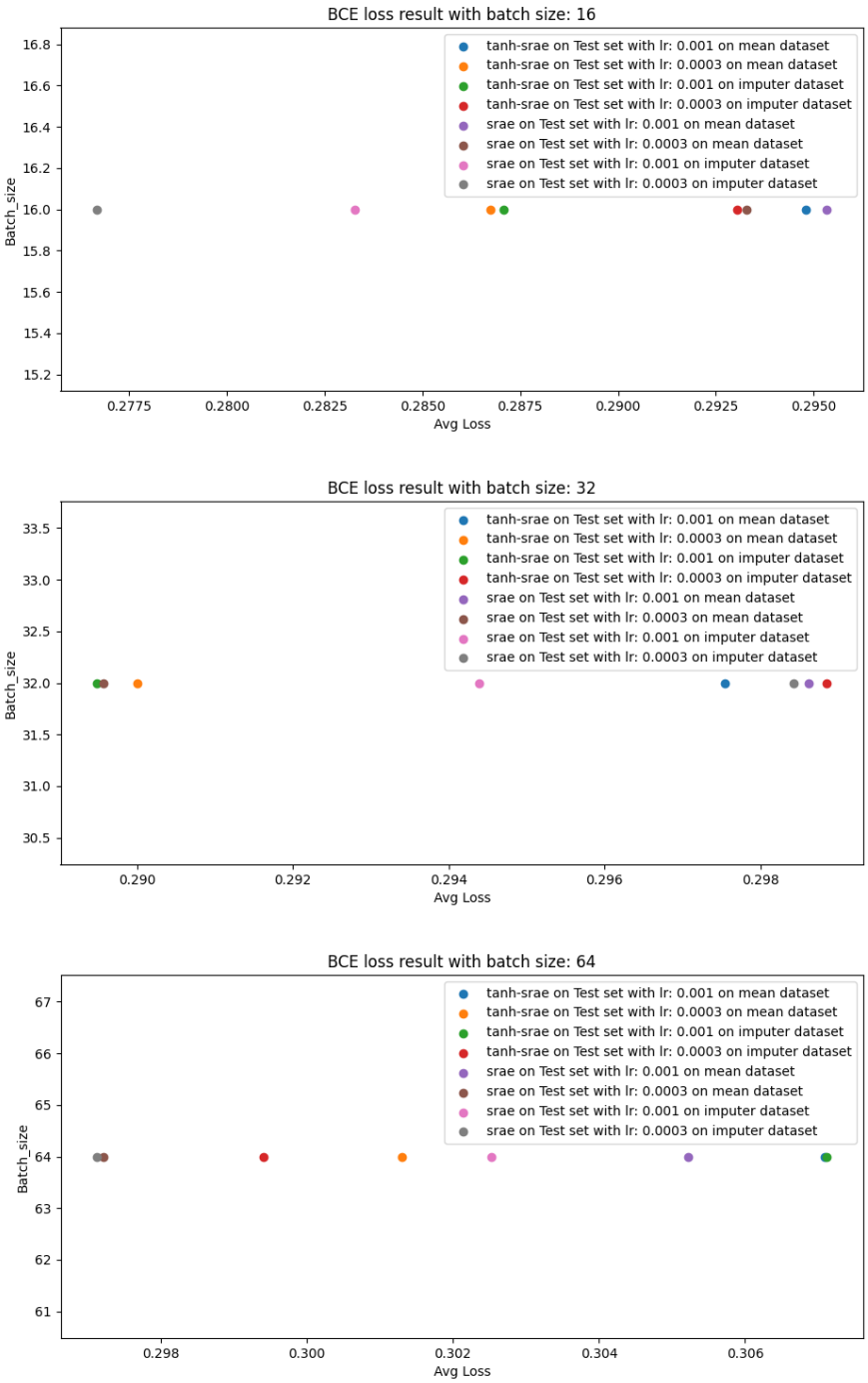

Figura A.1: Media dell'errore sul test set per i modelli allenati con BCE

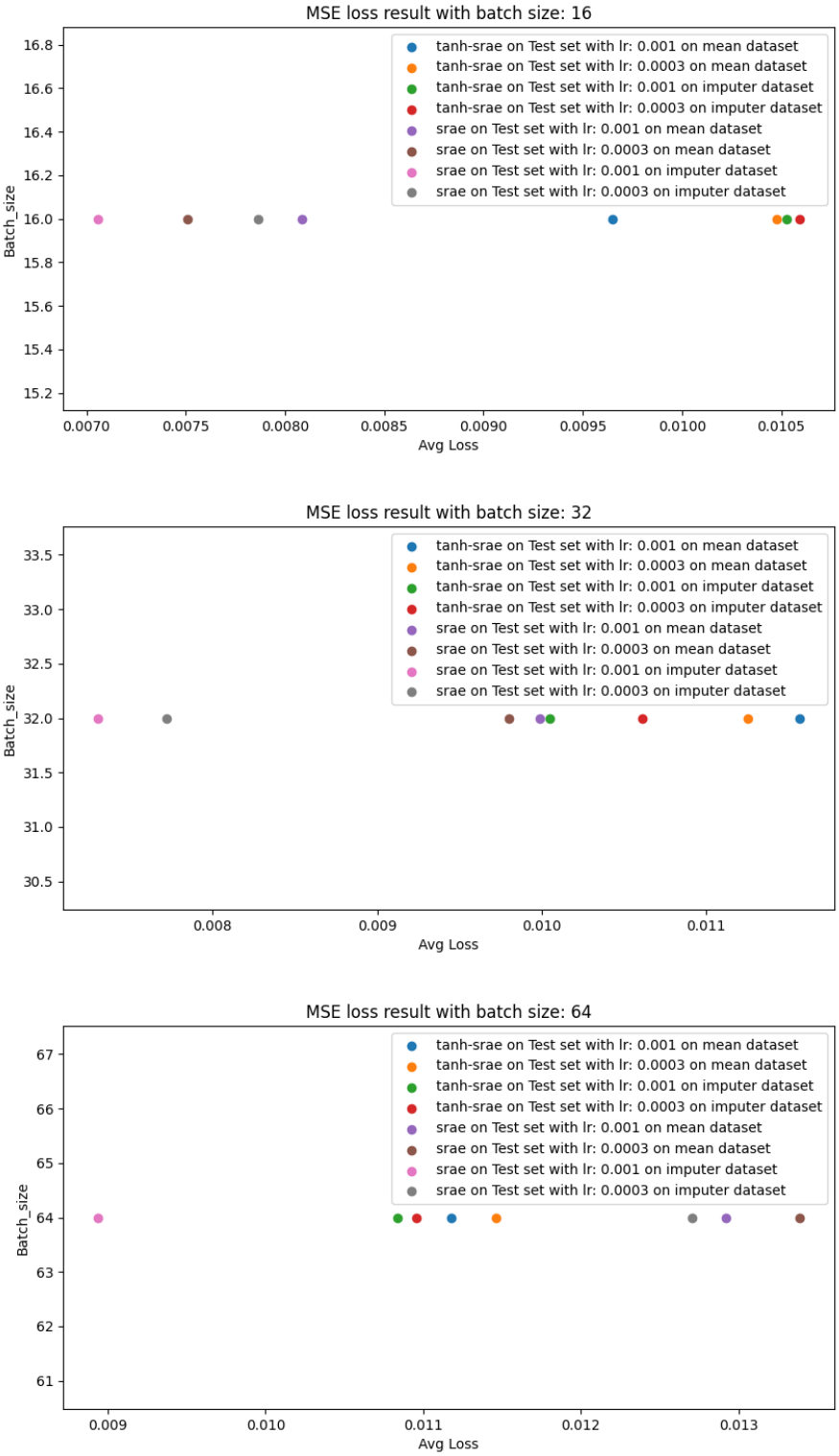

Figura A.2: Media dell'errore sul test set per i modelli allenati con MSE

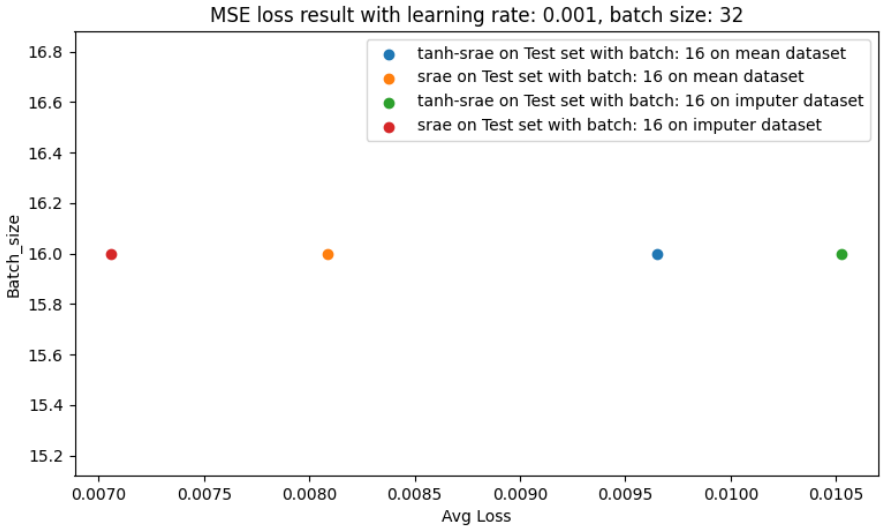

Figura A.3: Media dell'errore sul test set per i modelli allenati con MSE con batch: 16, lr: 0.001

Bibliografia

[1] Smith Jeroen & Co. Subnational hdi (v5.0), 2013.

[2] Antonia Creswell, Kai Arulkumaran, and Anil A. Bharath. On denoising autoencoders trained to minimise binary cross-entropy, 2017.

[3] Sara Cuevas, António Matheus, Paula Dupraz-Dobias, Obi Anyadike, Irwin Loy, Zuha Siddiqui, Janez Lenarčič, Jutta Urpilainen, Zainab Yunusa, Danielle Renwick, and et al. Environment and disasters, Jun 2022.

[4] AV Dorugade and DN Kashid. Alternative method for choosing ridge parameter for regression. *Applied Mathematical Sciences*, 4(9):447–456, 2010.

[5] National Geographic. Marine pollution, May 2022.

[6] gretel. How many epochs should i train my model with?, 2022.

[7] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[8] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014.

[9] Sayan Nag. Lookahead optimizer improves the performance of convolutional autoencoders for reconstruction of natural images, 2020.

[10] United Nation. Sustainable development goals: United nations development programme, Jan 2016.

[11] United Nations. Unsdg data portal, Mar 2017.

[12] United Nations. The sustainable development goals report 2020, 2020.

[13] United Nations. Human development index, Mar 2022.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Danielle Renwick. "no one has been spared": The climate crisis and the bleak future of food, Apr 2022.

[17] Wikipedia. Antropocene.

[18] Wikipedia. Frank rosenblatt.

[19] Wikipedia. Percettrone.

[20] Wikipedia. Antropocene, May 2022.

[21] Zainab Yunusa, Sara Cuevas, and Paula Dupraz-Dobias. Climate policies must allow women to control their bodies and their fates, Apr 2022.

[22] Jacob Zocherman. Drought and deforestation: How madagascar's children are paying the price, Mar 2022.