Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

I-eXplainer: applicazione web per spiegazioni interattive

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Luca Bracchi

Sessione III 2019-2020

Indice

Introduzione

Con l'aumento dell'impiego dei sistemi di intelligenza artificiale è aumentata anche la domanda di spiegazioni relative alle decisioni prese da tali sistemi. Queste decisioni sono importanti in quanto hanno il compito di garantire l'equità, identificare i problemi riguardanti l'allenamento del modello e assicurare che l'algoritmo funzioni correttamente, anche da un punto di vista legale.

L'eXplainable AI nasce con lo scopo di affrontare il problema delle macchine e degli algoritmi complessi che non forniscono una visione interna al loro processo di decision making. Ad un algoritmo spiegabile possono essere associate differenti tipi di spiegazioni, più o meno chiari dal punto di vista dell'utente che le richiede.

Lo scopo di questa tesi è dimostrare che è possibile trasformare una spiegazione statica relativa all'output di un algoritmo spiegabile in una spiegazione interattiva con una maggior grado di efficacia. La spiegazione cercata deve quindi essere human readable ed esplorabile.

Lo scopo della spiegazione che voglio creare è quello di essere più comprensibile per l'utente medio, in modo che questo possa capire quali decisioni hanno portato il sistema a generare un determinato output. Sarà appunto l'utente a guidare il processo di arricchimento della spiegazione, richiedendo direttamente al sistema le informazioni necessarie.

Per dimostrare la tesi userò la spiegazione di uno degli algoritmi spiegabili del toolkit AIX360 e andrò a generarne una statica che chiamerò spiegazione di base. Questa verrà poi espansa ed arricchita grazie a una struttura dati costruita per contenere informazioni formattate e connesse tra loro. Tali informazioni verranno richieste dall'utente, dandogli la possibilità di costruire una spiegazione interattiva che si sviluppa secondo il suo desiderio.

Il primo capitolo introdurrà il contesto scientifico e tecnologico: da un'infarinatura sull'IA si passerà alle spiegazioni relative ad algoritmi spiegabili, ovvero algoritmi i cui

risultati e decisioni possono essere compresi da utenti esperti, trattando in particolare temi quali la definizione di spiegazione, le regolamentazioni che le riguardano e il right to explanation. Quest'ultimo in particolare identifica il diritto di ricevere una spiegazione per un determinato output di un algoritmo che incida in modo significativo sulla persona.

Discuterò anche dei metodi per la valutazione di una spiegazione, basati sui principi fondamentali di interpretabilità e completezza, e della sfida che risiede nell'implementazione di un trade off tra i due.

Parlerò della FICO Explainable Machine Learning Challenge, soffermandomi sul dataset fornito e sul sistema vincitore della competizione, in quanto la soluzione da me implementata fa uso del dataset FICO HELOC e prende spunto dalla spiegazione fornita dall'app vincitrice, migliorandola attraverso l'interattività e un grado di interpretabilità maggiore.

La spiegazione fornita da questo sistema non è human readable ed è difficilmente interpretabile da un utente inesperto, in quanto è composta da grafici e tabelle che usano label e valori codificati in maniera astrusa senza spiegare significato e provenienza dei termini utilizzati. Non offre inoltre la possibilità di richiedere al sistema delucidazioni su questi termini, rendendo impossibile l'esplorazione della spiegazione.

Descriverò AIX360, il toolkit che supporta la comprensione di modelli di machine learning attraverso algoritmi spiegabili, descrivendo in seguito la web demo di IBM e il caso utente che presenta, ovvero quello dell'utente della banca che usa l'applicazione fornita da quest'ultima per capire perchè la sua richiesta di prestito è stata rifiutata dal sistema.

La spiegazione fornita da questo sistema è statica e risulta difficilmente comprensibile a un utente inesperto. Non è possibile argomentare la spiegazione, se non andando a cercare per conto proprio i termini non chiari nella documentazione, metodo noioso che richiede più tempo rispetto al permettere all'utente che legge la spiegazione di chiedere informazioni su quest'ultima direttamente al sistema. Da qui l'idea di implementare un

sistema che permette di argomentare la descrizione senza dover uscire dall'applicazione, ma interagendo con l'applicazione per esplorare la spiegazione attuale.

Nel capitolo 2 introdurrò I-eXplainer, la web app che ho costruito per dimostrare la mia tesi. I-eXplainer è un explainer system che presenta una breve introduzione del caso d'uso e dell'algoritmo utilizzato e mostra la spiegazione statica generata dall'algoritmo CEM del toolkit AIX360, dando poi all'utente la possibilità di interagire con quest'ultima per espanderla fino a raggiungere il livello di soddisfazione desiderato.

La spiegazione iniziale è una tabella di numeri e label codificate in maniera astrusa: il sistema la convertirà in una serie di stringhe human readable e le mostrerà all'utente, che comprenderà difficilmente, ma avrà la possibilità di interagire col sistema per raggiungere più facilmente il suo obiettivo: capire come migliorare il suo profilo in maniera da vedere accettata la sua prossima richiesta di prestito.

L'interazione utente-spiegazione è resa possibile grazie ad un'interfaccia grafica user-friendly: l'utente potrà domandare al sistema informazioni in merito a un determinato termine spiegabile cliccando direttamente su quest'ultimo.

Il capitolo 3 tratterà di dettagli implementativi di I-Explainer: struttura concettuale e concreta del sistema, strumenti utilizzati e implementazione di client, server e struttura dati.

Citerò anche le principali tecnologie utilizzate, quali AngularJS, Bootstrap, JQuery e JQuery UI, e lo scopo della loro introduzione nel sistema. Descriverò le caratteristiche fondamentali dell'interfaccia utente, mostrando anche figure di quest'ultima.

Nel capitolo 4 discuterò il protocollo di valutazione adottato per stimare l'usabilità di I-eXplainer rispetto a quella della demo di IBM. A causa delle difficoltà riscontrate nel progettare un protocollo di valutazione di efficacia e efficienza dei due sistemi(misure oggettive), ho optato per valutare invece la soddisfazione dell'utente(misura soggettiva).

Gli utenti compileranno un System Usability Scale form dopo l'utilizzo di ognuno dei due sistemi: il form ha lo scopo di verificare la soddisfazione dell'utente dopo l'utilizzo

di quella particolare applicazione e si articola in 10 frasi alle quali il cliente risponderà con un numero da 1 a 5 per definire il suo accordo o disaccordo rispetto a quest'ultima.

Per svolgere la valutazione ho diviso i tester in due gruppi: al primo ho fatto provare prima un sistema poi l'altro e con il secondo ho invertito l'ordine delle applicazioni testate. Al termine di ogni test ho fatto compilare all'utente il questionario SUS.

I dati dei form verranno elaborati da una API implementata nel server per decretare se I-eXplainer costituisce un explainer system migliore del suo avversario in termini di usabilità.

Nel capitolo finale definirò gli sviluppi futuri e introdurrò a grandi linee i problemi che potrebbe incontrare chi volesse implementare determinate funzioni nell'applicazione da me sviluppata: problemi come la risoluzione di conflitti dovuti a parole con stessa sintassi e diversa semantica.

Capitolo 1

1 Spiegabilità nell'IA

In questo capitolo andrò a introdurre i concetti fondamentali che saranno utilizzati nel corso della dissertazione per dimostrare la mia tesi. Definirò anche gli approcci al problema precedenti a questo lavoro focalizzandomi sui limiti e i motivi fondamentali della loro inadeguatezza. Dopodichè introdurrò le idee che mi hanno portato a creare una soluzione migliore rispetto alle precedenti.

1.1 Intelligenza artificiale e regolamentazione

Con intelligenza artificiale(AI) si intende un insieme di tecnologie che cercano di simulare l'intelligenza umana per risolvere compiti complessi. Uno di questi è il decision-making, processo cognitivo che ha come output la scelta di un'azione, parere o strada tra le diverse opzioni disponibili. L'IA è una disciplina che manifesta aspetti etici oltre che pratici e teorici. Nel giugno 2018 la Commissione Europea istituisce il gruppo indipendente di esperti ad alto livello sull'intelligenza artificiale con lo scopo di dipingere un quadro di riferimento sugli orientamenti etici[BY14] per un'IA affidabile. Secondo questo documento i principi etici per un'IA affidabile[Hle19] sono:

- il rispetto dell'autonomia umana: il sistema non deve subordinare, costringere, manipolare, condizionare o aggregare in modo ingiustificato gli esseri umani, ma deve essere progettato per aumentare e potenziare le abilità cognitive, sociali e culturali umane.
- la prevenzione dei danni[AOS+16]: il sistema non deve causare danni, aggravarli o avere un'influenza negativa sugli esseri umani. Sistema e ambiente in cui

opera devono essere sicuri[VA17], protetti, robusti e non esposti ad usi malevoli. Bisogna prestare particolare attenzione alle situazioni in cui il sistema di IA può causare o aggravare gli effetti negativi dovuti ad asimmetrie di potere o di informazione[Ott13], come nel rapporto tra datore di lavoro e dipendente, tra imprese e consumatori o tra governo e cittadini.

- l'equità e la non discriminazione[RPT10]: per quanto riguarda la dimensione sostanziale e procedurale. La dimensione sostanziale richiede impegno per garantire una distribuzione giusta di costi e benefici ed evitare discriminazione o emarginazione degli individui e dei gruppi[SHG+15]. La dimensione procedurale implica la capacità di impugnare le decisioni prese dal sistema e da chi lo gestisce e la possibilità di presentare un ricorso efficace contro di esse: l'organismo responsabile della decisione deve quindi essere identificabile.

- l'esplicabilità[GF17]: I processi devono essere trasparenti, lo scopo e le capacità del sistema apertamente comunicati e le decisioni spiegabili a coloro che ne sono direttamente o indirettamente interessati. L'assenza di tali informazioni può causare l'impossibilità di impugnazione di una decisione del sistema.

Tali principi dovrebbero coesistere in modo armonico, tuttavia è possibile che si creino tensioni tra loro: si pensi a un sistema di IA nell'ambito della polizia preventiva che può contribuire a ridurre la criminalità, ma con modalità di sorveglianza che interferiscono con la libertà individuale e la riservatezza, creando un conflitto tra il principio di prevenzione dei danni e quello di rispetto dell'autonomia umana. Non esistono soluzioni prestabilite per risolvere questo genere di conflitti.

Questo testo affronterà in particolare il principio di spiegabilità, principio non realizzabile negli algoritmi di machine learning detti a "scatola nera", dove neanche il programmatore può spiegare come l'IA sia arrivato a prendere certe decisioni piuttosto che altre: è il caso delle deep neural network[CMMS12] e dell'ensemble learning[D+02].

1.2 Spiegabilità e spiegazioni

Il concetto di spiegabile fa riferimento a sistemi i cui risultati e decisioni possono essere compresi da umani esperti. La trasparenza delle tecnologie di IA raramente è perseguibile senza danneggiare l'accuratezza della previsione: è quindi necessario un compromesso tra questi due aspetti[TWB17]. Ciò non toglie che la ricerca sull'Explainable AI ci possa portare ad avere modelli spiegabili pur mantenendo un alto livello di performance. La figura 1 mostra la relazione tra la capacità di apprendimento del modello e la sua spiegabilità: ad oggi le tecniche meno trasparenti, ma più performanti in termini di apprendimento sono quelle basate sul deep learning, mentre quelle più spiegabili sono quelle basate sugli alberi decisionali.

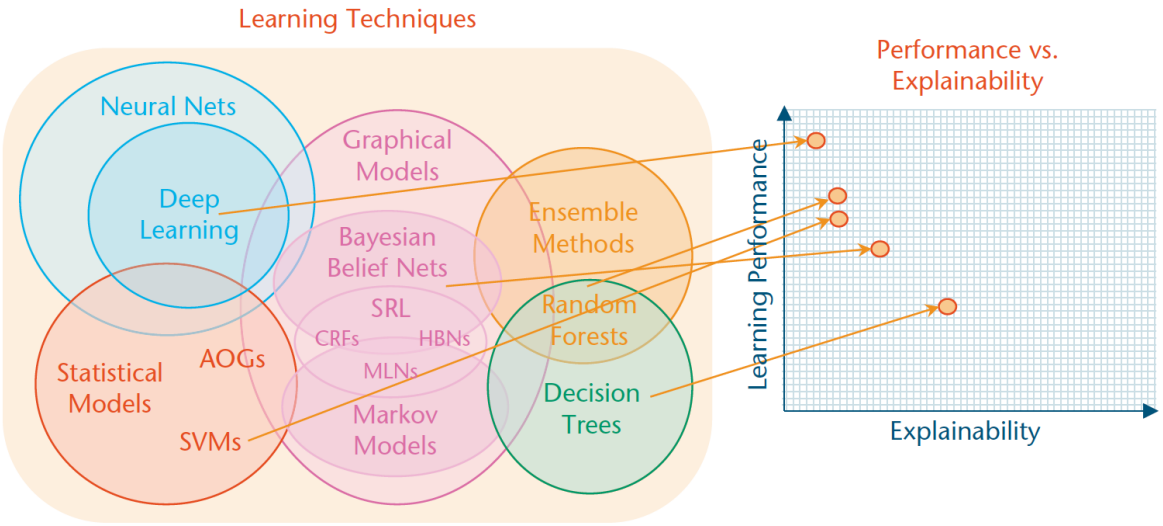

Figura 1: Trade off tra capacità di apprendimento e spiegabilità.

Nonostante l'aumento della dimensione della community dei ricercatori in materia di explainability[KVW18] ed esistano leggi che regolano determinate situazioni riguardanti le decisioni automatizzate, il tema del fornire spiegazioni su sistemi di IA trasparenti è

attualmente materia di dibattito per via dell'assenza di una definizione precisa di cosa costituisca una spiegazione e cosa la renda soddisfacente.

Si pensi a un applicazione che decide se accettare o rifiutare la richiesta di prestito di un cliente di una banca: quest'ultimo vorrà sapere, in caso di rifiuto, il motivo di tale risultato e come migliorare il proprio profilo per vedersi accettare la prossima richiesta; un giudice o, più in generale, qualcuno con lo scopo di regolare tale sistema, vorrà conoscere il comportamento generale di quest'ultimo in modo da decretare se è conforme o no alle leggi vigenti; uno sviluppatore potrebbe voler capire dove il modello è meno sicuro, in modo da migliorarne le prestazioni. L'assenza di una definizione precisa di spiegazione è dovuta al fatto che persone differenti in contesti differenti hanno necessità di descrizioni differenti. Tale mancanza ha generato ambiguità nella regolamentazione[WMF17].

La generazione attuale di sistemi di IA offre grandi vantaggi e benefici, ma il loro effetto viene talvolta limitato dall'impossibilità del sistema di spiegare le proprie decisioni all'utente. L'incremento del numero di sistemi di IA impiegati in settori ad alto rischio è direttamente correlato all'aumento della domanda di spiegazioni riguardanti le decisioni prese da questi particolari sistemi. Tale domanda ha fatto sì che venissero introdotte nuove regolamentazioni in merito alla richiesta di spiegazioni dei percorsi decisionali intrapresi dai sistemi di IA, con lo scopo di offrire all'utente una visione diretta di tali processi, così da aumentare confidenza e fiducia nei confronti dell'IA stessa. Il right to explanation è il diritto di ricevere una spiegazione per un determinato output di un algoritmo che incida in modo significativo sulla persona, in particolare dal punto di vista legale e finanziario.

Nell'aprile del 2018 è entrata in vigore la General Data Protection Regulation (GDPR)[1], che ha preso il posto delle Data Protection Directive (DPD) come regolamentazione europea sul mantenimento e l'utilizzo di informazioni personali: la differenza principale tra i documenti risiede nel fatto che, mentre la DPD era un insieme di direttive, ovve-

[1]Link al testo della GDPR: GDPR

ro un insieme di regole che ogni stato interpreta a suo modo traducendoli in leggi, la GDPR è un insieme di regolamentazioni, ovvero un insieme di leggi applicate ad ogni paese appartenente all'Unione Europea. L'articolo 71, in particolare, introduce il diritto dell'interessato a ricevere una spiegazione della decisione conseguita mediante un trattamento automatizzato qualora quest'ultima produca effetti giuridici che lo riguardano o incida significativamente sulla sua persona.

Sono numerosi i documenti[SZS+13][MDFFF17][PMJ+16] che evidenziano problematiche e debolezze di algoritmi di machine learning.

Nel 2016 una ricerca[ALMK16] sul Correctional Offender Management Profiling for Alternative Sanctions(COMPAS), software per la valutazione del rischio criminale, ha dimostrato che le previsioni effettuate da questo strumento erano inaffidabili e basate su presupposti razzisti.

Un'altro articolo[NYC15] interessante riguarda particolari Deep Neural Networks(DNNs) allenate per riconoscere, in base alla foto data loro in input, elementi come oggetti e animali presenti nell'immagine. L'articolo dimostra che si può fare in modo che l'AI sbagli la sua previsione in maniera eclatante, per esempio etichettando una libreria come un leone con una sicurezza di oltre il 90%, modificando l'input in maniera inpercettibile per l'occhio umano.

Come gli algoritmi che lavorano su immagini, anche le reti neurali che lavorano sul linguaggio naturale possono essere ingannate[JL17][LMA+17]. Queste vulnerabilità hanno dato vita a una nuova gamma di attacchi che sfruttano le debolezze dei sistemi di IA[GSS14][MMS+17][CW17].

Le spiegazioni hanno il compito di garantire equità e correttezza dell'algoritmo spiegabile, ma possono inoltre servire per identificare quali sono i problemi dell'algoritmo e in caso di previsioni sbagliate cosa ha causato l'errore in modo da poter risolvere il bug.

1.3 Valutare una spiegazione

Oltre a ciò che costituisce una spiegazione è motivo di interesse definire come valutare quest'ultima, ovvero come decretare se una spiegazione è "sufficientemente soddisfacente".

Un metodo interessante per la valutazione di una spiegazione è quello basato su due aspetti principali: l'interpretabilità e la completezza[GBY+18].

- L'interpretabilità riguarda il fatto di descrivere il funzionamento del sistema in una maniera compresibile agli umani. Questo requisito è legato a conoscenza, cognizione e pregiudizio dell'utente.

- La completezza fa riferimento all'accuratezza della spiegazione nel descrivere il funzionamento del sistema. Una spiegazione è più completa quando permette di anticipare il comportamento del sistema.

La sfida dello spiegare sistemi di IA risiede nel fatto di creare spiegazioni sia interpretabili che complete, ma questi requisiti sono difficili da soddisfare simultaneamente: le spiegazioni più accurate sono difficilmente comprensibili dalle persone e quelle più interpretabili non possiedono un gran potere predittivo.

Ma come billanciamo il bisogno di trasparenza e di eticità rispetto al desiderio di interpretabilità? Si consiglia[Her17] di non valutare l'interpretabilità delle spiegazioni solo in base all'interpretazione umana, perchè gli umani tendono ad avere pregiudizi nei confronti delle spiegazioni più semplici. Inoltre fare affidamento sulle valutazioni umane può portare i ricercatori a creare sistemi persuasivi piuttosto che trasparenti. Sono da evitare sia soluzioni semplici che peccano di completezza sia soluzioni complete ma difficilmente comprensibili. Quello che bisogna implementare è un trade off soddisfacente tra interpretabilità e completezza: il livello delle caratteristiche dipende da svariati fattori, come la tipologia di utente alla quale è indirizzato il sistema.

1.4 FICO Explainable Machine Learning Challenge

Ora introdurrò la soluzione che ha vinto la FICO Explainable Machine Learning Challenge[2] del 2018, una collaborazione tra Google, FICO e diverse università come Oxford, Imperial e Berkley.

Il tema della sfida era creare un modello di machine learning con un alto livello di accuratezza e spiegabilità usando un dataset comune fornito da FICO (Fair Isaac COrporation)[3], società americana di analisi dati focalizzata su servizi di credit scoring.

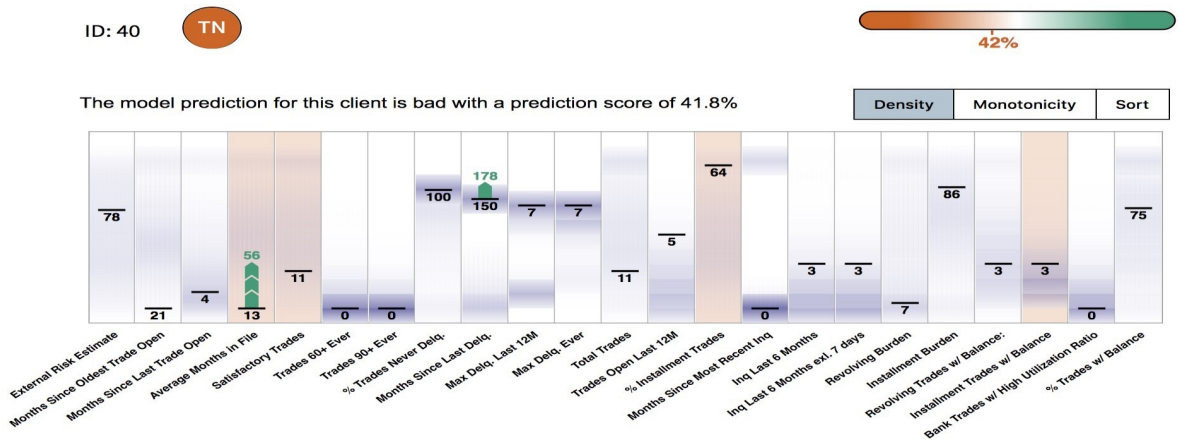

Figura 2: Spiegazione locale fornita dall'app vincitrice della FICO challenge

1.4.1 Il dataset

Una banca calcola il credit scoring di un richiedete finanziamento in base ai dati di quel cliente. In base a questo punteggio viene deciso se concedere o no il prestito.

Il dataset fornito per la sfida è il FICO HELOC dataset, una tabella contenente dati reali di anonimi richiedenti di un particolare tipo di linea di credito diffuso in America, l'Home Equity Line Of Credit(HELOC).

[2]Link alla pagina della competizione: FICO challenge
[3]Link alla homepage di FICO: FICO homepage

I campi del dataset sono 23 con valori interi, più un target con valore booleano, 'Good' o 'Bad', che indica la concessione o meno del prestito a quel determinato cliente. Il dizionario fornito insieme al dataset descrive i campi che lo compongono: in particolare troviamo una descrizione del significato della variabile e informazioni sulla sua monotonia e il suo ruolo.

Le descrizioni sono generalmente brevi e non esaustive, questo mi ha spinto a condurre una ricerca relativa a ogni singolo campo e a racchiuderne i risultati nella struttura che descriverò nei prossimi capitoli. La monotonia indica se per valori più alti di quel determinato dato la probabilità di ricevere il prestito aumenta o diminuisce. Il ruolo invece è comune per tutti i campi, tranne che per il valore target, e indica appunto il ruolo che questi hanno nella generazione della previsione(predicter).

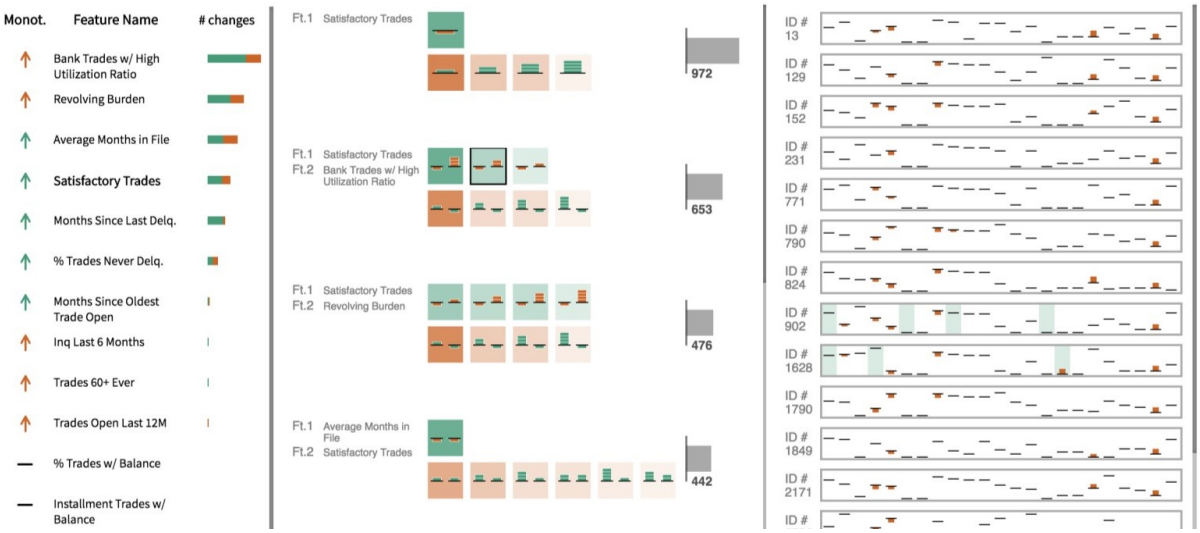

Figura 3: Spiegazione globale fornita dall'app vincitrice della FICO challenge

1.4.2 La soluzione

Il progetto vincitore della competizione è un sistema[HGB19] che genera spiegazioni locali relative ad un singolo utente e globali relative all'intero modello. Le figure 2 e 3 mostrano l'interfaccia che presenta le spiegazioni generate al fruitore del servizio.

Queste spiegazioni, per quanto complete possano essere, presentano un grado di interpretabilità basso e risultano criptiche e indecifrabili per un utente inesperto. Questi non avrà inoltre modo di espandere o argomentare la descrizione in alcun modo, limitando il target della spiegazione ai soli utenti esperti. Il requisito della comprensibilità risulta quindi insoddisfatto, a favore di un alto grado di completezza.

1.5 AI eXplainability 360

Sarà ora introdotto il toolkit AI eXplainability 360(AIX360) di IBM che fornirà l'algoritmo spiegabile il cui output sarà usato per generare la spiegazione di base nell'applicazione che userò per dimostrare la tesi. In seguito descriverò la soluzione fornita da IBM nella sua web demo per quanto riguarda la spiegazione del dataset FICO HELOC: tratterò in particolare i motivi di inadeguatezza della spiegazione che fornisce. Infine introdurrò l'idea che mi ha portato a migliorare tale spiegazione.

1.5.1 Il toolkit

Il toolkit AIX360 è una libreria open-source che supporta la comprensione di dataset e modelli di machine learning attraverso gli otto algoritmi spiegabili che incorpora. Nasce con lo scopo di portare la ricerca algoritmica in domini ad ampio raggio come finanza, sanità ed educazione.

Il pacchetto Python aix-360 è facilmente scaricabile e installabile, ma la documentazione consiglia di utilizzare un ambiente virtuale in quanto AIX360 richiede specifiche versioni di pacchetti Python che potrebbero generare conflitti con altri progetti nel sistema: in particolare viene consigliato Miniconda e vengono illustrati i passi necessari per il set-up dell'ambiente.

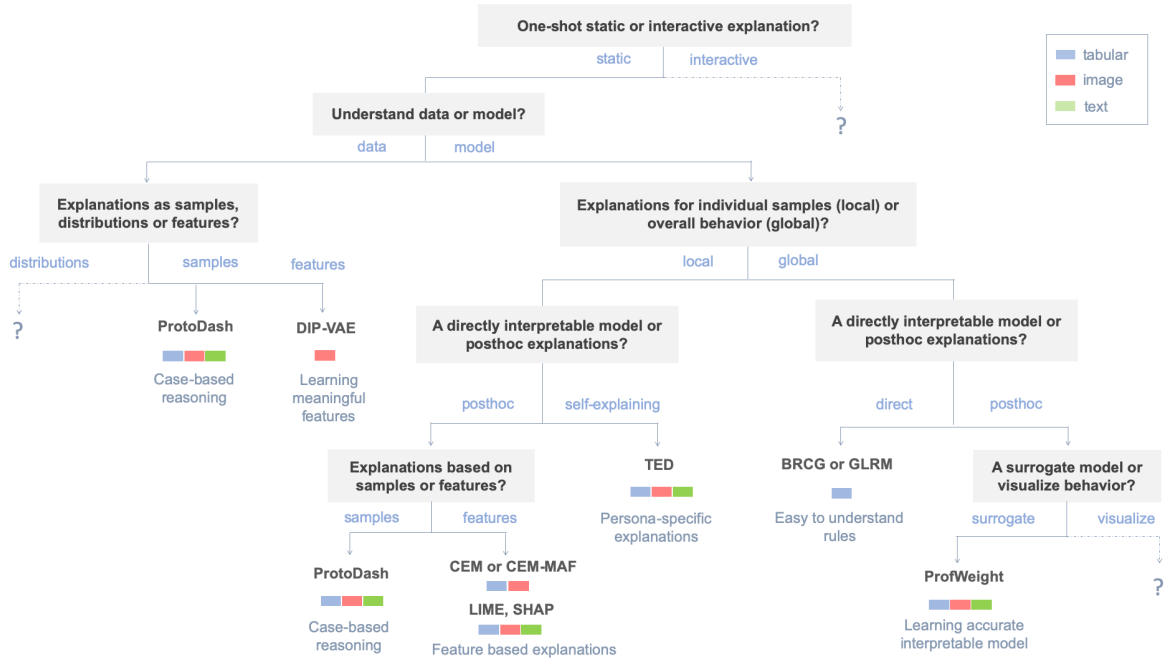

Figura 4: Metodo di selezione dell'algoritmo.

La figura 4 mostra una tassonomia degli algoritmi presenti nel toolkit basata su cosa si vuole spiegare(dataset o modello), come lo si vuole spiegare(metodo diretto o post-hoc, statico o interattivo) e a che livello(locale o globale). Le foglie dell'albero indicano il metodo attualmente disponibile nel toolkit per generare la spiegazione desiderata. I "?"indicano che non è ancora stato introdotto un metodo per quella specifica categoria.

In accordo con la documentazione di IBM possiamo classificare le spiegazioni nei modi seguenti:

- una spiegazione statica non cambia in base ai feedback ricevuti dall'utente, mentre una spiegazione interattiva permette a quest'ultimo di ampliare la spiegazione fino ad ottenere tutte le informazioni desiderate. Attualmente AIX360 non supporta algoritmi che generino spiegazioni interattive.

- una spiegazione locale è relativa ad un singolo elemento del dataset, mentre una spiegazione globale descrive l'ambiente dell'intero modello.

- una spiegazione direttamente interpretabile è una spiegazione diretta del modello, mentre una post-hoc è relativa a un modello che ne approssima uno a "scatola nera".

1.5.2 La web demo

IBM mette a disposizione una web demo che ha come tema quello di un software bancario che deve fornire spiegazioni a tre tipi di utenti differenti. Tali spiegazioni sono basate sul dataset messo a disposizione dalla FICO community per la Explainable Machine Learning Challenge: dataset contenente dati reali di utenti anonimi riguardanti una particolare linea di credito, tipicamente offerta dalle banche americane, chiamata HELOC (Home Equity Line Of Credit).

Per ognuno dei casi d'uso della web demo viene fornita una spiegazione diversa dello stesso dataset prodotta da un algoritmo diverso. Di seguito introduco gli algoritmi di machine learning spiegabili utilizzati e l'user case relativo ad ognuno di questi:

- Lo sviluppatore vuole assicurarsi che il sistema lavori in maniera appropriata prima di distribuirlo.

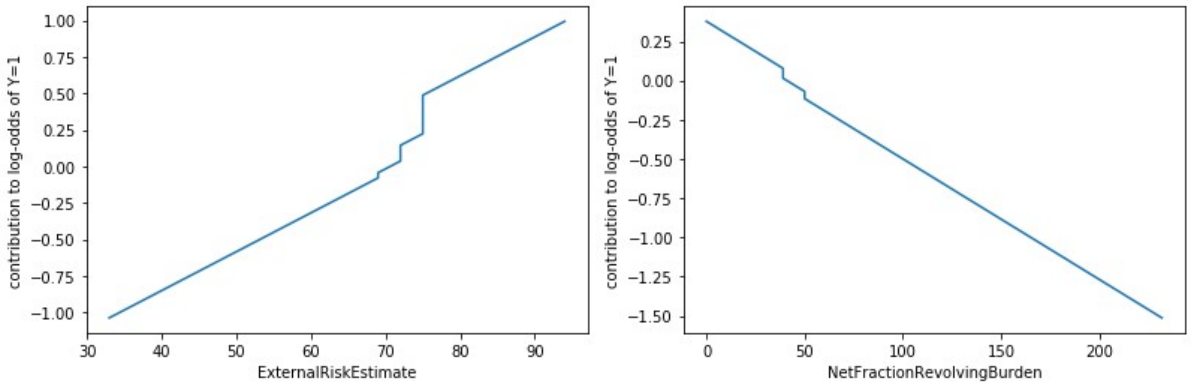

Figura 5: Peso di ExternalRiskEstimate e NetFractionRevolvingBurden

L'algoritmo usato è Generalized Linear Rule Model(GLRM)[DGW18] che fornisce spiegazioni globali e direttamente interpretabili su combinazioni pesate di regole generando grafici che mostrano il contributo della feature sulla previsione dell'algoritmo. Due grafici esemplificativi sono mostrati nella figura 5.

In questo modo lo sviluppatore può controllare la correttezza delle regole sulla previsione e verificare il peso del loro contributo.

- L'agente di prestito si basa sulla decisione dell'IA per prendere la decisione finale, accettando o rifiutando la richiesta del cliente. Vuole quindi capire come e perchè il modello ha prodotto quella determinata previsione.

  L'algoritmo usato è Protodash[GDCA19] che fornisce spiegazioni locali relative a un singolo cliente cercando nel dataset profili con valori simili e stesso valore target, detti prototipi, e generando una tabella che permette di paragonare il nuovo cliente con i prototipi utilizzati.

  In questo modo l'agente di prestito può confidare nella correttezza della previsione basandosi sui valori in comune tra cliente e prototipi.

- L'utente della banca vuole sapere come migliorare il suo profilo per fare in modo che il sistema accetti la sua prossima richiesta di prestito.

  L'algoritmo utilizzato è Contrastive Explanations Method(CEM)[DCL$^+$18] che genera spiegazioni locali relative all'utente attuale ricercando nel dataset il profilo più simile al quale il sistema ha concesso il prestito e generando una tabella che evidenzia le differenze tra questi due profili.

  In questo modo l'utente può identificare quali valori hanno fatto in modo che la sua richiesta di prestito venisse rifiutata.

Analiziamo ora il caso utente del cliente della banca, che è quello adottato nella web application I-Explaner introdotta nel prossimo capitolo: anche se resa human readable,

una spiegazione che fa riferimento a campi del dataset codificati in maniera incomprensibile non è soddisfacente per l'utente senza esperienza avanzata in materia di prestiti e che non può chiedere al sistema di argomentare la spiegazione fornita.

Il fatto che sia human readable è quindi un requisito necessario ma non sufficiente del tipo di spiegazione che stiamo perseguendo.

Le spiegazioni sopracitate presentano quindi due problematiche fondamentali: sono statiche e richiedono un livello di esperienza troppo elevato per essere capite dall'utente. Questi due problemi sono strettamente collegati e ho risolto entrambi progettando una spiegazione dinamica in maniera tale da renderla il più argomentabile possibile, così da poter spiegare qualsiasi termine causasse dubbi o perplessità all'utente.

Ho così introdotto quelli che sono i concetti principali utilizzati nel resto della dissertazione per dimostrare la tesi, gli approcci al problema precedenti alla mia soluzione e le idee che mi hanno portato a implementarla. I prossimi capitoli introdurranno I-eXplainer: prima verrà descritto il sistema ad alto livello, prestando particolare attenzione all'implementazione della spiegazione, poi più a basso livello focalizzandosi sui dettagli tecnici della soluzione. In seguito sarà introdotto il metodo di valutazione utilizzato per studiare I-eXplainer.

Capitolo 2

2 I-eXplainer

In questo capitolo introdurrò la mia soluzione partendo da una panoramica ad alto livello del sistema, cos'è, cosa fa e come lo si usa, passando dal metodo di implementazione della spiegazione esplorabile per poi andare a focalizzarmi sui punti di forza che gli permettono di risolvere i problemi lasciati aperti dalle soluzioni precedenti, in particolare dalla web demo di IBM.

2.1 Introduzione alla soluzione

La soluzione ideata è un explainer system, cioè un sistema con lo scopo di dimostrare all'utente la correttezza dell'output di un algoritmo spiegabile attraverso una spiegazione interattiva.

La spiegabilità dell'algoritmo è un requisito necessario. Non è possibile generare spiegazioni soddisfacenti riguardo alle decisioni e agli output di algoritmi a scatola nera, quindi per poter fare il suo lavoro, I-eXplainer richiede che l'algoritmo sia trasparente.

L'interattività del sistema è data dalla possibilità di interagire in maniera diretta con quest'ultimo ottenendo risposte in tempo reale: le interazioni dell'utente saranno richieste di delucidazioni su determinati termini presenti nella spiegazione e le risposte del sistema saranno pacchetti di informazioni relative a quegli specifici termini.

Le spiegazioni generate dal sistema sono human readable, locali e interattive: partendo da una spiegazione statica più complessa, l'utente ha la possibilità di interagirvi per chiarire dubbi e perplessità. In questo modo si ha la possibilità di esplorare la descrizione in maniera dinamica, andando ad ampliarla a piacimento fino a raggiungere il livello di soddisfazione desiderato.

La località della spiegazione è dovuta al caso utente utilizzato: al cliente della banca non interessano spiegazioni sull'ambiente dell'intero modello. Ciò non toglie che sia possibile, ed è prevista negli sviluppi futuri, fornire al cliente la possibilità di richiedere differenti tipi di spiegazioni al sistema.

Il beneficio per l'utente di una spiegazione esplorabile è il fatto che questa sia efficace per un range di fruitori più ampio: non ci si accontenta di generare spiegazioni capibili solo da utenti esperti, ma si costruisce in maniera dinamica una spiegazione nella quale ogni passo è ottenuto dallo sviluppo del passo precedente, nella direzione scelta dall'utente. Le scelte effettuate porteranno ogni utente ad ottenere una spiegazione diversa dalle altre, ma con il medesimo obiettivo. Lo scopo della spiegazione è quello di informare l'utente sui fattori che hanno fatto sì che l'algoritmo generasse quel determinato output.

2.2 L'applicazione

La soluzione è stata implementata attraverso una web app simile alla demo di IBM[4]. I dettagli implementativi dell'applicazione saranno forniti nel capitolo 3. Lo scenario utilizzato è quello dell'utente che vede rifiutata la sua richiesta di prestito perchè considerato non idoneo e usa l'applicazione per capire cosa ha indotto il sistema a giungere a tale conclusione.

Ho utilizzato uno degli script presenti negli esempi di IBM relativi al toolkit AIX360 per generare la spiegazione dell'algoritmo CEM[5]. Questa prima spiegazione non è tuttavia soddisfacente: CEM è un algoritmo spiegabile, ma la spiegazione generata non è human readable ed è capibile solo da utenti esperti. Ho quindi tradotto la tabella generata dallo script in una sequenza di stringhe human readable che costituiscono la spiegazione iniziale mostrata all'utente.

[4]Link alla webdemo: WEB DEMO

[5]Link al notebook di IBM: NOTEBOOK

La spiegazione iniziale non è sufficiente, in quanto fa uso di termini codificati in maniera difficilmente comprensibile. Le informazioni necessarie per capire il risultato dell'algoritmo sono tuttavia interamente contenute nella spiegazione iniziale: da qui l'idea di rendere la spiegazione comprensibile a un range di utenti più ampio attraverso lo sviluppo dei termini non chiari in quest'ultima. I-eXplainer sostanzialmente implementa un meccanismo di interazione tra l'utente e la spiegazione dell'algoritmo spiegabile.

Per dare la possibilità all'utente di espandere la spiegazione ho utilizzato una struttura dati che contenesse i blocchi di informazioni richiedibili dal fruitore dell'applicazione. I blocchi di informazioni sono triple RDF identificabili in modo univoco in base al termine a cui fanno riferimento e al tipo di informazione che contengono su quest'ultimo. Nel corso di questo documento chiameremo i termini contenuti in questa struttura termini spiegabili che a basso livello identificano le stringhe di caratteri per i quali l'utente può chiedere un approfondimento. I nodi sono collegati tra loro andando a creare un grafo, denominato grafo della conoscienza. L'implementazione della struttura sarà argomentata nel prossimo capitolo.

L'interfaccia utente evidenzia i termini spiegabili presenti nella descrizione notificando a chi utilizza il sistema la possibilità di interagirvi per ottenere maggiori informazioni sul termine attingendo al grafo della conoscienza. In questo modo si andranno ad aggiungere blocchi di informazioni alla spiegazione, espandendola ed aumentando il suo potere esplicativo.

Sono stati presi vari accorgimenti per garantire al sistema un buon grado di usabilità evitando di aggiungere funzionalità superflue o fuori luogo: ho minimizzato la ridondanza dei termini spiegati nella spiegazione e rimosso la possibilità di spiegare più volte lo stesso termine, inserito funzionalità per nascondere gli elementi non desiderati da quest'ultima e aggiunto tooltip sui termini spiegabili per permettere all'utente di non includere nella spiegazione informazioni non completamente necessarie. I dettagli implementativi delle varie funzionalità saranno discussi nel prossimo capitolo.

I-eXplainer permette un'interazione con la spiegazione che non è resa possibile da sistemi quali la web demo di IBM o l'applicazione vincitrice della FICO challenge. Questa interazione permette di partire da una spiegazione meno interpretabile di quelle offerte dalle soluzioni precedenti, ma di aumentare gradualmente la comprensibilità richiedendo spiegazioni specifiche e mirate sui termini spiegabili, ottenendo una spiegazione migliore rispetto a quelle offerte dai sistemi precedentemente descritti.

2.3 Interfaccia utente

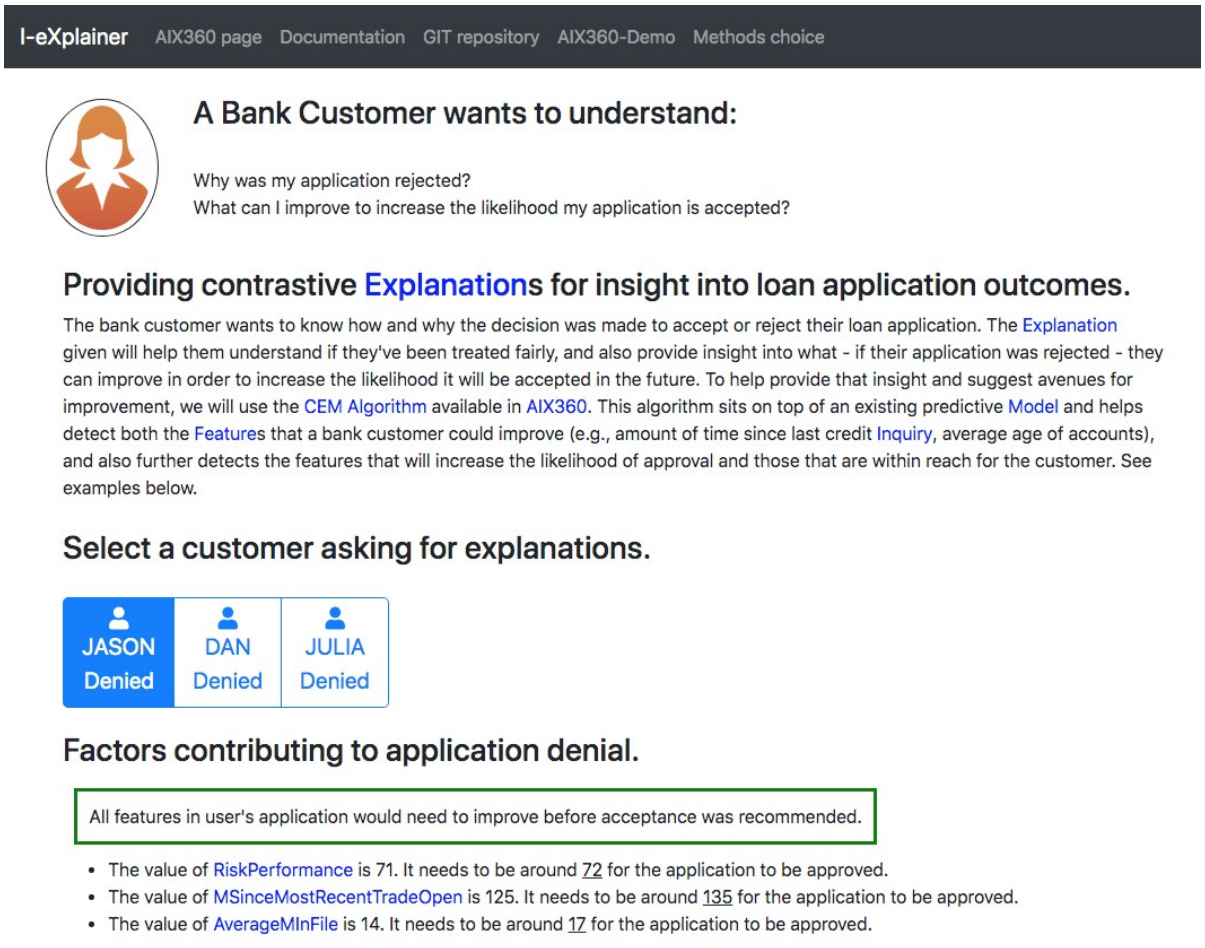

Figura 6: L'interfaccia utente di I-eXplainer

L'interfaccia utente di I-eXplainer si presenta semplice e minimale, ma senza trascurare usabilità e funzionalità. La figura 6 mostra l'interfaccia completa.

In testa alla pagina troviamo una navbar con alcuni link a contenuti utili, come la demo di IBM, il repository GitHub di AIX360 e la documentazione del toolkit.

Il corpo della pagina è formato da due elementi principali: l'introduzione e la spiegazione. L'introduzione è relativa al caso utente e all'algoritmo CEM: è minimale per quanto riguarda l'algoritmo, ma completamente interattiva, quindi in caso di dubbi su parole spiegabili è possibile richiedere informazioni aggiuntive. La spiegazione iniziale è formata da frasi del tipo "Il valore del campo A è X. Sarebbe dovuto essere Y affinchè la richiesta venisse approvata", dove A è il campo del dataset ed è un termine spiegabile, X è il valore del campo A nel profilo del cliente e Y è il valore del campo A nel profilo del cliente più simile che ha però ottenuto il prestito. Il mousover su Y mostra un tooltip che spiega la provenienza del valore.

L'interattività è implementata attaverso il click sulle parole spiegabili. Queste parole sono evidenziate all'utente attraverso una differente colorazione rispetto al resto del testo. Il click innesca il meccanismo che recupera dal grafo il blocco relativo a quel termine e lo renderizza e alloca nella spiegazione. Sarà poi possibile nascondere e mostrare i singoli nodi della spiegazione attraverso il click sugli appositi bottoni HIDE e SHOW. Inoltre ho implementato la lista in maniera tale che ogni nodo possa essere spostato, in modo da parmettere all'utente di ordinarli come preferisce.

Inoltre al mouseover sul termine spiegabile si apre un tooltip che permette di visualizzare la descrizione del termine senza includere effettivamente l'intero blocco di informazioni nella spiegazione, provvedendo a migliorare l'user-experience. Ho preso l'idea dall'interfaccia utente di Wikipedia. La figure 7 mostra i vari tooltip visualizzabili nell'interfaccia: oltre a quello sui termini spiegabili ci sono tooltip per le sorgenti delle informazioni e sul significato dei valori nella spiegazione.

Per evitare la ridondanza di termini spiegati nella spiegazione ho implementato un

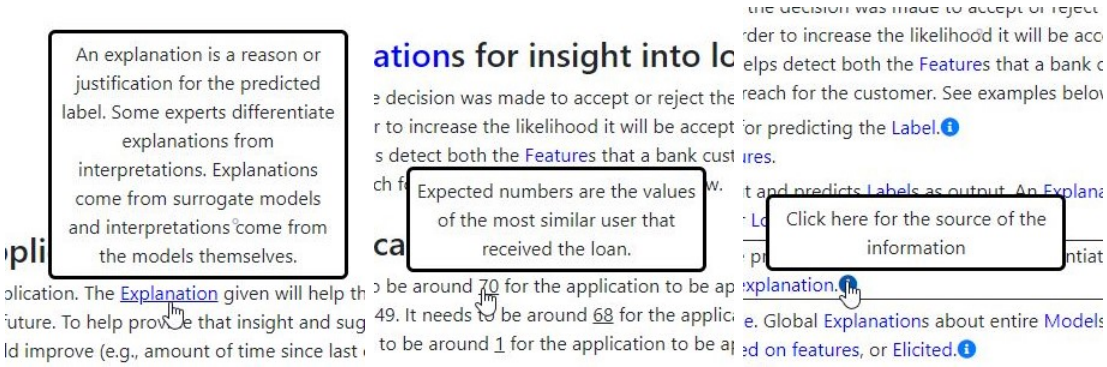

Figura 7: Tre esempi di tooltip nell'interfaccia grafica

meccanismo che impedisce di spiegare due volte lo stesso termine: se provo a spiegare un termine per la seconda volta, l'interfaccia utente evidenzia invece il blocco di informazioni relative a quel termine. Nel caso in cui il blocco sia stato precedentemente nascosto mediante il click sul bottone HIDE, l'interfaccia rende nuovamente visibile il nodo prima di evidenziarlo.

Ho implementato l'interfaccia utente utilizzando la libreria Bootstrap e JQuery UI. Ho usato le varie feature di AngularJS, soprattutto templating e binding, per la parte riguardante il contenuto della pagina. Le funzioni dell'interfaccia sono state interamente implementate in javascript, con l'ausilio della libreria jQuery.

Ho così introdotto ad alto livello la soluzione I-Explainer e la spiegazione che genera. Nel prossimo capitolo si passerà ad una descrizione a basso livello del sistema, dell'implementazione dei componenti e delle tecnologie usate. In seguito sarà introdotto il metodo di valutazione utilizzato per studiare I-eXplainer.

Capitolo 3

3 Dettagli implementativi

In questa sezione saranno descritti i dettagli tecnici dell'applicazione, analizzandone struttura concettuale, concreta e le implementazioni di server e client. Infine sarà discussa la realizzazione della struttura dati utilizzata.

L'idea di fondo era di implementare l'applicazione secondo una struttura il più modulare possibile: ogni modulo costituente l'app svolge un determinato set di funzioni lavorando su input e restituendo output formattati in una certa maniera. In questo modo si semplificano azioni come la sostituzione di un modulo, ad esempio per introdurne un altro implementato con una differente tecnologia.

Concettualmente l'applicazione ha bisogno dei seguenti componenti principali, ognuno con il suo insieme di funzioni:

- il server ha il compito di inviare al client gli statici, l'albero della conoscenza e l'output dell'algoritmo da trasformare nella prima spiegazione human readable. Inoltre salva le valutazioni degli utenti nel file apposito.

- il controller angular effettua le richieste al server per quanto riguarda grafo RDF e output di CEM. Inoltre si occupa di convertire le stringhe da mostrare nell'interfaccia utente in un formato che evidenzia i termini spiegabili e di effettuare le query di ricerca dei nodi nel grafo. Qui inoltre troviamo i contenuti della pagine, quindi testi, label e link, che sono mappati nella pagina html attraverso il binding del framework AngularJS.

- lo script client gestisce la lista delle parole spiegate e gestisce le interazioni dell'utente con l'interfaccia utente (come i click, i tooltip, la gestione del form di

valutazione e altro). In aggiunta costruisce e posiziona i blocchi di informazioni nella spiegazione.

Per quanto riguarda la struttura concreta riporto di seguito l'alberatura del progetto e una breve introduzione al contenuto delle singole cartelle:

```
/
├── resources
│   ├── explanation
│   │   └── cem_expl{id}.json
│   ├── graphs
│   │   └── graph.json
│   ├── script
│   │   ├── cem.py
│   │   ├── heloc.npz
│   │   └── heloc_nnsmall.h5
│   ├── static
│   │   ├── controller.js
│   │   ├── script.js
│   │   └── style.css
│   └── sus-form
│       ├── sus-form.html
│       ├── evaluations.json
│       └── sus-controller.js
├── index.html
└── server.js
```

La cartella explanation contiene la spiegazione non human readable relativa a CEM, mentre graph mantiene al suo interno il grafo della conoscienza. Script contiene i file relativi alla generazione delle spiegazioni: file python, modello allenato e dataset elaborato. Static contiene appunto gli statici che verranno richiesti dal client insieme alla pagina index.html.

Per il versionamento ho utilizzato Git: ho svolto la maggior parte degli sviluppi e dei test in locale, deployando sporadicamente l'ultima versione del repository su Gocker.

3.1 Server

Per realizzare una versione web dell'explainer system interattivo ho realizzato un server in grado di leggere il grafo rdf contenuto nel file json relativo e di inviarlo al client tramite servizio REST dedicato. Il server si occupa anche di produrre l'output dell'algoritmo CEM del toolkit AIX360 e di generare le spiegazioni non human readable che saranno inviate al client: ho fatto uso del pacchetto child_process per istanziare dinamicamente un processo python che eseguisse il codice nel file relativo all'algoritmo, contenuto nella cartella script. L'algoritmo elabora il file csv(Comma-Separed Values) del dataset contenuto nel pacchetto aix360 e lo usa per allenare il modello e costruire la spiegazione iniziale che sarà salvata nel file json cem_expl relativo all'utente. Ho implementato un servizio che si occupa di produrre la spiegazione dell'id passato come parametro dell'url e di salvarla in un file cem_expl{id}.json nella cartella explanations.

Questa prima spiegazione è una tabella di tre colonne e un numero di righe pari al numero di campi del dataset: la prima colonna contiene i valori dell'utente non idoneo, la seconda contiene quelli dell'utente idoneo più simile a quello della prima colonna e l'ultima contiene la differenza tra i valori delle prime due.

Questa prima spiegazione è una tabella di numeri e label che si presta a un'interpretazione impossibile da parte dell'utente non esperto, quindi è necessaria una lavorazione per renderla human readable in modo da generare la spiegazione di partenza che sarà visualizzata nell'interfaccia utente: questo processo è delegato al client perchè se una web page diversa volesse avere come punto di partenza una spiegazione diversa incontrerebbe più difficoltà ad analizzare la spiegazione human readable per risalire ai valori necessari piuttosto che crearne una ex novo.

Il dataset processato e il modello allenato vengono salvati rispettivamente nei file heloc.npz e heloc_nnsmall.h5 per questioni di performance in quanto processare il dataset e allenare il modello richiede tempo, quindi ripetere tutto il processo ogni volta che un utente richiede una spiegazione penalizzerebbe l'applicazione dal punto di vista dell'u-

sabilità. NPZ è un formato di file della libreria Python NumPy che permette di salvare array di dati usando la compressione gzip, mentre h5 è un formato di file contenenti array multidimensionali di dati scientifici.

Per testare le varie API realizzate nel server ho utilizzato l'estensione REST CLIENT[6] di Visual Studio Code. Questa permette di mantenere in maniera ordinata in un file .http la lista delle chiamate HTTP e di eseguirle in maniera semplice e visualizzando la risposta direttamente in un tab dell'IDE.

3.2 Client

Per l'implementazione del client ho realizzato una single page application attenendomi ai più recenti standard e utilizzando il framework AngularJS. Il compito della pagina è quello di contattare il server una volta caricata completamente e richiedere il grafo rdf dei termini spiegabili e la prima spiegazione dell'output dell'algoritmo CEM: quest'ultima sarà usata per produrre la spiegazione human readable iniziale, che verrà resa interattiva con l'ausilio del grafo della conoscenza.

I compiti relativi alla comunicazione client-server e quelli relativi all'elaborazione del grafo sono svolti dal controller angular. Il controller fornisce funzioni per elaborare i nodi del grafo, selezionare quelli richiesti dall'utente e rendere gli oggetti di tali nodi interattivi, ovvero marcare i termini che l'utente potrà cliccare per richiedere informazioni aggiuntive su di essi.

Il file script.js contiene invece le funzioni relative all'implementazione dell'interfaccia che si occupano della visualizzazione dei tooltip, della renderizzazione dei blocchi di informazioni, della loro allocazione nell'ui e dell'invio dei dati del form di valutazione al server.

Per evitare di poter spiegare più volte uno stesso termine ho affibbiato a ogni nodo della spiegazione un attributo che indica quale termine spiega quel determinato nodo.

[6]Link al repository Github di REST CLIENT: link.

Utilizzando le funzioni di ricerca nel DOM di JQuery ho fatto in modo di verificare se quella spiegazione contiene già un nodo contenente informazioni su un determinato termine spiegabile. In questo modo quando l'utente tenta di ampliare la spiegazione controllo se il termine è già stato spiegato: solo in caso negativo procedo con la creazione del nuovo blocco, mentre in caso positivo non faccio niente, ma il mouseover su termini già spiegati evidenzia il blocco relativo a quel termine. In questo modo evito che si possano creare spiegazioni infinitamente lunghe con termini ridondanti.

Ho usato la libreria Bootstrap per le componenti offerte, come la navbar e i vari elementi che contiene, il form di valutazione, gli alert per i blocchi della spiegazione e il pagegrid per l'intelaiatura della pagina. Ho utilizzato la libreria JQuery UI, un'estensione di JQuery, per aggiungere la funzione che permette di cambiare l'ordine ai nodi nella spiegazione direttamente dall'interfaccia grafica.

Ho usato le funzioni della libreria JQuery per la manipolazione e la ricerca di nodi nel DOM e per effettuare le chiamate AJAX alle varie API del server.

3.3 Codifica delle informazioni

Il grafo della conoscenza è implementato come una lista di quadruple RDF. Ogni singolo nodo contiene quattro campi di tipo stringa:

- Il soggetto indica il termine a cui fa riferimento l'informazione contenuta nel blocco. L'insieme dei soggetti corrisponde all'insieme dei termini spiegabili per i quali posso richiedere informazioni attraverso l'interfaccia grafica di I-eXplainer.

- Il predicato identifica il tipo di informazione. In seguito verranno elencati i predicati presenti nel grafo e il relativo significato, ma ogni soggetto ha almeno due tipologie di informazioni: il tipo e la descrizione.

- L'oggetto è una stringa human readable che contiene l'informazione vera e propria.

- La sorgente è un url alla pagina dove ho reperito l'informazione contenuta nell'oggetto. In seguito elencherò le principali fonti di informazioni adottate per reperire gli oggetti del grafo.

I nodi sono identificabili in modo univoco attraverso la chiave soggetto-predicato(identificazione implementata nel controller). Le funzioni necessarie per la ricerca di nodi nel grafo sono due: ricerca per chiave primaria di singoli nodi e ricerca per soggetto di insiemi di nomi. La prima fa uso del costrutto JavaScript findper trovare il nodo che soddisfi la guardia e la seconda usa il forEach per scorrere tutti i nodi del grafo e costruire progressivamente la lista da ritornare.

Il contesto può essere visto come l'ambiente dentro il quale le affermazioni codificate in triple rdf sono considerate vere. Nel knowledge graph faccio uso del mio specifico contesto usando il prefisso "my:"e riferendomi all'uri https://site181963.tw.cs.unibo.it/graph.

Di seguito riporto un singolo nodo a titolo esemplificativo:

```
{
"subject": "my:XXX",
"predicate": "my:YYY",
"object": "ZZZ",
"source": "URL"
}
```

I predicati utilizzati nel grafo della conoscenza sono i seguenti:

- description: predicato comune, l'oggetto descrive il soggetto.
- type_of: predicato comune, l'oggetto fornisce una classificazione per tipo del soggetto.
- explanation_types: l'oggetto indica il tipo di spiegazione prodotta dal soggetto.

- used_for: predicato generico che indica dove viene impiegato il soggetto.
- understand: l'oggetto indica cosa analizza il soggetto: es. il dataset o il modello.
- opposite: predicato generico, l'oggetto corrisponde all'opposto del soggetto.
- used_dataset: predicato specifico, l'oggetto indica il dataset utilizzato.
- kind_of: l'oggetto contiene il supertipo del soggetto.
- algorithms: predicato generico, l'oggetto contiene algoritmi associati al soggetto: es. di quel tipo o appartenenti a quella categoria.
- fields: predicato specifico, l'oggetto contiene i campi del dataset nel soggetto.
- field_of: predicato specifico, l'oggetto contiene il dataset a cui appartiene il soggetto.
- monotonicity_constraint: predicato specifico, l'oggetto contiene il vincolo di monotonia del campo contenuto nel soggetto: es. monotona crescente, monotona decrescente.

I predicati specifici, al contrario di quelli comuni, sono utilizzati solo e unicamente per determinati soggetti.

Ho recuperato la maggior parte delle informazioni contenute nel grafo da Investopedia(link), il glossario di FICO(link) e quello presente nella documentazione del toolkit(link).

Ho implementato la connessione tra i nodi del grafo attraverso l'ausilio della funzione MakeExplainable: quest'ultima cerca nell'oggetto che si vuole rendere interattivo tutte le occorrenze di ogni soggetto presente nel grafo e le marchia con un tag comune. Lo script della pagina imposta un listener per ogni elemento html con questo tag: la funzione in ascolto usa il soggetto marchiato per cercare i nodi contenenti informazioni su quel termine spiegabile nel grafo.

Il problema che ha causato più difficoltà, per quanto riguarda l'implementazione delle connessioni del grafo, è quello che riguarda la disambiguazione dei termini spiegabili: i casi in cui volevo spiegare termini spiegabili che contenevano altri termini spiegabili(ad esempio "local explanation" contiene il termine "explanation") creavano conflitti nel processo di interattivizzazione, rischiando di associare la spiegazione sbagliata al termine richiesto. Per risolvere i conflitti ho pensato di formattare i soggetti e i soggetti contenuti negli oggetti utilizzando un carattere speciale al posto degli spazi, l'underscore, sostituendoli poi con gli spazi prima di mostrarli nell'interfaccia.

Capitolo 4

4 Valutazione

In questo capitolo introdurrò il metodo che ho usato per valutare I-eXplainer e la metodologia utilizzata per recuperare i dati dagli utenti ed elaborarli.

4.1 System Usability Scale

L'ISO definisce l'usabilità di un sistema come efficacia, efficienza e soddisfazione con le quali l'utente raggiunge un determinato obiettivo in un determinato contesto. In pratica si tratta del grado di facilità con cui avviene l'interazione tra utente e sistema.

L'efficacia corrisponde alla completezza e accuratezza con le quali un utente raggiunge il suo obiettivo, mentre l'efficienza identifica le risorse spese in relazione all'accuratezza e alla completezza degli obiettivi raggiunti. Lo studio della metodologia di valutazione ha escluso la possibilità di valutare efficacia ed efficienza di I-eXplainer per motivi di assenza delle risorse necessarie.

Ho deciso quindi di stimare la soddisfazione del mio sistema rispetto a quella di AIX360-Demo: soddisfazione intesa come comfort del sistema per i suoi utenti. Mentre efficacia ed efficienza sono misure oggettive, la soddisfazione è soggettiva, quindi dipendente dal singolo soggetto. Essendo una misura soggettiva, la soddisfazione può essere perturbata da fattori esterni non legati al sistema in esame, come l'umore dell'utente e i suoi pregiudizi.

Ho utilizzato un questionario System Usability Scale(SUS) per valutare la soddisfazione dell'utente dopo l'utilizzo di ognuna delle due web app in esame: I-eXplainer e AIX360-Demo. Il SUS è stato ideato come test generico e veloce per stimare la soddisfazione dell'utente in merito a un determinato sistema: non diagnostica i problemi specifici, ma permette di verificare se l'usabilità del nostro sistema è accettabile o se

bisogna lavorarci ancora. Il SUS è indipendente dalla tecnologia in esame: trova impiego nei test di hardware, software, cellulari, siti web e altro ancora.

Le caratteristiche principali di questo mezzo di valutazione sono il basso costo a livello di risorse da amministrare e la velocità del test: esistono infatti svariati template del questionario pronti all'uso e varie guide su come interpretarne i dati.

Il questionario è articolato in 10 frasi, elencate di seguito, che alternano affermazioni positive ad affermazioni negative(dispari=positive, pari=negative):

- "Penso che mi piacerebbe utilizzare questo sistema frequentemente."
- "Ho trovato il sistema complesso senza che ce ne fosse bisogno."
- "Ho trovato il sistema molto semplice da usare."
- "Penso che avrei bisogno del supporto di una persona già in grado di usare il sistema."
- "Ho trovato le varie funzionalità del sistema ben integrate."
- "Ho trovato incoerenze tra le varie funzionalità del sistema."
- "Penso che la maggior parte delle persone potrebbero imparare ad utilizzare il sistema facilmente."
- "Ho trovato il sistema molto macchinoso da utilizzare."
- "Ho avuto molta confidenza con il sistema durante l'uso."
- "Ho avuto bisogno di imparare molti processi prima di riuscire ad utilizzare al meglio il sistema.

Le affermazioni 4, 8 e 10 riguardano la proprietà di learnability del sistema, mentre le affermazioni 1, 2, 3, 5, 6, 7, 8 e 9 si riferiscono alla proprietà di usabilità del sistema.

L'utente valuta ognuna delle frasi precedenti con un valore da 1 a 5 che indica il suo accordo o disaccordo con la frase associata: 1 significa che si trova fortemente in disaccordo con l'affermazione, 5 invece significa che è pienamente d'accordo con quest'ultima.

Per calcolare il risultato SUS ho adottato il seguente procedimento:

- per ogni frase dispari ho sottratto 1 alla risposta(<risposta>- 1).
- per ogni frase pari ho sottratto la risposta a 5(5 - <risposta>).
- ho preso i 10 valori ottenuti e li ho sommati, ottenendo un valore in quarantesimi, cioè da 0 a 40.
- ho moltiplicato il totale per 2.5 per ottenere un valore in centesimi, cioè da 0 a 100.
- ho fatto la media tra i SUS relativi al sistema in esame, così da ottenere il risultato SUS per quel sistema.

Il risultato ottenuto è in centesimi e non è una percentuale, ma un valore calcolato con un procedimento che vuole fornire una maniera chiara di visualizzare il punteggio.

Nel sito MeasuringU[7] viene indicato 68 come valore di soglia per un sistema con una buona usabilità. Il sito prende in esame i punteggi di un insieme di 500 SUS relativi a circa 5000 utenti e riguardanti differenti tecnologie. Il valore è stato calcolando facendo la media dei risultati dei SUS presi in esame.

In questo testo considero 68 come valore di sufficienza dell'usabilità. Se il punteggio è minore di 68 significa che probabilmente ci sono problemi riguardanti l'usabilità del sistema in esame.

4.2 Il test

Il test è stato svolto fornendo ai collaudatori un pdf contenente tutte le informazioni necessarie: caso d'uso e link alle web app e alla pagina del questionario, scopo, motivo

[7]Link a MeasuringU: LINK

e descrizione del test. Ho reso disponibile il questionario implementando una web page che presenta un form con le varie frasi, un campo per il nome del tester e uno switch per identificare quale applicazione si sta testando.

I risultati sono stati raccolti in un file JSON e elaborati da una API implementata nel server che restituisce un oggetto contenente i risultati del questionario dei singoli utenti e il gruppo di appartenenza, i valori elaborati secondo la metodologia presentata in precedenza e i valori medi dei SUS relativi ai due sistemi testati.

Di seguito riporto i risultati relativi agli utenti in forma anonima che hanno testato l'applicazione(le righe relative ad I-eXplainer e ad AIX360-Demo sono in ordine di quale sistema è stato testato prima e quale dopo):

Tabella 1: Dati raccolti attraverso il questionario SUS: gruppo 1.

Utente	App testata	Frase										Risultato
		1	2	3	4	5	6	7	8	9	10	
1	I-eXplainer	1	3	3	5	3	5	1	2	1	1	32.5
	AIX360-Demo	2	3	2	5	4	2	1	2	3	1	47.5
2	I-eXplainer	2	2	4	5	1	1	3	2	3	1	55
	AIX360-Demo	4	2	5	1	1	1	3	1	4	2	75
3	I-eXplainer	3	1	3	3	4	2	5	3	3	2	67.5
	AIX360-Demo	5	2	2	4	3	1	5	3	3	5	57.5
4	I-eXplainer	4	2	4	3	5	3	5	2	4	1	77.5
	AIX360-Demo	3	5	2	5	3	1	3	1	3	1	53.5
5	I-eXplainer	5	2	5	1	4	1	4	3	5	3	82.5
	AIX360-Demo	5	3	4	2	5	1	4	3	5	1	82.5

I risultati del test dimostrano che I-eXplainer presenta un grado di usabilità superiore rispetto ad AIX360-Demo: la media dei punteggio del questionario è infatti di 73.75 punti

Tabella 2: Dati raccolti attraverso il questionario SUS: gruppo 2.

Utente	App testata	Frase										Risultato
		1	2	3	4	5	6	7	8	9	10	
6	AIX360-Demo	3	5	1	4	5	3	2	2	2	4	37.5
	I-eXplainer	3	1	4	2	5	2	4	1	4	2	80
7	AIX360-Demo	1	5	1	5	2	2	2	2	1	5	20
	I-eXplainer	1	3	3	4	4	3	4	1	3	4	50
8	AIX360-Demo	4	1	4	2	4	1	5	2	4	2	82.5
	I-eXplainer	5	1	5	1	3	1	5	1	4	2	90
9	AIX360-Demo	2	3	4	2	4	5	5	3	5	1	65
	I-eXplainer	5	2	5	1	5	1	3	3	5	1	87.5
10	AIX360-Demo	4	2	5	1	3	3	4	2	4	2	75
	I-eXplainer	5	1	5	1	5	3	5	2	4	1	90

per il primo sistema, contro i 57 punti per il secondo.

Si nota inoltre che il mio sistema ha ricevuto valutazioni maggiori rispetto al suo avversario nel gruppo 2, ovvero quando veniva testato in seguito ad AIX360-Demo. Reputo positivo questo fatto in quanto dimostra che gli utenti hanno trovato I-eXplainer migliore in termini di usabilità rispetto alla prima applicazione testata.

In conclusione un punteggio medio di 73.75 punti nel questionario indica che l'usabilità del sistema I-eXplainer può essere considerata sufficiente, ma con un buon margine di miglioramento.

Ho così introdotto il metodo di valutazione usato per valutare il mio sistema I-eXplainer rispetto a AIX360-Demo e la metodologia utilizzata per raccogliere i dati relativi alla soddisfazione dei singoli utenti e per elaborarli. Nel prossimo capitolo verranno ricapitolati i punti trattati nei capitoli precedenti e verranno accennate le possibili

migliorie apportabili alla soluzione, discutendone anche i problemi che si troverà ad affrontare chiunque volesse implementarle.

Capitolo 5

5 Conclusioni

Con questa dissertazione ho voluto introdurre un metodo interattivo di spiegazione dell'output di algoritmi di IA spiegabili, metodo per produrre spiegazioni human readable esplorabili in base al desiderio dell'utente.

Inizialmente ho introdotto i concetti fondamentali, necessari per capire a pieno i capitoli che compongono il presente documento.

L'obiettivo era quello di generare una spiegazione interattiva per l'output dell'algoritmo CEM del toolkit AIX360 che potesse svilupparsi in maniera tale da diventare capibile da una fascia di utenti più estesa.

Per farlo ho sviluppato un'applicazione web che permette al fruitore del servizio di interagire, attraverso l'interfaccia utente, con la spiegazione in maniera tale da comunicare al sistema quali sono i termini che desidera spiegare. In questo modo il modulo explainer preleva nodi contenenti informazioni su tale termine da un grafo della conoscenza fornito dal server e li mostra all'utente, incrementando la dimensione della spiegazione con questo nuovo blocco di informazioni.

Ho descritto la struttura concettuale e concreta del sistema da me costruito, l'implementazione della struttura dati contenente le informazioni accessibili dall'utente e le principali tecnologie utilizzate per implementare quanto appena citato.

Ho descritto il metodo di valutazione dell'usabilità di I-eXplainer rispetto al sistema da cui prende spunto, AIX360-Demo. In seguito ho mostrato i risultati del questionario SUS compilato dagli utenti che hanno testato i due sistemi e definito le conclusioni traibili da tali risultati.

La valutazione dei due sistemi ha dimostrato che I-eXplainer presenta un'usabilità maggiore rispetto a AIX360-Demo e l'utente ha manifestato più soddisfazione in seguito

all'utilizzo della mia applicazione rispetto a quella manifestata in seguito all'utilizzo della demo di AIX360.

Ritengo accettabili i risultati ottenuti dal mio lavoro, ma nonostante questo c'è del margine di miglioramento.

Una futura versione aggiungerà la possibilità di espandere il grafo tramite i consigli dell'utente e di reperire informazioni su termini non presenti nel grafo da Wikipedia utilizzando DBpedia: questi aggiornamenti richiederanno però la gestione di una nuova gamma di problematiche, come la necessità di un automatismo per validare le informazioni suggerite dall'utente o per risolvere casi di omonimia e polisemia, in cui abbiamo termini che ammettono più significati e dobbiamo implementare un metodo che permetta all'explainer system di scegliere quale spiegazione dare all'utente per quel determinato lessema.

Riferimenti bibliografici

[ALMK16] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias risk assessments in criminal sentencing. *ProPublica, May*, 23, 2016.

[AOS+16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. doi:https://arxiv.org/abs/1606.06565.

[BY14] Nick Bostrom and Eliezer Yudkowsky. The ethics of artificial intelligence. *The Cambridge handbook of artificial intelligence*, 1:316–334, 2014. doi:https://svia.nl/static/files/ethics_of_artificial_intelligence.pdf.

[CMMS12] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333 – 338, 2012. Selected Papers from IJCNN 2011. doi:http://www.sciencedirect.com/science/article/pii/S0893608012000524.

[CW17] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017. doi:https://arxiv.org/abs/1705.07263.

[D+02] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002. doi:https://courses.cs.washington.edu/courses/cse446/12wi/tgd-ensembles.pdf.

[DCL+18] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In

Advances in neural information processing systems, pages 592–603, 2018. doi:https://arxiv.org/abs/1802.07623.

[DGW18] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems*, pages 4655–4665, 2018. doi:https://arxiv.org/abs/1805.09901.

[GBY+18] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018. doi:https://arxiv.org/abs/1806.00069.

[GDCA19] Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 260–269. IEEE, 2019. doi:https://ieeexplore.ieee.org/abstract/document/8970791.

[GF17] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017. doi:https://wvvw.aaai.org/ojs/index.php/aimagazine/article/view/2741.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. doi:https://arxiv.org/abs/1412.6572.

[Her17] Bernease Herman. The promise and peril of human evaluation for model interpretability. *arXiv preprint arXiv:1711.07414*, 2017. doi:https://arxiv.org/abs/1711.07414.

[HGB19] Steffen Holter, Oscar Gomez, and Enrico Bertini. Fico explainable machine learning challenge. 2019. doi:http://www.ml-explainer.com/static/images/FICO_paper.pdf.

[Hle19] AI Hleg. Ethics guidelines for trustworthy ai. *B-1049 Brussels*, 2019. doi:https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[JL17] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017. doi:https://arxiv.org/abs/1707.07328.

[KVW18] Been Kim, Kush R. Varshney, and Adrian Weller. Proceedings of the 2018 icml workshop on human interpretability in machine learning (whi 2018), 2018. arXiv:1807.01308.

[LMA+17] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017. doi:https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2782&context=cstech.

[MDFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. doi:http://openaccess.thecvf.com/content_cvpr_2017/html/Moosavi-Dezfooli_Universal_Adversarial_Perturbations_CVPR_2017_paper.html.

[MMS+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial

attacks. *arXiv preprint arXiv:1706.06083*, 2017. doi:https://arxiv.org/abs/1706.06083.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015. doi:https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Nguyen_Deep_Neural_Networks_2015_CVPR_paper.html.

[Ott13] Clemens Otte. Safe and interpretable machine learning: A methodological review. In *Computational intelligence in intelligent data analysis*, pages 111–122. Springer, 2013. doi:https://link.springer.com/chapter/10.1007/978-3-642-32378-2_8.

[PMJ+16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016. doi:https://ieeexplore.ieee.org/abstract/document/7467366.

[RPT10] Salvatore Ruggieri, Dino Pedreschi, and Franco Turini. Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(2):1–40, 2010. doi:https://dl.acm.org/doi/abs/10.1145/1754428.1754432.

[SHG+15] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing*

systems, pages 2503–2511, 2015. doi:http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. doi:https://arxiv.org/abs/1312.6199.

[TWB17] Andreas Theodorou, Robert H Wortham, and Joanna J Bryson. Designing and implementing transparency for real time inspection of autonomous robots. *Connection Science*, 29(3):230–241, 5 2017. doi:https://www.tandfonline.com/doi/abs/10.1080/09540091.2017.1310182.

[VA17] Kush R Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255, 2017. doi:https://www.liebertpub.com/doi/abs/10.1089/big.2016.0051.

[WMF17] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

Lista delle figure

Elenco delle figure