UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

TESI DI LAUREA

Sistemi di Machine Learning Fair: Un'Analisi Empirica dello Stato della Pratica

RELATORE

Prof. **Fabio Palomba**

Università degli studi di Salerno

CANDIDATO

Carmine Ferrara

Matricola: 052250090

Anno Accademico 2021/2022

When you think of A.I., it's forward-looking.
But A.I. is based on data, and data is a reflection of our history - Joy Buolamwini

Sommario

Oggigiorno, è sempre più importante osservare come applicativi della società moderna richiedano soluzioni di intelligenza artificiale per la più variegata tipologia di problemi, alcuni degli esempi più noti sono sistemi di diagnostica per le malattie, sistemi bancari per la concessione dei prestiti, ma ne esistono tanti altri. Al fine di operare in maniera opportuna nel settore di utilizzo, queste soluzioni necessitano che tutta una serie di prerogative non funzionali vengano soddisfatte, tra le più importanti vi è senz'altro considerare l'abilità di lavorare in maniera eticamente corretta. In altri termini, come suggerisce la ricerca è necessario che un sistema AI-Intensive che si rispetti debba essere necessariamente Fair-Aware, cioè libero da bias di tipo etico che ne influenzino l'operato per alcune tipologie minoritarie di utenti. Il mondo della ricerca, in particolar modo le comunità di ricerca inerenti l'intelligenza artificiale e l'ingegneria del software, ha studiato il problema della Fairness in maniera più approfondita, osservando che il concetto è difficilmente generalizzabile e dipende molto dalle caratteristiche degli specifici problemi. In particolare, molte sono le definizioni di Fairness emerse dai differenti studi e di varia natura sono le metriche ad esse correlate. Ma effettivamente, fin dove queste problematiche sono considerate nella pratica? Come esperti, quali data scientist o ingegneri del software si approcciano a questo problema? In che modo la Fairness viene considerata durante lo sviluppo? E soprattutto in quali fasi del ciclo di vita di una soluzione AI-Intensive viene considerata?. Obiettivo principale di questo lavoro di tesi è, quindi, quello di interrogare gli esperti del settore, al fine di fornire un'idea più precisa di come il concetto di Software Fairness sia recepito nella pratica, identificare quali sono le vulnerabilità maggiori che possono rendere un tool AI eticalmente scorretto e soprattutto capire e formalizzare quali possono essere le best-practice da adottare quando il concetto di Software Fairness è una delle prerogative fondamentali da rispettare durante l'intero ciclo di vita di un sistema AI-Intensive.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Motivazioni e Obiettivi

1.2 Risultati

1.3 Struttura della tesi

CAPITOLO 2

Background

Al giorno d'oggi è sempre più frequente adottare soluzioni che facciano uso di moduli intelligenti all'interno dei più svariati ambiti lavorativi, ed è oramai risaputo come tali soluzioni debbano rispettare tutta una serie di standard qualitativi, affinché possano essere ritenuti attendibili ai loro scopi. Tuttavia recenti studi hanno dimostrato un nuova gamma di difetti che evidenziano una serie di vulnerabilità, fin ora non riscontrate all'interno dello sviluppo software (tra cui, come verrà approfondito in seguito in questo capitolo, quelle dovute alla la Software Fairness), legati all'operare in maniera imparziale ed equa nel loro contesto di utilizzo [1]. Per capire come il mondo dello sviluppo software (ed in particolare quello delle soluzioni "intelligenti"), siano influenzati dagli aspetti di equità o un qualsiasi altro aspetto qualitativo in generale è doveroso introdurre gli aspetti essenziali che si pongono alla base dello sviluppo di tali applicativi.

2.1 Ingegneria del Software

Prima di tutto è senz'altro necessario far riferimento all'ingegneria del Software, disciplina che nasce proprio in risposta alle problematiche di sviluppo di prodotti software di qualità in un preciso tempo, con uno specifico budget [2]. L'ingegneria del Software si pone l'obiettivo di applicare tutta una serie di attività che facciano dello sviluppo software un vero e proprio processo ingegneristico [3]. Le attività cardine della materia, cosi come definite da Bernd Bruegge e Allen Dutoit, [2], sono:

- La *modellazione*: capacità standard di focalizzarsi sui dettagli rilevanti ignorando tutto il resto, tramite svariate strategie di raffinamento e astrazione;
- Il *problem solving*: l'utilizzo di modelli per la ricerca di soluzioni a specifici problemi;
- L' *acquisizione di conoscenza*: la raccolta di singoli dati e informazioni di uno specifico dominio, per poi formularne informazioni e conoscenza prima di applicare standard di sviluppo;
- La *ricerca di un razionale*: cioè la ricerca delle motivazioni e delle necessità che si pongono a priori dello sviluppo di un tool software;

Si osserva quindi come effettivamente un tool software per essere progettato in maniera congrua a molteplici aspetti qualitativi - ad esempio: sicurezza, manutenibilità e adattabilità (o qual si voglia tipologia di attributo non funzionale) - che soddisfino le aspettative del cliente che lo commissiona, sia necessario applicare un processo standard di ingegnerizzazione. Nello specifico, anche problematiche legate all'emergente tematica dell'equità e dei vincoli di imparzialità devono essere trattati sullo stesso piano di un qualsiasi altro attributo qualitativo e la ricerca si sta muovendo sempre di più in questa direzione [1].

Ovviamente, è molto complesso cercare di riassumere in poche parole ogni peculiarità dell'ingegneria del software, molteplici sono gli aspetti di disciplina che caratterizzano un prodotto software, e.g. il suo ciclo di vita, la sua manutenzione ed evoluzione, e tutti gli aspetti correlati alla sua gestione. Lo stesso testo di riferimento [2], definisce l'ingegneria del software, come un attività complessa, non come un algoritmo standard, quindi è importante osservare come essa si adatti in maniera dinamica, agli specifici problemi, e a seconda dei casi faccia uso in maniera oculata di strumenti di vario tipo, come la sperimentazione e la misurazione, il riuso di pattern o l'evoluzione incrementale dei sistemi [2] e altre variegate tecniche e metodologie che possano offrire pronta risposta alla più ampia gamma di problematiche e necessità specifiche. Volendo porre un esempio più specifico di come l'ingegneria del software vada a specializzarsi nei contesti specifici di sviluppo, è possibile far riferimento all'ampiamente utilizzato sviluppo software Object-Oriented, che specializza un *tradizionale* ciclo di sviluppo software in sei attività di sviluppo [2]:

- *Raccolta e Analisi dei requisiti*, grazie alle quali gli ingegneri del software, analizzano il problema con il cliente al fine di definire, i confini del dominio applicativo;

- *System Design*, fase in cui gli ingegneri del software analizzano il problema e lo dividono in piccoli pezzi, al fine di selezionare strategie generali di design per il problema proposto;
- l'*Object Design*, fase in cui vengono selezionate soluzioni dettagliate per ogni *sotto-problema*;
- l'*Implementazione del sistema*, che corrisponde solo ad uno degli ultimi passi del ciclo di sviluppo, durante la quale gli sviluppatori *traducono* la soluzione al problema in codice sorgente;
- Il *Testing*, fase in cui gli sviluppatori cercano differenze tra il sistema implementato e i suoi modelli sulla base di ben definiti campioni di dati di input.

Come osservato, l'ingegneria del software, offre anche tutta una serie di attività gestionali che interessano ed influenzano un canonico progetto software. Le attività di gestione si focalizzano sulla pianificazione di un progetto software, il monitoraggio del suo status di avanzamento, il tracciamento dei suoi cambiamenti e la coordinazione delle risorse, al fine di consegnare un prodotto di alta qualità in relazione a quelli che sono i vincoli a cui esso è soggetto, come ad esempio quelli di tempo e di costo [2].

Altra branca dell'ingegneria del software, che da attività finale di un canonico modello ciclo di vita del software è diventa un processo a se stante che caratterizza l'evoluzione del prodotto per tutta la sua durata di vita è la *manutenzione*, macro-processo che include tutte quelle attività che occorrono dopo la consegna del sistema al cliente. La manutenzione è un aspetto complesso, ma sempre più necessario al fine di garantire il successo dei sistemi software, sempre più orientati al cambiamento [2]. Basti pensare che già nel 1974, uno dei più noti studiosi del campo Henry Lehman, osservava l'importanza dell'evoluzione e della manutenzione, con la formulazione delle prime leggi sull'evoluzione del software. Le leggi di Lehman, sono ancora oggi dei principi cardine degli studi sull'evoluzione del software, e se si pensa anche solo alle prime due, si intuisce l'importanza del processo di manutenzione [4] :

- *Prima Legge di Lehman*: I programmi di tipo evolutivo, devono necessariamente essere adattati *al cambiamento*, altrimenti essi diventano progressivamente meno soddisfacenti;

- *Seconda Legge di Lehman*: Così come un programma (sistema software) evolve, così la sua complessità aumenta, a meno che del lavoro non venga fatto al fine di manutenerla e ridurla.

Per concludere, in maniera opportuna l'introduzione all'ingegneria del software è doveroso far riferimento alle metodologie agili e allo sviluppo incrementale, che nel corso degli ultimi decenni stanno prendendo sempre più piede nel contesto ingegneristico. Secondo le più note fonti presenti in letteratura, questa tipologia di approcci e metodologie, sono caratterizzati da alcuni principi innovativi che si discostano quasi del tutto dall'idea di ciclo di sviluppo tradizionale. Ian Sommerville, nel suo volume *Software Engineering*, riassume il core dei processi agili, facendo riferimento alle seguenti caratteristiche[5]:

- Il rilascio di nuove release dei sistemi caratterizzate da piccoli cambiamenti ogni due o tre settimane circa;
- Il coinvolgimento continuo dei clienti al fine di ottenere rapidi feedback e tracciare i continui cambiamenti;
- La riduzione della documentazione, *al minimo indispensabile*, preferendo l'uso di comunicazioni informali, piuttosto che meeting formali con documenti scritti;

2.2 Intelligenza Artificiale e Machine Learining

L'Intelligenza Artificiale ed in particolare i moduli di Machine Learning, stanno diventando sempre di più una parte fondamentale delle applicazioni commerciali e dei progetti di ricerca nell'ambito IT. In particolare si osserva come i moduli addestrati di maggior successo, siano realizzate a partire dalla generalizzazione di esempi noti (basi di conoscenza), sulle quali i moduli stessi vengono addestrati, al fine di produrre, sulla base di dati di input forniti, un output desiderato senza che l'utente umano dia informazioni aggiuntive. La branca del Machine Learning, nella quale vengono racchiusi tutti gli algoritmi che si basano su tali tuple di *input-output* viene definita come **Apprendimento Supervisionato** [6]. Dalla stessa fonte possiamo osservare come esempi di apprendimento supervisionato di interesse possano essere:

- L'identificazione di *topix* da un blog o un sito internet;
- l'identificazione di pattern di accesso anomali in un sito web;

- la suddivisione dei clienti di uno shop per preferenze similari.

Soprattutto negli ultimi anni, si ci sta accorgendo che l'applicazione combinata di discipline come la statistica, la teoria dell'informazione e il machine learning, stanno portando alla creazione di una scienza sempre più solida, con una ferma base matematica, e a tool sempre più potenti. In particolare, se si fa riferimento al learning supervisionato tecniche e algoritmi come alberi di decisione, regressione lineare, regressione logistica, clustering, sono alcune delle tecniche più utilizzate per la progettazione di componenti AI, per innumerevoli campi applicativi, ed in particolare quello che ne risulta dall'applicazione di una generica tecnica di learning supervisionato, su un dataset di addestramento, è definito come processo di classificazione [7].

Per capire praticamente la problematica di classificazione in ambito machine learning, si può pensare di far riferimento al comune esempio di identificazione delle mail di spam, tecnica automatica di filtering basata su di grandi data set ampiamente utilizzata dai gestori di mailing. In pratica, sulla base dei dati (strutturali, storici e similari) a disposizione di mail già contrassegnate o meno come "SPAM" è possibile addestrare un modello, al fine di classificare una nuova mail che un generico utente riceve come "SPAM MAIL" oppure "NON SPAM MAIL". Volendo dare quindi una definizione semi formale: il problema di classificazione del Machine Learning consiste nell'*individuare una* ***categoria*** *(ad esempio MAIL di SPAM oppure MAIL NON DI SPAM) per una* ***nuova osservazione*** *(e.g. una nuova mail ricevuta), sulla base di precedenti* ***dati di addestramento*** *contenenti informazioni già classificati secondo* ***categorie note*** *(ad esempio archivio di mail già classificate come SPAM oppure NON SPAM a disposizione del modulo addestrato*".

Questa problematica è particolarmente attenzionata dalla ricerca, dato che è molto complesso generare classificatori che non soffrano di alcuni problemi noti in letteratura, come ad esempio, la stretta dipendenza dal dataset di partenza [7], il così detto fenomeno del *garbage-in,garbage-out*, cioè la presenza di Bias nei dataset di addestramento, che poi tenderanno a riflettersi all'interno delle predizione dei moduli di previsione stessi [8]. Volendo porre un esempio pratico, nell'ambito dell'etica del software, è facile che un classificatore addestrato con un dataset, non immune a dei bias su specifici attributi sensibili (quali razza o sesso), possa essere addestrato in modo tale da effettuare predizioni imparziali, spesso rivolte a favore degli individui del campione di addestramento dei gruppi di maggioranza (ovvero quegli individui che posseggono il valore più ricorrente dell'attributo sensibile) [8] .

2.2.1 Algoritmi di ricerca e Algoritmi Genetici

Il machine learning è sicuramente uno degli aspetti più caratterizzanti dell'intelligenza artificiale, al fine di fornirne una panoramica più ampia è interessare considerare anche gli algoritmi di ricerca. Essi per definizione sono usati per restituire informazioni memorizzate all'interno di strutture dati o ricercare soluzioni in un complesso spazio di ricerca formalmente definito sulla base del dominio del problema, il tipo di informazioni gestite dagli algoritmi di ricerca possono essere formate da valori discreti e continui. Di algoritmi di ricerca ne esistono in letteratura di vari tipi, basati su vari approcci: algoritmi basati sull'uso di una funzione di costo (e.g. depth-first, bread-first e cost-uniform first...), basati sull'utilizzo di euristiche (e.g. A* greedy best-first), algoritmi di ricerca locale (e.g. hill-climbing search) ecc., ma tra i più usati in ambito di ricerca ci sono sicuramente gli algoritmi di tipo evolutivo ed in particolare quelli di tipo genetico. Gli algoritmi genetici, sono algoritmi basati sui principi della selezione naturale e della genetica, introdotti negli anni '70 da J.Holland e ispirati alle teorie dell'evoluzione degli esseri viventi [9]. Gli algoritmi genetici astraggono lo spazio del problema come una popolazione di individui, e provano ad esplorare le caratteristiche degli individui, "producendone" di nuovi in maniera iterativa. I GA evolvono la popolazione da individui iniziali (spesso generati casualmente) in individui di alta qualità, laddove ogni individuo rappresenta una soluzione al problema di interesse codificata in stringa. La qualità di ogni individuo è misurata da una funzione matematica detta funzione di fitness che è formulata in modo tale da valutare in maniera quantitativa la bontà di un individuo, a seconda di alcune caratteristiche qualitative definite dallo sviluppatore. A seconda del problema la funzione di fitness, può essere di massimizzazione, di minimizzazione, a singolo obiettivo o multi obiettivo, e spesso rappresenta l'ostacolo più grande nella progettazione di un algoritmo genetico.

Durante ogni generazione, tre operatori di base della genetica sono applicati in sequenza con una certa probabilità: selezione, crossover e mutazione [9] . I passaggi base di un algoritmo di ricerca genetico sono:

1. Generazione randomica di una popolazione di n individui (rappresentanti di soluzioni randomiche al problema);
2. Valutazione di ogni individui tramite la funzione di fitness;
3. Selezione di due individui per generarne di nuovi (nuova generazione), le tecniche

di selezione possono essere diverse, tra le più famose ci sono: la roulette wheel e l'approccio a torneo;

4. Con una certa probabilità, viene applicata un operazione di cross over al fine di mischiare singole parti degli individui selezionati per generarne di nuovi (anche qui ne esistono vari tipi). Se il crossover non viene applicato, in sostituzione vengono copiati "i genitori";

5. Con una certa probabilità ai nuovi individui vengono applicate operazioni di mutazione (ovvero vengono cambiati uno o più dati della stringa casualmente);

6. Gli individui generati, vengono aggiunti alla popolazione, al fine di far ripartire l'algoritmo;

7. Se uno degli individui generati soddisfa le condizioni di accettazione e di arresto dell'algoritmo, allora si ritorna la soluzione migliore trovata fino a quel momento;

8. Altrimenti l'algoritmo riparte dal passo 2;

2.3 Ingegneria del Software nell'Intelligenza artificiale

Si può osservare come negli ultimi decenni, l'intelligenza artificiale e l'ingegneria del software, si siano evolute separatamente, oggi giorno, si osserva però come la ricerca attuale, stia portando alla specifica e alla costituzione di nuovi studi e approcci allo sviluppo di soluzioni AI-Intensive, che tengano proprio conto dell'intersezione che c'è tra l'ingegneria del software e l'intelligenza artificiale [10]. L'Intelligenza artificiale odierna fa riferimento a sistemi che sulla base degli input che ricevono in considerazione devono assumere rischi, fare predizioni o assumere comportamenti in risposta a specifici problemi [11], in dettaglio, nell'era dei computer dalle elevate prestazioni e dei Big Data, molte soluzioni AI-Intensive, sono sviluppate in risposta ai bisogni che la quotidianità sociale necessita, ma come noto in letteratura, molti sistemi software di grandi dimensioni, non sono privi di bug, ed in particolare i sistemi di Intelligenza Artificiale e di Machine Learning non fanno eccezione. Per questa tipologia di sistemi, bug di progettazione o addestramento, possono essere causa di crush di sistema, output errati fino all'esecuzione troppo lenta che non rende possibile l'utilizzo di tali soluzioni nell'ambiente di lavoro [12]. In casi critici gli errori dei sistemi intelligenti, possono addirittura portare alla morte di chi anche involontariamente interagisce con essi, il caso più noto è quello della ciclista Elaine Herzberg, morta investita da un'auto

con pilota automatico, per errore di valutazione del modulo di guida [11]. Quindi come è possibile produrre soluzioni AI Intensive per il mondo reale, che tengano conto di tali problematiche? L'applicazione dell'ingegneria del software cerca in qualche modo di dare risposta a questa tipologia di problematica, in dettaglio, si cerca di includere metodi di Requirement Engineering, Design Engineering, Code Engineering e Project Management, che possono essere di supporto, per lo sviluppo di Sistemi Ai-Intensive efficienti. La ricerca stessa ne fa utilizzo, ed infatti molti ambiti di studio sono nati dall'intersezione tra SE e AI, in particolare vale la pena citare l'Agent Oriented Software Engineering oppure l'Ambient Intelligence [13]. L'Agent Oriented Software Engineering, si concentra sullo sviluppo di soluzioni che siano, intelligenti, agili e pro-attive. In dettaglio il suo scopo principale è quello di sviluppare soluzioni AI-intensive tramite dei riadattamenti stessi delle tecniche di Ingegneria del Software, per lo sviluppo di Agenti di piccoli e grandi dimensioni. Nel dettaglio risultano essere di rilievo l' *Agent UML*, per la progettazione dei moduli agente, oppure metodi di specifica e valutazione formali dei sistemi, i quali hanno lo scopo intrinseco di validare gli obiettivi, i comportamenti di un singolo agente, e soprattutto le interazioni nei sistemi multi agente [10]. l'Ambient Intelligence invece si pone lo scopo di progettare ambienti di addestramento (secondo tecniche specifiche dell'ingegneria del software) che siano sensibili ed adattabili agli stimoli esterni e in conseguenza, sistemi reattivi che siano informati circa i bisogni, le abitudini e le emozioni degli utenti per supportare il loro lavoro quotidiano[10].

Qualsiasi branca o studio di ricerca a cui si voglia far riferimento (come gli esempi riportati) definita nella così detta "intersezione tra intelligenza artificiale e ingegneria del software", non può prescindere dall'analizzare quelli che sono i requisiti non funzionali di qualità che una soluzione AI-Intensive deve rispettare. In particolare in letteratura [14], si può osservare come, ad esempio, il Machine Learning sia soggetto a specifici vincoli di qualità quali:

- *Accuracy and Performances*, come l'output di un agente risulta "corretto" se paragonato alla realtà;
- *Fairness*, Requisito che si pone l'obiettivo di rendere gli algoritmi di ML più **imparziali** e indipendenti da bias di dati;
- *Transparency*, ovvero la capacità di dimostrare come i risultati elaborati da un modulo intelligente siano affidabili e trasparenti, ricostruendo le fonti di partenza;

- *Security and Privacy*, *Testability* e *Reliability* del modulo addestrato.

Per progettare, realizzare e controllare questi aspetti qualitativi essenziali per un modulo di intelligenza artificiale, la ricerca è enormemente incentrata nello studio di questi aspetti. In particolare, si nota come questi requisiti non funzionali, siano particolarmente attenzionati dall'ingegneria dei requisiti, branca dell'ingegneria del software che si incentra nella specifica, l'analisi, la verifica e la validazione dei requisiti di un sistema software [15], la cui ricerca ricerca sta incentrando buona parte dell'effort nello studio di aspetti quali fairness, privacy, sostenibilità e modificabilità anche per tecniche di Machine Learning e per lo sviluppo di soluzioni AI-Intensive in generale [14]. In particolare, la letteratura afferma come l'industria dell'intelligenza artificiale è particolarmente incentrata nello sviluppo di soluzioni di Machine Learning e basate sugli approcci Data Driven, ed una delle principali aree di interesse per lo sviluppo di questo tipo di soluzione è il settore dell'Healthcare, particolarmente caratterizzato dalla mancanza di standard di progettazione e dalla continua evoluzione dei dati a disposizione[15], per far fronte a questa tipologia di problematiche, l'ingegneria del software ed in particolare l'ingegneria dei requisiti, pongono l'accento su nuove metodologie e tecniche che toccano vari processi di un sistema AI-Intensive: la specifica e l'analisi dei suoi requisiti, la validazione del modello (e quindi delle sue specifiche), la documentazione e il management dell'intero processo di sviluppo dei requisiti formulati. Le sfide principali che la ricerca ha davanti in quella che è stata definita *intersezione* tra intelligenza artificiale e ingegneria del software sono:

- *Lo Skill Gap*: la necessità di creare lo giusto spirito di collaborazione aziendale tra Data Scientist e Ingegneri del Software;
- *Il Data Gap*: Ovvero la necessità di rendere disponibili (big) dataset necessari alla realizzazione di soluzioni AI complesse;
- *L'Engeeniring Gap*, ovvero la necessità di creare prototipi generalizzabili dei sistemi AI, con il giusto supporto all'intero ciclo di vita della Soluzione AI-Intensive;

2.3.1 Machine Learning Life Cicle e ML Pipeline

Uno dei contributi generali che l'ingegneria del software cerca di dare all'intelligenza artificiale è proprio quello di sistematizzare la progettazione e lo sviluppo dei suoi modelli. In tal senso lo sviluppo di una soluzione AI-Intensive, così come un generico progetto di Machine Learning, può essere sistematizzato definendo opportuni processi ingegneristici e

di conseguenza un vero e proprio ciclo di vita della soluzione che si sta progettando. Ogni progetto AI-Intensive, quindi può avere uno specifico ciclo di vita, e concentrandosi nel del machine learning, Burkov e Andriy affermano che un progetto ML-Intensive è caratterizzato inizialmente dalla comprensione e dall'analisi dello scope applicativo e degli obiettivi di business il modello [16], successivamente il primo passo che porta alla nascita di un modello di machine learning è capire se il modello stesso ha dei precisi obiettivi. Gli obiettivi di un modello ML-Intensive sono ovviamente diversi da quelli di business e dettano le basi di progettazione del modello stesso, un generale obiettivo di un modello ML, deve essere formalizzato in modo tale da tener conto [16]:

1. Di cosa il modello riceve come input;
2. Di cosa genera come output;
3. Dei criteri di accettabilità (o inaccettabilità) del comportamento del modello;

Il ciclo di vita di un modello di machine learning può essere sistematizzato tramite i seguenti passaggi [16]:

1. Definizione degli obiettivi;
2. Collezione e preparazione dei dati;
3. Ingegnerizzazione delle feature;
4. Training del modello;
5. Evoluzione del Modello;
6. Deployment del Modello;
7. Fase operativa del modello e monitoraggio;
8. Manutenzione del modello.

Approcci più innovativi allo sviluppo ML-Intensive, sottolineano come aspetti di monitoraggio, architecturing e monitoring continuo, siano essenziali per far evolvere un modello di pari passo con il suo ambiente di utilizzo, uno degli approcci innovativi che cerca di rispondere a queste esigenze è la specializzazione ML di DevOps, che appunto prende il nome di MLOps.

I passaggi del ciclo di vita di vita di un modello ML-Intensive, possono essere anche organizzati e automatizzati in modo tale da configurare una vera e propria **pipeline di machine learning**, che viene definita come una sequenza di operazioni su un dataset che vanno dal suo stato iniziale, fino al rilascio del modello e la sua evoluzione. Una pipeline può includere tra i passaggi di preparazione dei dati, imputation dei dati mancanti, estrazione delle feature, data augmentation e model training[16]. In pratica, una delle innovazioni più forti portati dalla formalizzazione di una pipeline di machine learning, è che quando un modello di machine learning è deployato in produzione, in realtà è deployata l'intera pipeline[16] (rispondendo in maniera *automatizzata* alle necessità evolutive del modello. Da notare che una pipeline di machine learning è generalmente ottimizzata quando i suoi hyperparameters di configurazione sono ottimizzati (Hyperparameters tuning) [16].

2.3.2 Piattaforme per Machine Learning Pipelines e MLOps

Dare la definizione di Machine Learning Pipeline, fa capire come lo sviluppo e il deploying di applicazioni ML-Intensive, è un processo che va ben oltre i dati collezionati e i modelli addestrati e fare predizioni. Come osservato da Zhou et al. connettere queste parti ignorando la fase di manutenzione del modello, costituisce un enorme debito tecnico [17]. Costruire un workflow dal data pre-processing fino al porre il modello in run nel contesto di utilizzo, può facilmente essere un processo dispendioso in termini di tempo e soprattutto non privo di errori, tra l'altro una produzione efficiente e affidabile è fondamentalmente critico per molti contesti reali, si pensi a casi d'uso quali la guida automatica oppure l'health care[17].

In risposta a queste esigenze critiche, sono nate piattaforme di machine learning, che in maniera embedded forniscono un ciclo di vita manageriale end-to-end per applicazioni ML-Intensive, core primario di queste applicazioni cercano di accorpare sistemi stand-alone che in maniera specificano si assumono responsabilità di task quali: data-preprocessing, model training, model evolution e messa in servizio. Goal primario di queste piattaforme è quello fornire una soluzione generica per molteplici casi d'uso di sviluppo, collegando e configurando componenti indipendenti per specifiche problematiche. Con questa tipologia di configurazione, un generico workflow ML può essere orchestrato e convertito in una vera e propria Pipeline di Machine Learning, la cui esecuzione è supportata da queste piattaforme [17]. Oltretutto il livello di affidabilità, scalabilità, abilità di continuous training, fornito da questi strumenti, offre la possibilità di adattare e ed evolvere i dati oppure incrementare i

cambiamenti frequentemente [17].

Con la creazione di Tools che permettono di ingegnerizzare una pipeline di machine learning, hanno portato nell'ambito dell'ingegneria del software alla nascita di vere e proprie culture e pratiche di sviluppo specifiche. la più nota nel mondo dell'intelligenza artificiale è sicuramente *MLOps*, che specializza nell'ambito AI, la più generica cultura di sviluppo software *DevOps*.

Per comprendere bene un processo di sviluppo ML-intensive, basato sulla filosofia ML-Ops, è necessario comprendere innanzitutto, quali sono i processi generali che entrano in gioco durante il ciclo di vita di una soluzione ML-Intensive. Innanzitutto è necessario distinguere quelli che sono i processi di sviluppo, che racchiudono [18]:

- Raccolta e manipolazione dei dati grezzi;
- Analisi e processing dei dati per il training;
- Costruzione, training e testing del modello;
- Overfitting e tuning del modello;
- Validazione del modello.

da quelli che vengono invece definiti processi derivanti dall'uso del modello nell'ambiente di utilizzo, chiamati appunto processi operazionali eseguiti da figure professionali dedicate, in particolare possono essere definiti processi operazionali per una soluzione ML-Intensive [18]:

- Deployment del modello;
- Monitoring del modello;
- Formulazione di statistiche di reporting;
- Feedback retrival al team di manutenzione;

DevOps, o development operations, è una cultura più generale di sviluppo ingegneristica, che si riferisce a set di pratiche che combinano i generici processi di sviluppo software, con quelli operazionali, in modo tale da creare un set comune di pratiche che assicurino un'accelerazione dei tempi di sviluppo, oltre che una rapida risposta alle necessità di cambiamento data dal monitoraggio continuo del tool rilasciato, incorporando quindi in maniera naturale

quello che è il concetto di continuous delivery[18]. Con l'utilizzo di un tipico workflow DevOps, si osserva come i costi totali di manutenzione si riducono in maniera considerevole dato che la manutenzione stessa è integrata in maniera nativa all'interno dell'efficiente meta-processo [18]. Similarmente MLOps adotta i principi di DevOps e li specializza per i modelli di machine learning piuttosto che per un generico prodotto software. In particolare l'approccio ML-Ops, crea un meta-modello di sviluppo iterativo, che integra il ciclo di sviluppo, di responsabilità dei data scientists e dei Machine Learning engineers, con i processi operazionali del team addetto, al fine di assicurare continuous delivery di modelli di machine learning dalle alte prestazioni[18].

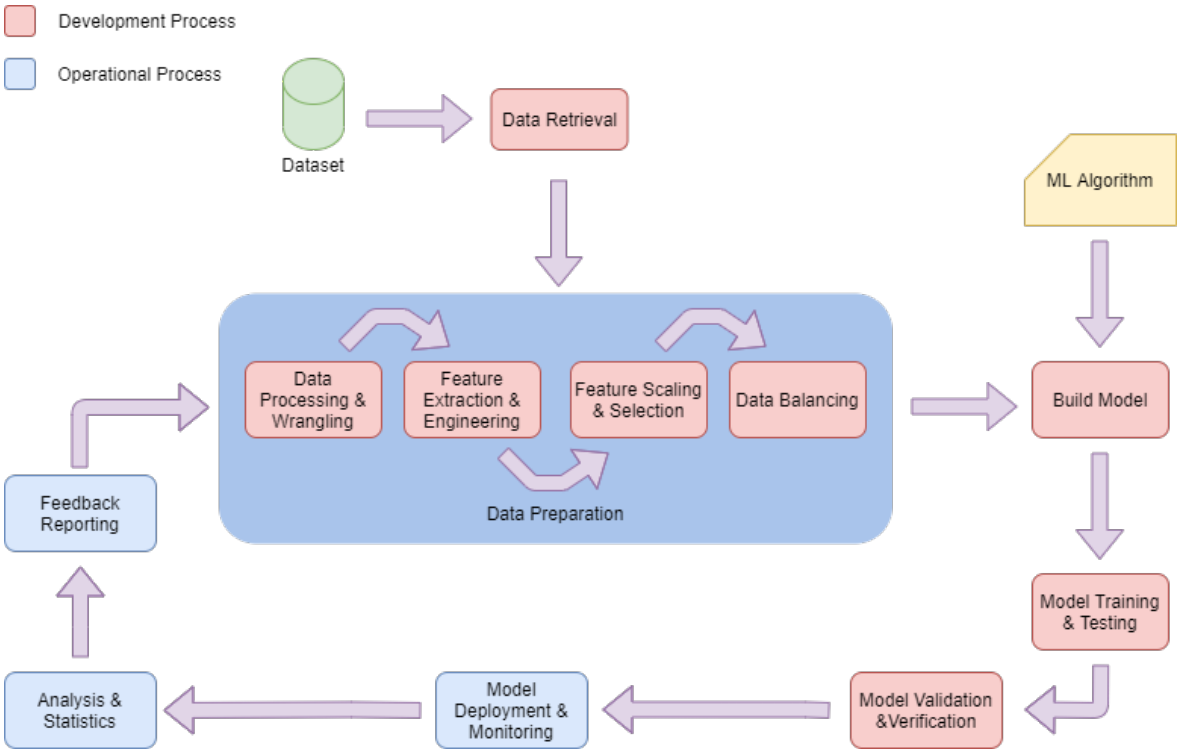

Figura 2.1: Esempio generico di pipeline di machine learning basato su MLOps

CAPITOLO 3

Stato dell'arte

3.1 Definizioni e Metriche di Software Fairness

Dopo aver introdotto il background in cui il lavoro di tesi si si colloca, è senz'altro necessario introdurre quello che è l'aspetto principale che caratterizzerà il lavoro illustrato nei successivi capitoli, ovvero la Software Fairness e il codice eticamente corretto. Horkoff introduce alla fairness come un requisito non funzionale dei moduli di machine learning, per il quale le attività di ricerca attuale sta investendo molto. In particolare si cerca di rendere gli algoritmi di machine learning più imparziali, non solo cercando di rimuovere features sensibili (come razza o sesso), ma cercando soprattutto di definire e implementare soluzioni AI-Intensive che tengano conto del livello di fairness richiesto dal dominio di applicazione[14].

Implementare un algoritmo eticamente corretto, significa definire formalmente cosa si intende per fairness, e cercare in conseguenza di misurare come ciò viene implementato[14]. Dato che concetto di Fairness è relativamente nuovo per la comunità scientifica, e soprattutto requisiti sociali e principi legali spesso ne evidenziano differenti caratteristiche, sono emerse differenti definizioni del concetto e un numero speculare di metriche per misurarne il livello nei sistemi software[8]. Prima di cercare di definire formalmente il concetto però è necessario cercare di capire in che maniera un sistema software mostra dei comportamenti scorretti e nello specifico , e come la comunità scientifica si pone rispetto ad esso.

Recenti studi, inerenti gli aspetti di qualità e il successo del processo ingegneristico del software, hanno dimostrato che i sistemi software presentano un nuovo tipo di vulnerabilità, correlate appunti alla loro abilità di operare in maniera imparziale (fair) e priva di pregiudizi. Da dove nasce, quindi il problema della software fairness? [1] come espresso in letteratura, il livello di fairness è strettamente correlato al concetto di Bias (pregiudizi algoritmici) che un sistema software ha al suo interno, questi comportamenti erronei possono emergere da vari aspetti, soprattutto appresi dai dati di addestramento (per quanto concerne le soluzioni AI-Intensive), ma anche per specifiche di requisiti incomplete, design povero, bug di implementazioni o interazioni errate tra componenti. Esistono differenti esempi interessanti di come fair bias nei sistemi software abbiano portato a spiacevoli inconvenienti, tra i più famosi vale la pena citare:

- *Software di Recruiting*: Gli specialisti di Machine Learning di Amazon, hanno reso noto che nel 2015, il loro nuovo tool di recruiting per lo sviluppo o altre posizioni tecniche, non offriva opportunità lavorative in maniera imparziale rispetto al sesso dei candidati, questo perché tale sistema era addestrato con dei dati di un periodi di 10 anni antecedente al 2015, per il quale la prevalenza degli impiegati tecnici dell'azienda è stata maschile, il gender e altri fattori (quali razza, oppure lo stesso linguaggio che gli impiegati tecnici maschili hanno adottato negli anni), sono stati classificati come feature sensibili che hanno addirittura portato Amazon, a chiudere il progetto, secondo quanto stabilito dalla loro versione ufficiale [19];

- *Healthcare*: Gli ospedali statunitensi per anni hanno utilizzato un software di predizione, per le cure mediche, che nel tempo è arrivato a "preferire" i pazienti di razza bianca, rispetto a quelli di colore. Tale comportamento "unfair" è da attribuire, purtroppo ad un bias di dati che considerava più "conveniente" la popolazione bianca, dato che quest'ultima era soggetta a spese mediche maggiori rispetto a quelle di colore, quindi "secondo il tool" preferendo tali individui su un campione di più di 200 milioni di abitanti, ci sarebbero stati maggiori guadagni per gli ospedali pubblici americani. Tale assunzione però si è convertita nella pratica nel considerare le persone di colore, maggiormente in salute rispetto quelle di carnagione chiara [20];

- *Crimine*: Nel 2014, si è osservato come un tool (denominato Correctional Offender Management Profiling for Alternative Sanctions, meglio conosciuto con l'acronimo di COMPAS), di predizione di responsabili di futuri crimini tra i più usati in Florida tra

il 203 e il 2014, il cui funzionamento si basa sul confronto di analisi facciali partendo da un dataset di immagini di fotografie di criminali condannati, avesse la tendenza a giudicare le persone di colore come più più predisposte a crimini violenti quali, la causa di questa tematica è ancora oggi molto discussa, ma si ritiene che la causa principale, sia nuovamente nei dati con cui il modello veniva addestrato, nel quale erano presenti feature sensibili, quali razza o sesso [21];

- *Traduzioni Automatiche*: Si è riscontrato come Google Traduttore, il più popolare motore di traduzione al mondo, mostrava un bias legato al sesso. in particolare, se si traduce dall'inglese al turco la frase "She is an engineer, He is a nurse", ne risulta un inversinone di soggetti: "He is an engineer, She is a nurse", quasi ad indicare come le professioni tecniche, come un generico Ingegnere, in Turchia sia automaticamente associato al sesso [21];

- *Generazione automatica di sottotitoli*:Su Youtube, se si seleziona la traduzione automatica, si osserva come effettivamente la traduzione del video risulti essere maggiormente accurata con voce maschile, rispetto a voce femminile, inoltre si osserva come che per lingue come l'inglese o l'arabo, ci siano delle differenze qualitative circa il risultato della traduzione, per alcuni specifici dialetti [22].

Dagli esempi citati si osserva, quindi, come ricercatori e ingegneri del software producano sempre di più software di qualità, in risposta alla necessità di prendere delle decisioni eticamente corrette, che riguardano sempre di più la vita umana[23]. Di conseguenza, si nota come un tool software o una soluzione AI-Intensive utilizzata su larga scala, non possa più presentare vulnerabilità che vadano ad inficiare il suo livello di Fairness. In risposta a questa nuova issiue ingegneristica, ne deriva in maniera naturale che è necessario come applicare definire formalmente il concetto di Software Fairness e conseguenti processi standard di misurazione. Negli ultimi anni questa problematica, ha attirato l'attenzione di ricercatori nell'ambito dell'intelligenza artificiale, l'ingegneria del software e la comunità legislativa, con più di venti differenti notazioni di Software Fairness proposte e ovviamente quello che ne deriva è che non è che non è possibile darne un'unica definizione specifica, ma è necessario specializzare il concetto in riferimento ad una specifica problematica analizzata [24]. Ponendo l'attenzione al problema di Classificazione del Machine Learning, già definito nel precedente paragrafo di introduzione all'intelligenza artificiale, è possibile dare numerose definizioni di Fairness per un generico classificatore ML, sulla base di alcuni strumenti ampiamente

utilizzati nell'ambito dell'Intelligenza Artificiale, ovvero: Le misure statistiche di validazione, le misure di similarità e distanza e il Casual Reasoning [24].

3.1.1 Definizioni di Fairness basate su Metriche Statistiche

Sahil Verma e Julia Rubin, introducono a questa tipologia di definizioni basate su metriche statistiche fornendo alcuni concetti di base [24]:

- Attributi sensibili o protetti: attributo di un dato dataset, sul quale si sta effettuando classificazione, per il quale il generico modulo AI di classificazione potrebbe produrre discriminazioni (come ad esempio gender o razza);
- Diretta conseguenza degli attributi sensibili possono essere i Gruppi Protetti, gruppi di individui che assumono un valore dell'attributo sensibile che potrebbe essere soggetto a discriminazione;
- Il valore reale di classificazione: ovvero il valore (categoria di appartenenza) assegnato un individuo del dataset assume sulla base base delle sue feature di riferimento;
- La probabilità di predizione [0-1]: ovvero la probabilità condizionata che un individuo appartenga ad una data categoria, sulla base dei dati che lo caratterizzano, i classificatori, per stimare queste probabilità fanno riferimento in generale a tutte le feature del dataset, e in casi particolari possono anche riferirsi ad attributi sensibili, per particolari scelte di implementazione;
- La decisione predetta [0-1]: ovvero la categoria di appartenenza dell'individuo determinata dal classificatore, sulla base del dataset.

Le metodologie statistiche di validazione di un classificatore, ed in particolare i classificatori binari (con due singole categorie di classificazione, per convenzione una positiva e una negativa), fanno utilizzo di questi concetti, un determinato individuo può risultare:

- *Reale Positivo - True Positive* (TP), se il valore reale di predizione e la decisione predetta dal corrispondono entrambi alla categoria "positiva";
- *Reale Negativo - True Negative* (TN), se il valore reale di predizione e la decisione predetta corrispondono entrambi alla categoria "negativa";
- *Falso Positivo - False Positive* (FP), se la decisione predetta corrisponde alla categoria "positiva", mentre il valore reale di predizione è la categoria "negativa";

- *Falso Negativo - False Negative* (FN, se la decisione predetta corrisponde alla categoria "negativa", mentre il valore reale di predizione è la categoria "positiva".

,

Determinando questi valori per ciascuno degli individui del dataset, è possibile estrarre numerose metriche utili alla validazione di un generico classificatore binario, le più famose sono note in letteratura sono:

...la *precision*:

$$\frac{\sum TP}{\sum TP + FP}$$

...e la *recall*:

$$\frac{\sum TP}{\sum TP + FN}$$

Tali formule, con altre metriche statistiche ampiamente riconosciute, sono spiegate nel dettaglio, nel paper di riferimento[24], vale la pena osservare che tali metriche possono essere generalizzate anche nel caso si stia validando un classificatore con N categorie.

Questa piccola introduzione ai concetti basilari di classificazione, permette quindi di introdurre a qualche definizione di Fairness tra le più utilizzate nell'ambito della classificazione, tra le più importanti e utilizzate, vale la pena citare (N.B. Le definizioni riportate prendono come dominio di applicazione i classificatori binari, così come gli attributi sensibili con due possibili valori)[24]:

1. *Group Fairness - Statistical Parity*: Un classificatore, soddisfa questa definizione di fairness, se individui di un gruppo protetto (G1), definito per un dato attributo sensibile (g), possano essere assegnati dal classificatore alla categoria positiva (che quindi il valore predetto d sia pari ad 1) , con la stessa probabilità degli individui del gruppo non protetto (G2), per il quale l'attributo sensibile assume valore non discriminante:

   $$P(d = 1|g = X1) = P(d = 1|g = X2)$$

   dove X1 e X2 sono rispettivamente il valore discriminate e non discriminate dell'attributo sensibile g;

2. *Predictive Parity - Outcome Test*: Un classificatore soddisfa questa definizione, se entrambi i gruppi, protetto e non protetto, hanno stessa probabilità di assumere valore reale di

classificazione (Y), pari ad 1 (categoria positiva), data decisione predetta pari ad 1;

$$P(Y = 1|d = 1\&g = X1) = P(Y = 1|d = 1\&g = X2)$$

3. *False Positive Error Rate Balance*: Un classificatore soddisfa questa definizione, se, la probabilità di essere classificati come appartenenti alla categoria positiva, (decisione predetta d = 1), pur appartenendo in realtà alla categoria negativa (valore reale di classificazione Y = 1), sia equivalente sia per individui appartenenti al gruppo protetto che per quelli appartenenti al gruppo non protetto.

$$P(d = 1|Y = 0\&g = X1) = P(d = 1|Y = 0\&g = X2)$$

Queste ed altre definizioni statistiche del concetto di Software Fairness, vengono illustrate più in dettaglio nel paper di riferimento [8], anche in termini di metriche di validazione statistica quali Precision e Recall. Quello che però è importante sottolineare è l'enorme dinamicità di queste definizioni statistiche, come si nota come gli esempi riportati (come altre definizioni statistiche di Fairness presenti in letteratura), modellano concetti di probabilità differenti, ognuno avente un significato semantico ben preciso, e per avere una buona idea del livello statistico di Fairness di un classificatore sotto analisi, è necessario identificare bene quali metriche tenere in considerazione. Come è buona pratica di questi studi statistici può essere anche opportuno considerare e incrociare i risultati più metriche per formalizzare considerazioni più attendibili.

3.1.2 Definizioni di Fairness basate su Metriche di Similarità

Continuando questo excursus, nell'analisi delle definizioni di Software Fairness, nell'ambito dei classificatori ML, è possibile è possibile notare, come le definizioni statistiche, ignorino largamente tutti gli attributi di classificazione di un soggetto (sia X l'insieme degli attributi non sensibili di un generico individuo per cui si vuole effettuare classificazione), a meno di quelli sensibili G. Ad esempio, la Statistical Parity può determinare un classificatore come *Fair*, senza tener conto delle evidenze dei valori sensibili (G), tralasciando il valore di tutte le altre features (X) del dataset. Quello che ne evince è che tali misure, per quanto di largo utilizzo, possono definire unfair un classificatore, senza però tenere conto di altri vincoli di implementazione[24]. Per venire incontro a questa problematica, il paper di riferimento, propone altre definizioni che non fanno utilizzo del concetto di probabilità, ma osservano

appunto la similarità statistica tra i singoli individui, tenendo anche conto degli attributi sensibili[24].

1. *Casual Discrimination*: Un classificatore soddisfa questa definizione, se produce lo stesso risultato di classificazione (d - decisione predetta), per ogni coppia di individui con i medesimi attributi X (non sensibili): dati due individui m ed f, tale definizione è rispettata se rispetta la seguente implicazione;

$$X_m = X_f \;\&\&\; G_f != G_M \rightarrow d_m = d_f$$

2. *Fairness Through Unawareness*: un classificatore soddisfa questa definizione, se per ogni coppia di individui, con lo stesso insieme di attributi non sensibili, si ottiene la stessa decisione predetta, in altre parole, nessun attributo sensibile è coinvolto nel processo di classificazione:dati due individui I1 e I2, il classificatore rispetta questa definizione se vale la seguente implicazione;

$$X_1 = X_2 \rightarrow d_1 = d_2$$

3.1.3 Definizioni basate sul concetto di Casual Reasoning

Oltre le definizioni basate sulla statistica matematica, il paper di riferimento, riporta anche qualche definizione, basata sul concetto di Casual Reasoning: Processo di individuazione di relazioni causale (dipendenze cause/effetto tra attributi), molto utilizzato per costruire Classificatori e altri algoritmi ML [24]. L'output di un processo di Casual Reasoning per un classificatore ML, è solitamente un Casual Graph, un grafo aciclico, orientato, che connette le singole feature del dataset in relazione causa/effetto. Caratteristica interessante di un casual graph per un classificatore ML, è la presenza di un nodo terminale con solo archi entranti che rappresenta la decisione predetta dal classificatore (d). Facendo utilizzo di un Casual Graph è possibile identificare facilmente i percorsi di dipendenza (causa/effetto) che caratterizzano le scelte del classificatore corrispondente.

Il paper definisce come componenti di un Casual Graph [24], anche:

- I *Proxy Attribute*: attributi di un casual graph utilizzati per derivarne altri, che spesso sono utilizzati per ottemperare a necessarie trasformazioni matematiche all'interno di del processo di classificazione;

- I *Resolving Attribute*: attributi di un casual graph influenzati da attributi di tipo sensibile (quindi interconnessi con i medesimi), in maniera non discriminatoria;

Anche per il Casual Reasoning, il paper riporta qualche definizione di Fairness, in questo caso direttamente individuabili dalla composizione strutturale di un Casual Graph, relativo al classificatore sotto analisi [24].

1. *Conterfactual Fairness*: un classificatore rispetta questa definizione, se nessun risultato predetto d non discende in alcun modo (tramite percorsi di casualità) da un attributo identificato come sensibile;

2. *No Unresolved Discrimination*: Un classificatore rispetta questa definizione, se nel Casual Graph di riferimento, non ci sono path da attributi sensibili ai risultati predetti, a meno che nel path non siano identificabili *Resolving Attribute*;

3. *No Proxy Discrimination*: Un classificatore rispetta questa definizione, se nel Casual Graph di riferimento, non ci sono path da attributi sensibili ai risultati predetti, "bloccati" da *Proxy Attribute*, in altri termini, gli attributi sensibili non devono incidere nelle decisioni del classificatore per effetto di processi di trasformazione applicanti la strategia dei Proxy Attribute;

3.2 Software Fairness come tematica di ricerca

Autori	*Obiettivi*	*Soluzione*
* ***Paper***: Bias in Machine Learing Software: Why? How? What to Do? [23]		
Chakraborty et al.	Identificare principali cause di fair bias in un datasets.	Produzione dell'algoritmo Fair-Smoot per il Data Balancing di feature sensibili
* ***Paper***: Ignorance and Prejudice in Software Fairness [25]		
M.Zhang e Mark Harmon	Bilanciare dati e features di un dataset, al fine di elaborare un modello efficiente di Machine Learning Fair Oriented	Analisi dei dati statistica, sulla base di differenti datasets, circa il bilanciamento ottimale di feature e campioni di dati al fine di ottenere livelli di fairness ottimali secondo opportune metriche
* ***Paper***: Diversity Data Selection under Fairness Constraint [26]		
Moumoulidou at al.	Analizzare e valutare la complessità del problema della data-diversity per soluzioni ML-Intensive	Dimostrazione formale dell'NP-Completezza dell'approccio fair Max-Min Diversification e creazione di alcune approssimazioni dello stesso
** ***Paper***: Software Fairness [1]		
Yuriy Brun e Alexandra Meliou	Valutare il concetto di Software Fairness come un requisito non funzionale prioritario al pari di altri aspetti di qualità	Definizione di approcci e analisi dello status di avanzamento circa l'adozione di tecniche e metodologie specifiche per ogni fase di un canonico ciclo di vita di un prodotto software

Autori	*Obiettivi*	*Soluzione*
** ***Paper***: "Fairness Analysis" in Requirement Assignments [27]		
Finkelstein at al.	Identificare un processo standard di identificazione di specifica dei requisiti fair-oriented, in relazione con definizioni di fairness spesso contrastanti	Analisi dei requisiti correlati da fair trade-off modellata come problema di ricerca, con vincoli specifici dettati da differenti approcci matematici di Software Fairness
** ***Paper***: Fairness Testing: Testing Software for Discrimination [28]		
Yuriy Brun at al.	Definire nuovi approcci di testing fair oriented al fine di misurare se e in che modo i programmi effettuano discriminazioni	Definizione dell'approccio Themis, al fine di generare ed eseguire test suite per valutare, secondo alcune specifiche definizioni di fairness, l'impatto di feature plausibilmente discriminanti in un tool
*** Intelligenza Artificiale **Ingegneria Del Software**		

Tabella 3.1: Software Fairness Related Work

Si osserva come nell'ambito dell'intelligenza artificiale e del machine learning i fair-bias possono influenzare la creazione di dataset di addestramento, l'addestramento dei moduli di apprendimento (che di conseguenza soffriranno a loro volta di bias), e i risultati stessi di un processo di addestramento possono essere potenzialmente discriminanti per i gruppi minoritari [29]. Per tutti questi motivi, la comunità di ricerca, nell'ambito dell'intelligenza artificiale è sempre più propensa a studiare e eliminare questi bias di tipo algoritmico, al fine di produrre soluzioni AI-Intensive, e Sistemi di Machine Learning che risentano sempre di meno delle problematiche connesse alla Fairness.

Recentemente la fairness nei software di machine learning ha destato notevole interesse anche nella comunità dell'ingegneria del software [25], per esempio grandi studiosi dell'ambito, quali Yuriy Brun and Alexandra Meliou dell'università del Massachusetts, affermano che

numerose iniziative in diverse aree dell'ingegneria del software (quali: specifica dei requisiti, design, testing e verifica) necessitano di essere prese al fine di risolvere il problema. Altri studiosi invece descrivono la fairness come una vera e propria proprietà non funzionale per i sistemi di machine learning e che sostanziale effort di testing sia necessario al fine di scovare le vulnerabilità e violazioni di fairness all'interno dei moduli di machine learning.

Una panoramica completa di tutti i related work di Software Fairness presentati nel documento è riportata in tabella 3.1

3.2.1 Fairness come oggetto di studio nell'ambito AI

Bias di dati come causa di discriminazioni di un modulo AI

Rendere un modulo di machine learning, fair, significa renderlo innanzitutto indipendente dalle dipendenze correlate ai bias immessi, quindi una delle primissime attività su cui la ricerca si basa, è proprio l'individuazione di bias all'interno dei set di dati,i quali devono essere mitigati tramite intensive attività di pre-processing.

Come affermato da Chakraborty et al.[23] la principali cause dei bias, sono le decisioni prese a priori circa i dati che vengono selezionati per un modulo di Machine Learining e l'assegnazione di "label" che possono portare alla creazione di disparità di gruppo tra gli individui del dataset, esempi di "etichette" possono derivare da attributi quali sesso, razza, età, status sociale, etc (volendo porre un esempio, gli individui del dataset possono essere "etichettati" in maniera discriminante come "ricco" o "povero" in base al guadagno netto mensile). Il loro algoritmo Fair-Smoot che dati in input dataset e attributo sensibile, partiziona l'intero dataset in 4 sottogruppi (individui favoriti e privilegiati, favoriti e non privilegiati, individui non favoriti e privilegiati ed infine individui non favoriti e non privilegiati). Sinteticamente, da questa divisione iniziale, l'algoritmo va a generare nuovi *data points*, unità discrete di informazione, quindi nuove entry del dataset, per ciascuno dei sottogruppi, ad eccezione di quello che risulta averne il numero maggiore. Come risultato, tutti e 4 i sottogruppi del dataset diventeranno della stessa taglia rispetto l'attributo sensibile (la stessa del gruppo più numeroso). Lo scopo di questo algoritmo, come altri due presenti nello stato dell'arte (Fairway di Chakraborty e Oprimized pre-processing for discrimination prevention di I.Guyon) è quello di fornire bias mitigation e quindi bilanciare il dataset di partenza (con attività di pre-processing), con il compromesso di condurre a priori un analisi sui dati di partenza che non sia complessa in termini di performances (misurata in termini di Recall e F-Measure).

Statisticamente il loro algoritmo Fair-SMOTE è tra le soluzioni più promettenti effettuando data-balancing in media 200 volte più velocemente di altre soluzioni ed a seguito di studi empirici su più dataset è tra i più consigliati al fine di mitigare i bias all'interno di un dataset di addestramento.

Fairness come conseguenza delle Feature e del Training Set

Altri studi come quelli condotti da Zhang e Harmon, [25], affermano che la Fairness è naturalmente un problema specifico del dominio di utilizzo, ma che è comunque possibile generalizzare il concetto analizzando il numero di feature e l'ammontare dei dati di training. In particolare nel suddetto paper di ricerca, vengono riportati i risultati di uno studio empirico sull'impatto delle fattezze del feature set e del dataset di training quando si cerca di sviluppare *fair machine learning software*, ed in particolare viene valutate le implicazioni che questi aspetti hanno nel costruire *Fair ML Models*. Per lo studio vengono utilizzati diversi datasets presenti e noti in letteratura (quali il COMPAS score, per la valutazione di recidività nel crimine) e differenti definizioni e metriche di fairness (riconducibili a quanto illustrato nel capitolo precedente). Applicando differenti considerazioni matematiche, si arriva ad affermare che avere modello ML con un grande numero di feature, aiuta a migliorare (secondo varie definizioni matematiche) i livelli di fairness d del 38% rispetto la media e che un grande numero di dati possa provocare l'effetto contrario, ovvero una diminuzione sostanziale dei livelli di fairness.

Diversità nella selezione dei dati con vincoli di Fairness

Considerando che i dati sono generati e collezionati da tutti gli aspetti dell'attività umana, in domini come commercio, medicina e trasporti così come la misurazione scientifica, le simulazioni e il monitoring ambientale, è facile incorrere nella pratica di raggruppare i dati, in modo tale da garantire il principio di diversità il quale resta uno degli elementi molti campi applicativi dell'Intelligenza artificiale come la Summarization, la Facility Location e i sistemi di raccomandazione. Ad oggi però non sono molti gli studi che mettono a confronto la Diversificazione con il concetto di Fairness, i quali sono strettamente correlati, ma modellano concetti differenti, la prima cerca di massimizzare la dissimilarità di items in un insieme di dati, mentre la seconda cerca di raggiungere specifici livelli di rappresentazione considerando diverse categorie e gruppi.
Moumoulidou at al. [24] osservano proprio come sia importante e non banale selezionare

sotto-insiemi di di addestramento più differenti possibili (massimizzando la dissimilarità degli individui in ogni insieme), soprattutto qualora sia necessario raggiungere specifici livelli di rappresentazione di differenti categorie e gruppi, in altre parole il problema che il paper analizza è proprio quello di creare diversificazione tra gli individui di un dataset secondo specifici vincoli di Fairness (tipicamente definiti su attributi sensibili). Viene menzionato il più studiato, analizzato e frequentemente usato modello di diversificazione usato dalla comunità del Data Management, ovvero il Max-Min diversification model. Dopo aver introdotto il modello base, lo studio si concentra su una specializzazione del problema con l'introduzione di un numero k di vincoli di fairness prespecificati (per specificare i vincoli è possibile utilizzare per esempio la definizione di Statistical Parity). Quello che, infine, si osserva è che la *fair - Max-Min Diversification* sia un esempio di algoritmo NP-Completo (conclusione ricavata dimostrando prima l'NP-Completezza del problema generale), in ragion di ciò vengono mostrate delle forti approssimazioni dell'algoritmo base che garantiscono la diversità in caso di gruppi non sovrapposti (ad intersezione vuota) [26].

3.2.2 Fairness come oggetto di studio nell'ambito SE

Software Fairness come requisito non funzionale prioritario

Come già osservato più volte all'interno del capitolo Stato dell'Arte, progettare e produrre fair software, sta diventando un ambito che interessa sempre di più il dominio dell'ingegneria del software, Yuriy Brun e Alexandra Meliou, [1] osservano come la Software Fairness, debba essere un entità di "prima classe" in un tipico processo di ingegnerizzazione del software, al pari di altri aspetti non funzionali come qualità e sicurezza. In particolare, al fine di rendere immune un tool immune da discriminazioni, quindi ridurre quelli che sono i difetti intrinseci che diminuiscono i livelli di fairness, è senz'altro importante adottare buone pratiche di design e e algoritmi mirati, ma è necessario porre sullo stesso piano anche la necessità di supportare attività di "fairness testing". Al fine di misurare le discriminazioni software, è necessario identificare e riportare quelli che vengono definiti come "discrimination bugs", in modo tale da cambiare il codice o i dati che introducono tali discriminazioni. Cercando appunto di rispondere a tali problematiche, l'articolo evidenzia tutta una serie di challenge aperte in ciascuno degli aspetti chiave del ciclo di sviluppo del software [1]:

1. *Requirement and Specification*: ponendo l'accento sulla moltitudine di definizioni emerse per il concetto di fairness algoritmica, e sulla correlata difficoltà di definire un software "fair" in maniera univoca, il paper osserva come la consistenza dei requisiti e le analisi

correlate siano una una delle challenge aperte nell'ambito della Requirement Engineering quando si parla di Software Fairness, infatti, quando si considerano combinazioni di "fairness requirements" si può aver a che fare con più definizioni del concetto che possono essere contro intuitive e mutualmente esclusive, e allo stesso tempo analisi automatizzate possono identificare requisiti insoddisfabili o inconsistenti. Il risultato ultimo, infatti, è senz'altro la produzione di software soggetto a risultati inattesi e comportamenti inaspettati, per esempio si osserva, come un tool addestrato con la tecnica degli alberi di decisione derivato da un dataset con feature sensibili, fosse finito a discriminare in maniera molto forte sulla razza dei singoli individui. Al fine di evitare tali problematiche si sottolinea come l'analisi possa aiutare a comprendere come i requisiti di fairness influenzino gli altri requisiti (fairness trade-off) al fine ultimo di realizzare una specifica dei requisiti corretta;

2. *Architecture and design*: è noto come le inconsistenze tra le proprietà di design desiderate per i sistemi software siano comuni. Infatti, in generale una challenge di design aperta è proprio quella di creare tools che aiutino a modellare le architetture dei sistemi, identificando i conflitti. In particolare, nell'ambito del la software fairness, si osserva come una delle ricerche aperte, sia proprio quella di sviluppare stili di sviluppo e pattern di design per le proprietà di fairness, con l'obiettivo di trattare i trade-off dei fair design goals in maniera semi-automatica, come già viene fatto per altre specifiche non funzionali, ad esempio tramite l'ottimizzazione multi-obiettivo. Il paper inoltre osserva come per i sistemi ML-Intensive, il design di algoritmi fairness-aware possa produrre dei fair models soprattutto laddove lavorare con dati di training affetti da bias è critico. La ricerca da questo punto di vista è molto attiva, e molti algoritmi sono in sviluppo e molti framework di progettazione sono in sviluppo;

3. *Testing and Debugging*: È ormai noto come il primo metodo per assicurare la qualità del software sia il testing. Questa è la principale ragione per credere che ciò sia vero anche per la software fairness. In particolare il paper afferma come i Fairness bug siano comuni per sistemi con complessi input e output (si pensi alle forti dipendenze dalla lingua dei sistemi di Speach to Text, all'accuracy dei sistemi di riconoscimento facciale, strettamente dipendenti dalle informazioni demografiche come sesso e razza) e come questi sistemi siano la challenge più grande per la generazione di casi di test per questa tipologia di tool. Il paper evidenzia come il fairness testing richieda che i moduli vengano posti sotto test, svariate volte, e ricordando che il testing esaustivo è

inapplicabile per sistemi complessi, sottolinea l'importanza di eseguire test con input similari. Si evidenzia infatti come l'ottimizzazione di esecuzione incrementale, sulla base dei test già eseguiti con input similari, possa, potenzialmente, ridurre i tempi di esecuzione dei test e aumentare l'applicabilità del fairness testing per i grandi sistemi. Similarmente, la prioritizzazione e selezione dei casi di test possono migliorare l'efficienza dei sistemi di fairness testing. Contestualmente si specifica anche la necessità per gli sviluppatori di identificare e rimuovere le "root causes" dei bias, ciò comporta come la ricerca si stia attivando al fine di fornire strumenti di debugging appositi da mettere a disposizione degli sviluppatori;

4. *Verification*: al pari della correttezza del software, il paper evidenzia come anche la verificabilità sia un gola altamente desiderabile per la software fairness. L'esecuzione multipla dello stesso codice che può portare ad output diversi (*non determinismo*), la stretta dipendenza del concetto di fairness con l'esecutore (*la multi utenza*) e la *natura probabilistica* delle proprietà di fairness, sono tutti aspetti che riducono lo spettro di tecniche di verifica e validazione esistenti ed applicabili direttamente al problema. Quando si parla di fairness, è essenziale verificare il comportamento dei tool già durante lo sviluppo, perciò, si evidenzia come, creare ambienti di runtime e tool di debugging mirati all'identificazione dei bug o warning di fairness sia essenziale, al fine di verificare formalmente il comportamento dei tool. Il problema principale però è nuovamente, identificare modi per codificare le definizioni di fairness come proprietà verificabili di un programma, ciò è strettamente connesso alla natura intrinseca delle metriche, alcune sono di tipo probabilistico e plausibilmente verificabili, altre però sono di natura diversa (e.g. casual reasoning e metriche strutturali). Tutto ciò rende la verifica una sfida di ricerca ancora molto aperta e avvincente.

Il "problema fairness" nella specifica dei requisiti

Finkelstein at Al. [27], nel 2008 hanno introdotto il concetto di fairness nell'ambito dell'analisi e ottimizzazione dei requisiti. Il lavoro è particolarmente interessante perché introduce modelli valutativi, basati su funzione di valutazione multi obiettivo, al fine di bilanciare i trade-off derivanti da differenti clienti. I modelli proposti adottano scenari semplificati al fine di bilanciare il problema della fairness tra le sue differenti definizioni, infatti il primo step che il paper cita mostra come sia possibile utilizzare le tecniche di "search based optimization" al fine di giungere ad un compromesso tra le varie definizioni di fairness in specifici contesti.

L'esperimento poi dimostra come le tecniche di ricerca possano essere anche applicate a dataset reali e illustra come tali tecniche possano essere anche utilizzate per identificare i unfair bias intrinseci in tali dataset. Anche se un po' datato, questo paper evidenzia un problema che è tutt'ora attuale: aspetti intrinseci dello sviluppo software e delle tematiche AI-Intensive al giorno d'oggi mettono in luce ancora innumerevoli sfide nel campo dell'ingegnerizzazione dei requisiti e dei dati. Lo sviluppo di soluzioni AI-Intensive è strettamente influenzato da trade-off qualitativi tra cui quelli legati al mondo della fairness, e soluzioni di intelligenza artificiale, come tecniche di ricerca (e.g. algoritmi genetici), oppure modelli di ottimizzazione multi obiettivo, potrebbero sicuramente dare un'ottima risposta a queste esigenze.

Il Testing in ambito Software Fairness

Yuriy Brun et al. oltre ad essere famosi per aver posto l'accento all'emergente problema della Software Fairness, hanno condotto anche attività di ricerca specifiche come studi specifici circa il fairness testing. Uno dei loro studi più famosi del 2017 [28], propone un nuovo approccio di fair-testing, chiamato dai ricercatori Themis, al fine di misurare se e in che modo i programmi effettuano discriminazioni, focalizzandosi sulle casualità dei comportamenti discriminatori. L'approccio Themis genera test suite al fine di computare score inerenti la casual discrimination per particolari caratteristiche, e.g. secondo specifiche definizioni di fairness, il tool è in grado di generare uno score che determina quanto un sistema software discrimina contro razza ed età. Individuato un problema di discriminazione, Themis genera una test suite al fine di computare tutti i sets di caratteristiche che potrebbero essere alla base del problema. Fornendo, infine, in input al sistema di testing, una test suite manuale o auto generata, esso è in grado di verificare, su specifici input rappresentativi della popolazione, se effettivamente sono presenti feature del dataset discriminanti. L'obiettivo principale di Themis è quello di rispondere al problema di esecuzione del fairness testing per sistemi reali, (citato anche nel paper generale [1]), infatti, le tre tecniche di ottimizzazione che il sistema di testing adotta, riducono il numero di test cases necessario a computare informazioni circa i gruppi sensibili più significativi di un dataset, con l'obiettivo di individuare le cause di discriminazione che influenzano il comportamento del tool sotto test.

3.2.3 Riflessioni sullo stato dell'arte e sull'evoluzione della Software Fairness

Visione comune del mondo della ricerca, è che il concetto di Software Fairness è ancora in evoluzione, negli ultimi anni molti studi emergenti, hanno rivalutato l'importanza del

concetto cercando in vari modi di sistematizzarlo e formalizzarlo il più possibile. Come osservato in questo capitolo, sforzo principale della ricerca, è stato infatti, analizzare i vari aspetti che il concetto di Fairness racchiude, molte sono le metriche emerse che cercano di descrivere in maniera qualitativa o quantitativa i livelli di Fairness di uno specifico tool, come un generico modello di Machine Learning o una soluzione AI-Intensive.

Dagli studi analizzati, si osserva come prerogativa principale dei ricercatori nel campo dell'intelligenza artificiale, siano orientati attivamente per rendere dataset e tecniche AI-based sempre più conformi alle definizioni semantiche dell'una o l'altra metrica. A problematiche specifiche dell'intelligenza artificiale, quali il rilevamento di fair-bias nei dataset oppure applicare operazioni di fair-diversification in fase di pre- processing, l'ingegneria del software sta cercando di sistematizzare il concetto in modo tale da rendere ai Data Scientist più agevole il trattamento della Fairness. Come osservato, negli ultimi anni sono nati molti quesiti che pongono il problema del software eticamente corretto sotto una nuova luce, ovvero quella di vero e proprio requisito non funzionale prioritario per una soluzione AI-Intensive, di conseguenza, la comunità ingegneristica, cerca di considerare, come per ogni specifica non funzionale, quali sono le tecniche e metodologie migliori per trattare la Fairness in ogni fase del ciclo di vita del software. Studi come quelli inerenti al Fairness Testing, oppure le tecniche Search-based per il bilanciamento dei requisiti di fairness tra definizioni strutturalmente e semanticamente contrastanti, sono solo alcuni degli esempi di soluzioni che gli studiosi di Ingegneria del Software hanno formulato negli ultimi anni, e molti altri saranno sicuramente proposti, considerando che la ricerca in ambito di Software Fairness, con tutta probabilità andrà avanti, in maniera molto pronunciata nei prossimi anni.

Ma guardare al futuro della ricerca, significa quindi capire in che direzione sarà opportuno far evolvere il concetto di Software Fairness, al fine di renderlo sempre più vicino al mondo delle aziende di sviluppo, ed in particolare, nel contesto AI, sempre più vicino al mondo dei Data Scientist e dei modelli AI-Intensive per problematiche reali. Ovviamente è molto complesso sistematizzare il concetto di Software Fairness nella pratica lavorativa, così come è stato fatto fin ora in modo teorico, molte sono le variabili in gioco:

- Quanto e come figure come Data Scientist, Ingegneri del Software in ambito AI e figure manageriali si approccino al problema?

- Quali sono le definizioni più adottate in ambito lavorativo per i problemi fair-critical negli ambiti in cui lo sviluppo AI-Intensive si concentra oggi giorno? Quali lo saranno in futuro?

- Esistono già best-practices da seguire per garantire alti livelli di fairness di un modello di machine learning? se si, secondo quali definizioni?

Di quesiti di questo tipo se ne possono fare tanti, e ovviamente rispondere a tutto in maniera definitiva è un qualcosa che va al di fuori degli obiettivi di questo lavoro di tesi. Però in un contesto così ancora inesplorato, il budget a disposizione, permette sicuramente di partire con l'interpellare i diretti interessati. Tramite tecniche empiriche mirate si tenterà, infatti, di capire se diretti interessati (e.g. Data Scientist e Project Manager) si approcciano attualmente al problema, se effettivamente le definizioni teoriche di fairness hanno effettivo riscontro sul campo e se e come processi ingegneristici fair-orinted vengono applicati durante il ciclo di sviluppo di una soluzione AI-Intensive con obiettivo ultimo di creare una vera e propria raccolta riassuntiva dello status della pratica che abbia un duplice obiettivo a lungo termine, ovvero:

- Provare a suggerire ai ricercatori quali sono i punti di forza e di debolezza per futuri lavori plausibilmente sempre più vicini al mondo del lavoro;

- Cercare di avvicinare sempre di più gli esperti del settore AI al contesto della fairness, in modo tale che adottino sempre di più processi di sviluppo fair-oriented.

CAPITOLO 4

Obiettivi di ricerca e Survey empirico

4.1 Metodologia di ricerca

La software fairness è un argomento molto dinamico ed in continua evoluzione, che necessita di essere sistematizzato anche rispetto a come esso viene inteso all'interno del mondo lavorativo. Come osservato molti possono essere gli obiettivi da fissare e i quesiti a cui rispondere quando si parla di etica in un modulo di machine learning, ma al fine di fornire una panoramica più completa possibile sullo stato della pratica, è stato deciso di strutturare un survey empirico al fine di capire:

- Quanto e come il problema della software fairness sia percepito in azienda dal punto di vista di Manager, Data Scientist e Ingegneri del software;
- Come e in quali fasi di un canonico ciclo di vita di un modello AI-Intensive venga trattata la software fairness;
- Se è possibile sistematizzare buone e cattive pratiche da adottare durante lo sviluppo di un modulo di machine learning fair-oriented.

Perché un Survey di ricerca empirico? Progettare un Survey di ricerca che indaghi sullo status della pratica dello sviluppo si sistemi di machine learning fair, può avere un duplice vantaggio: innanzitutto interpellare esperti del dominio è un qualcosa che indirizzerà le future attività di ricerca verso la progettazione di strumentazioni e strategie che realmente possano rispecchiare le necessità di chi quotidianamente lavora nel mondo del machine

learning secondo vincoli etici sempre più stringenti, oltretutto fornire un overview iniziale delle pratiche più utilizzate può essere sicuramente d'aiuto anche agli esperti dell'ambito nel definire processi standard per trattare la fairness come altri aspetti di qualità già più sistematizzati.

4.1.1 Quesiti di ricerca

DA DEFINIRE

4.2 Design del Survey Empirico

4.2.1 Struttura e design del Survey

Al fine di bilanciare il bisogno di avere un survey ragionevolmente corto, con la necessità di renderlo abbastanza efficace da rispondere agli obiettivi di ricerca riportati nel precedente paragrafo, si è preso spunto dalla guida strutturale fornita da Andrews et al. [30], tra i principi più importanti di design da cui si è preso spunto, vale senz'altro la pena ricordare:

- La formulazione di risposte a scelta multipla o in scala (numerica o qualitativa), in modo tale da dare un carattere più analitico ai dati estraibili dalle risposte al questionario;
- L'utilizzo di un vocabolario chiaro, non ambiguo e coinciso al fine di eliminare ambiguità circa il significato della domanda;
- Specificare a priori quelli che sono gli obiettivi del Survey e chiarire da subito che le informazioni prese non saranno utilizzate per altri scopi:
- Raccogliere informazioni circa i partecipanti all'indagine, utili a categorizzare in maniera strategica i dati a disposizione;
- Garantire il rispetto della privacy specificando che i dati saranno trattati in forma anonima da un punto di vista di analisi e pubblicazione dei risultati, senza far riferimento alcuno a chi ha fornito le risposte;

Sulla base dei principi riportati, il survey di ricerca è stato formalizzato in 6 sezioni principali. Al fine di garantire consistenza di contenuto, sono presenti due domande discriminanti, utili a comprendere se il partecipante è adatto o meno a trattare aspetti specifici di software Fairness nel contesto del machine learning. Al fine di verificare se l'intervistato stia compili il questionario in maniera consona e con la dovuta attenzione, sono stati definiti due Attenction

Check Strategici che permetteranno in fase di analisi di scartare risposte non valide ai fini dell'indagine.

Per realizzare il Survey si è scelto di utilizzare la piattaforma *Google Form*. la quale in maniera nativa permette di:

- Formulare domande di vario tipo secondo le differenti esigenze di indagine;
- Formulare flussi alternativi di compilazione in base alle risposte;
- Suddividere le domande in differenti sottosezioni, coerenti con quanto progettato.

Si è scelto di adottare la strategia dei flussi alternativi, al fine di non collezionare risposte da figure senza esperienza nell'ambito dello sviluppo ML-Intensive e Fair oriented. La durata del questionario, onde evitare cali di attenzione prima della sottomissione è stata stimata attorno ai 10/15 minuti. Per reclutare i partecipanti in modo opportuno ed incentivarli alla compilazione, si è deciso di utilizzare la piattaforma specifica Prolific.

La figura 4.1 mostra una visione riassuntiva della struttura del Survey, identificando il flusso di domande principale in blu e i flussi alternativi in celeste ed in viola.

Introduzione del Survey

Prima ancora di cominciare con la compilazione del Survey, si introduce velocemente il partecipante alla problematica di ricerca connessa alla software Fairness nei sistemi AI-Intensive, fornendo anche un piccolo esempio pratico, tra quelli definiti nel capitolo stato dell'arte. Volutamente non si fornisce già all'inizio una caratterizzazione mirata del concetto, dato che è scopo dell'indagine capire se il concetto di Fairness e di Fair Bias, sia approcciato in ambito lavorativo in maniera similare rispetto a quanto formalizzato in letteratura. Inoltre viene fornita qualche informazione circa i conduttori dell'indagine empirica e sul trattamento dei dati raccolti.

Background del partecipante

La prima sezione è mirata ad acquisire informazioni circa il background dei partecipanti, al fine di poter suddividere e manipolare successivamente le risposte al questionario sulla base delle informazioni sociali, culturali, lavorative dei partecipanti. La sezione è strutturata

in modo tale da ricevere informazioni circa informazioni anagrafiche e etniche del partecipante, lo status lavorativo del partecipante, la sua esperienza lavorativa circa lo sviluppo e la realizzazione di sistemi ML-Intensive. Di seguito viene riportata una tabella riassuntiva della sezione con tutte le domande poste circa il background del partecipante.

In questa sezione, viene chiesto al partecipante se ha mai lavorato a sistemi di intelligenza artificiale o che includano moduli chi machine learning, qualora la risposta sia affermativa, il partecipante che decide di continuare con la compilazione viene indirizzato alla successiva sezione del questionario. Qualora il partecipante non avesse esperienza nello sviluppo di questi sistemi, alla pressione del tasto "Avanti", il partecipante viene condotto alla sezione di chiusura del Survey.

Domanda	***Tipo di Domanda***	***Obbligatoria***
Inserisci il tuo codice prolific	Testo Breve	No
Quanti anni hai?	Scelta multipla	No
In quale gender ti rispecchi maggiormente?	Caselle di controllo	No
Dove lavori?	Scelta multipla	No
Qual'è il maggior livello di istruzione che hai conseguito?	Scelta multipla	Sì
Qual'è la tua posizione lavorativa attuale?	Caselle di controllo	Sì
In che settori lavori attualmente?	Caselle di controllo	Sì
Qual'è il tuo ruolo professionale?	Caselle di controllo	Sì
Quanti anni di esperienza hai in questo ruolo?	Scelta multipla	Sì

Domanda	*Tipo di Domanda*	*Obbligatoria*
Hai mai lavorato allo sviluppo di soluzioni AI-Intensive o a sistemi che includono moduli di machine learning?	Scelta multipla	Sì
*** Per domanda obbligatoria si intende che il partecipante è obbligato a fornire una risposta**		

Tabella 4.1: Domande della sezione Background del Survey

Come la fairness è approcciatà a lavoro?

In questa sezione si cerca di investigare circa le attuali pratiche lavorative nello sviluppo di sistemi Fair-Critical, vengono fornite specifiche domande per fornire una visione specifica di come ed in che modo le aziende trattino la fairness in ambito lavorativo, si cerca quindi di identificare:

- Quali **aspetti specifici** del concetto di fairness sono più utili durante lo sviluppo di soluzioni ML: in riferimento alle principali categorie di metriche e approcci presenti allo stato della pratica, sono stati formulati, in maniera semplificata, una serie di approcci alla misurazione fedeli alle principali categorie di metriche presenti in letteratura [24], e viene richiesto in prima istanza quali *categorie* di metriche sia più utile per quantificare la fairness di un sistema Fair-Critical, oltre le metodologie estratte dalla letteratura, viene richiesto al partecipante di fornire anche altri approcci alternativi eventualmente utilizzati;

- Quali sono i **ruoli professionali** connessi al concetto di fairness che doverebbero essere coinvolti nello sviluppo di soluzioni fair-critical;

- Qual'è il **livello di maturità** delle aziende nel trattare software Fairness nello sviluppo ML-Intensive in maniera sistematica e standardizzata - a tal proposito è stata formalizzata una specializzazione fair-oriented del CMM (Capability Maturity Model) [31].

- In che misura **altri specifici aspetti funzionali e non funzionali** siano da confrontare rispetto la software Fairness, in ottica tale da formalizzare una visione generale di quali potrebbero essere eventuali trade-off durante lo sviluppo di sistemi ML Fair-Critical.

Ovviamente questa sezione del Survey empirico tocca tutta una serie di aspetti che meritano di essere approfonditi eventualmente con altri studi mirati, però sicuramente indagare i concetti, di cui sopra, è senz'altro un rilevante punto di inizio per la standardizzazione dei processi circa il trattamento della software Fairness.

La tabella 4.2 riporta nel dettaglio tutte le domande poste nella sezione del Survey circa l'esperienza lavorativa rispetto al concetto di Fairness.

Definizione-Domanda	***Tipo di Domanda***	***Obbligatoria***
Definizione generica di Software Fairness	Descrizione	–
Secondo te, quali dei seguenti aspetti rappresentano la definizione generica di fairness fornita in precedenza?	Griglia scelta multipla	Sì
Considerando la tua esperienza lavorativa, quanto i seguenti (approcci) sono trattati?	Griglia scelta multipla	Sì
Generalmente utilizzi altri approcci per lavorare con il concetto di Software Fairness?	Testo breve	No
Quale Bevanda(e) preferisci il sabato sera?	Caselle di controllo **	Sì
Considerando i seguenti ruoli (professionali), chi ha impatto sulle scelte inerenti la software fairness?	Griglia scelta multipla	Sì
In quale dei seguenti livelli di maturità, classificheresti il tuo ambiente lavorativo circa il trattamento della fairness?	Scelta multipla	Sì

Definizione-Domanda	*Tipo di Domanda*	*Obbligatoria*
Considerando i seguenti aspetti (funzionali e non funzionali) dello sviluppo software, quanto li ritieni importanti se comparati alla fairness?	Griglia scelta multipla	Sì
*** Per domanda obbligatoria si intende che il partecipante è obbligato a fornire una risposta**		
**** In questa sezione è presente un attenction check**		

Tabella 4.2: Domande della sezione Definizione Generale ed esperienza lavorativa del Survey

Fairness come aspetto integrante del ciclo di vita di un sistema ML-Intensive

Volendo condurre un indagine sullo stato della pratica dello sviluppo fair-oriented, è senz'altro necessario capire in quali fasi e processi dello sviluppo ML-Intensive la fairness necessiti di essere maggiormente attenzionata. Per porre quesiti che rispondano a questa macro-problematica è senz'altro necessario capire quale modello di sviluppo si adatti di più allo sviluppo di soluzioni ML-Intensive. A tal proposito, i quesiti sono stati formulati tenendo in considerazione una generica Pipeline per lo sviluppo di sistemi di machine Learning basata sulla filosofia di sviluppo MlOps (la sezione 2.3.2 del capitolo di background fornisce dettagli teorici in materia). Sulla base della pipeline, sono stati proposti dei quesiti mirati per capire effettivamente in quali fasi dello sviluppo ML-Intensive sia attualmente attenzionata e trattata come requisito primario la software Fairness nel contesto lavorativo del partecipante. Immaginando poi che le pratiche aziendali possano differire o convergere con le opinioni personali dell'intervistato, è stato previsto un quesito specifico circa l'opinione personale dell'intervistato. Progettando questa sezione, si è osservato che potrebbe essere anche utile capire quali tool commerciali o specifici possano essere particolarmente utili per trattare la Software Fairness in una pipeline di machine learning, perciò è stato posto un quesito mirato a riguardo.

La tabella 4.3 riporta nel dettaglio tutte le domande poste nella sezione del Survey circa il ciclo di vita di una soluzione ml-intensive.

Domanda	*Tipo di Domanda*	*Obbligatoria*
Considerando una generica pipeline di machine learning (come la seguente - figura 4.1), quanto consideri l'equità come un aspetto rilevante per ciascuna delle seguenti fasi nel tuo contesto lavorativo?	Griglia scelta multipla	Sì
Quali tool utilizzi (se previsti) per trattare la fairness in una pipeline di machine learning ?	Caselle di controllo	No
Contando indietro dal 5, quale numero viene dopo il 3?	Scelta multipla **	Sì
*** Per domanda obbligatoria si intende che il partecipante è obbligato a fornire una risposta**		
**** In questa sezione è presente un attenction check**		

Tabella 4.3: Domande della sezione ciclo di vita fair-oriented del Survey

Root cause e bad & best practice per trattare la fairness durante lo sviluppo ML-Intensive

Lo scopo primario di questa sezione, è quello di capire quali fattori possono essere influenti durante lo sviluppo di una soluzione ML fair-critical in termini di: cause di discriminazione, buone e cattive pratiche da adottare dorante lo sviluppo pipeline-oriented.
La sezione è stata organizzata in modo tale da includere quesiti multipli circa:

- Features sensibili note in letteratura che possono causare o meno discriminazioni se non trattate in fase di data preprocessing;
- Aspetti pratici di gestione e configurazione nell'intero flusso di una pipeline di machine learning che possono essere causa di discriminazione se non attenzionati correttamente;
- Cattive pratiche che se adottate potrebbero incentivare il nascere di fattori discriminanti nello sviluppo di una soluzione fair-critical;
- Buone pratiche che se adottate potrebbero, invece, ottimizzare la rilevazione e la mitigazione di bias durante lo sviluppo di una soluzione fair-critical.

Questa sezione, assieme alla precedente, è da considerare particolarmente rilevante per gli obiettivi dell'indagine, dato che nella sua composizione, mira proprio a rilevare e sistematizzare aspetti pratici, magari poco approfonditi fin ora dalla ricerca, che abitualmente esperti del dominio adottano durante una pipeline di sviluppo ML proprio per gestire problematiche discriminatorie.

La tabella 4.4 riporta nel dettaglio l'intero elenco di domande relative alla sezione appena descritta.

Domanda	***Tipo di Domanda***	***Obbligatoria***
Secondo la tua opinione personale, quali dei seguenti attributi possono causare discriminazioni?	Griglia scelta multipla	Sì
Secondo la tua opinione personale, ci sono altri attributi sensibili che possono causare discriminazioni? se sì, quali?	Testo Breve	No
Secondo la tua opinione personale, quali dei seguenti aspetti possono causare discriminazioni?	Griglia scelta multipla	Sì
Considerando la tua personale esperienza lavorativa, puoi classificare i seguenti aspetti come buone o cattive pratiche?	Griglia scelta multipla	Sì
Sulla base della tua esperienza lavorativa, esistono altre bad practice da non adottare per trattare software fairness in una soluzione di machine learing?	Testo lungo	No

Domanda	*Tipo di Domanda*	*Obbligatoria*
Sulla base della tua esperienza lavorativa, ci sono altre good practices che tu adotti per trattare software fairness in una soluzione di machine learing?	Testo lungo	No
*** Per domanda obbligatoria si intende che il partecipante è obbligato a fornire una risposta**		

Tabella 4.4: Domande della sezione Root causes, Bad e Best Practice del Survey

Chiusura del survey

Dopo la compilazione intrinseca del questionario, è stata preparata una sezione di chiusura che consentisse al partecipante di lasciare il proprio recapito e-mail e il riferimento al proprio profilo linkedin, in modo talr da:

- Ottenere maggiori informazioni circa le risposte fornite qualora se ne riscontrasse la necessità;
- Richiedere maggiori informazioni circa il partecipante se necessario;
- Renderlo partecipe per future indagini quali follow-up interview di approfondimento.
- Fornirgli dettagli sui risultati dell'indagine qualora fosse interessato.

La sezione inoltre prevede che il partecipante possa fornire un opinione aggiuntiva personale circa la tematica affrontata al fine di formalizzare altre informazioni utili al trattamento della software fairness nello sviluppo di soluzioni ML Intensive.

La tabella 4.5 riporta in maniera riassuntiva tutte le domande poste nella sezione di chiusura del documento.

Domanda	*Tipo di Domanda*	*Obbligatoria*
Se vuoi, puoi lasciarci maggiori informazioni circa la fairness. Qualsiasi informazione che non abbiamo considerato è importante.	Testo lungo	No

Domanda	*Tipo di Domanda*	*Obbligatoria*
Se desideri restare aggiornato circa i risultati dello studio oppure essere contattato per partecipare ad interviste di approfondimento sul topic, gentilmente scrivi qui il tuo indirizzo e-mail.	Testo breve	No
*** Per domanda obbligatoria si intende che il partecipante è obbligato a fornire una risposta**		

Tabella 4.5: Domande della sezione di chiusura del Survey

4.2.2 Validazione del Survey

Ricordare che uno survey empirico troppo lungo può facilmente causare cali di attenzione, fattore che può notevolmente inficiare la validità delle risposte raccolte[30] In tal senso è stato deciso di realizzare una simulazione pilota con studenti magistrali, frequentanti un corso accademico specifico di Ingegneria del Software per l'intelligenza artificiale. I suddetti studenti saranno invitati a compilare una copia del Survey elettronico in modo tale da stimare il tempo di esecuzione del survey su larga scala e verificare la presenza di di quesiti/definizioni di difficile interpretazione da eventualmente semplificare a seguito del test pilota.

Prerequisiti di partecipazione al test pilota

- Laurea Triennale in informatica: tutti i partecipanti dispongono di tale titolo di studi;
- Nozioni generiche di Software engineering for Artificial intelligence e MLOps Pipeline - Lo status di avanzamento del suddetto corso all'avvio del test pilota copre tutte le conoscenze basilari necessarie per affrontare l'argomento;
- Panoramica introduttiva circa la Fairness in ambito ML-Intensive, l'attività di testing pilota sarà preceduta da una lezione teorica, circa tutte le definizioni necessarie alla fairness.

Attenzione - le risposte raccolte con il test pilota, non saranno utilizzate ai fini dell'analisi dei risultati, anche perchè come osservato nella sezione successiva, il survey è rivolto a figure professionali e non accademiche. Inoltre l'aver acquisito nozioni teoriche di software fairness, basate sulle stesse fonti dello studio empirico, è una chiara minaccia alla validità dei risultati.

Risultati del test pilota

Il test pilota, nelle modalità di cui sopra, ha avuto luogo in data 09/05/2022, in totale, sono state raccolte 4 risposte di studenti, magistrali o dottorandi. Ne emerso che la durata media di compilazione del survey è stata di oscilla tra i 12 e 15 minuti, quindi del tutto accettabile con i criteri di accettazione prefissati. Contestualmente all'esecuzione del test pilota è stato richiesto agli studenti di fornire qualche feedback circa le loro impressioni sui contenuti del survey tramite l'utilizzo di una bacheca condivisa (nello specifico è stata creata un'istanza di bacheca tramite il tool web *Padlet*). Di seguito sono riassunti i principali cambiamenti apportati al Survey elettronico a seguito della chiusura del test Pilota:

Rimodulazione di contenuti forvianti: Dalle risposte ottenute al Survey pilota, ne è derivato che nonostante la stima dei tempi fosse soddisfacente, molte domande fossero state compilate con poca attenzione al contenuto, ipotesi poi confermata con i partecipanti al test pilota in una successiva riunione informale. Al fine di ridurre questo fenomeno nella più delicata fase di disseminazione del Survey, il team ha quindi deciso di rimodulare poi il questionario, rimuovendo quesiti ridondanti o non affini ai quesiti di ricerca (La strutturazione del survey presente in questo capitolo è consistente con le modifiche apportate al survey elettronico a seguito dell'esecuzione del test pilota).

Cambiamenti circa il trattamento delle informazioni sensibili: tramite il Padlet uno dei partecipanti al Survey, ha espresso delle perplessità circa il dover per forza rispondere a domande inerenti dati sensibili non di carattere lavorativo (quali sesso o età), nonostante fosse prevista la risposta "Preferisco non rispondere" per ciascuna di esse. Al fine di evitare che tale fattore possa arrecare disturbo anche durante la fase di propagazione del survey, si è deciso quindi di rendere queste domande non obbligatorie oltre a lasciare l'opzione "Preferisco non rispondere", e di valutare successivamente risposte non ottenute in modo appropriato in fase di analisi. Dopo aver apportato le opportune modifiche al survey, si ritiene che il test pilota nel suo piccolo, abbia dato riscontro positivo circa l'efficacia del survey progettato, per tanto si ritiene possibile procedere con la diffusione del survey su larga scala (nelle modalità di seguito espresse).

4.3 Reclutamento e diffusione del Survey

Lo studio empirico da condurre, è mirato per acquisire informazioni da figure professionali che abbiano lavorato all'interno dell'ambito dell'intelligenza artificiale con particolare focus su progetti fair-critical, in particolare il survey è rivolto figure professionali quali:

- Ingegneri del Software;
- Data Scientists;
- Data & Feature Engineers
- Programmatori Junior o Senior affini all'ambito ML-Intensive;
- Junior o Senior Manager aziendali affini all'ambito ML-Intensive;

4.3.1 Reclutamento dei partecipanti

Innanzitutto, una scelta chiave per la diffusione del Survey e l'acquisizione di risposte consiste nella scelta della piattaforma da utilizzare per raggiungere le figure professionali di nostro interesse. Fissando che il numero di risposte minimo da ottenere, per evitare minacce alla validità (come sarà successivamente discusso) deve necessariamente superare le 100 risposte, dopo aver effettuato un'attenta analisi delle piattaforme Social si è deciso di escludere a priori le principali piattaforme social non mirate al contesto lavorativo.

D'altra parte la crescente crescita professionale e l'enorme compatibilità con il mondo del Data Mining di **Linkedin** [32], ne fanno lo strumento ideale per ricercare ed identificare direttamente partecipanti interessanti all'indagine, nonostante la pratica possa essere più dispendiosa, si è deciso di verificare manualmente la presenza di figure di interesse a cui chiedere gentilmente di partecipare all'indagine tramite la condivisione del Survey a mezzo di e-mail. Oltre le principali piattaforme social, è stata identificata, sotto consiglio di esperti di ricerca, la piattaforma di recruitment **Prolific**, la quale permette di formalizzare vincoli circa le categorie di destinatari a cui condividere l'indagine.

Al fine di garantire una corretta separazione tra i partecipanti all'indagine raggiunti a mezzo di Linkedin, rispetto quelli raggiunti con Prolific, si è deciso di condividere due copie distinte del Survey, ognuna specifica per piattaforma in modo tale da poter classificare le risposte più agevolmente, distinguendo anche la fonte di appartenenza. Le domande tra le due distinte copie del Survey elettronico sono identiche, a meno dell'inserimento del proprio ID Prolific, ovviamente specifica per quest'ultima piattaforma.

Considerazioni aggiuntive sull'utilizzo di Prolific Come suggeriscono Reid et al. circa il 33% delle risposte sottomesse ad un Survey sottomesso con Prolific, statisticamente possono risultare invalide [33]. Al fine di ridurre al massimo le sottomissioni invalide, si è deciso di vincolare i partecipanti all'indagine Prolific tramite i seguenti filtri messi a disposizione dala piattaforma:

- Conoscenza fluente dell'inglese;
- Settore lavorativo: Informatica, Tecnologia, Ingegneria...;
- Settore Lavorativo: Informatica, Scienze, Tecnologia, Ingegneria e Matematica;
- Completamento di un alto livello di studi: Diploma o superiori.

Non essendoci su Prolific un filtro apposito per selezionare facilmente il target di utenza di interesse, si è deciso di esplicitare testualmente i requisiti di partecipazione sovra riportati, invitando gentilmente partecipanti non qualificati ad ignorare la compilazione.

4.3.2 Disponibilità e diffusione del Survey

Si è deciso di rendere disponibile ai professionisti la compilazione del survey empirico solo a seguito dell'esecuzione del test pilota, e dopo eventuali modifiche di adattamento di issiue emerse a seguito dello stesso.
Il survey empirico, sarà disponibile per la compilazione nelle modalità di cui sopra, dal 12/05/2022 fino al 29/04/2082, la data di fine potrebbe variare sensibilmente in funzione del numero di risposte ricevute e del budget temporale a disposizione.

In ogni caso, non appena sarà raggiunto un livello significativo di risposte collezionate, o allo scadere del termine ultimo fissato, tali da poter passare alla successiva fase di Data Cleaning e Analisi degli stessi, la compilazione del Survey sarà disabilitata tramite apposita funzione presente sulla piattaforma Google Moduli.

Chiusura del survey

DA COMPILARE A SEGUITO DELLA CHIUSURA DELLE ATTIVITÀ di compilazione

4.3.3 Considerazioni Etiche

Considerando le norme vigenti in Italia, è strettamente necessario porre alcune considerazioni etiche. Dato che il questionario considera l'introduzione di figure terze è stato chiarito fin da subito che:

- Il partecipante sarà invitato a rilasciare informazioni che potrebbero essere soggette a vincoli di riservatezza aziendale, Di conseguenza si rammenta che la compilazione può essere abbandonata in qualsiasi momento prima della sottomissione;
- Sarà garantito il rispetto della privacy, evitando di utilizzare le informazioni rilasciate, se non per gli scopi stessi stabiliti nella sezione introduttiva, e in ogni caso anonimizzando riferimenti diretti a dati sensibili nella rielaborazione e generalizzazione dei dati ottenuti;
- Il partecipante non vorrà comunicare tutti o alcuni dei dati sensibili richiesti, è stata prevista un opzione di risposta apposita, *Preferisco non comunicarlo*, che appunto consentirà al partecipante di restare neutrale circa la specifica informazione richiesta.
- A seguito del test pilota e prima della diffusione del survey su larga scala, è stato deciso di rendere non obbligatorie domande circa dati sensibili di carattere sociali quali sesso o etnia, per maggiori dettagli si rimanda alla sezione **Validazione del Survey**;

Qualora i partecipanti siano interessati ai futuri sviluppi dell'indagini, essi potranno in maniera facoltativa rilasciare la propria mail per ottenere un overview dei risultai e/o essere ricontattati per futuri approfondimenti. I partecipanti inoltre nella sezione di chiusura sono lasciati liberi di rilasciare qualsiasi informazione aggiuntiva che ritengano rilevante ai fini dell'indagine tramite apposito quesito aperto facoltativo.

4.4 Data Cleaning e Analisi

Scala a 5 punti, va definito in laboratorio quali strumenti potranno essere utilizzati - in ogni caso è necessario prima capire quante risposte al survey ci saranno;

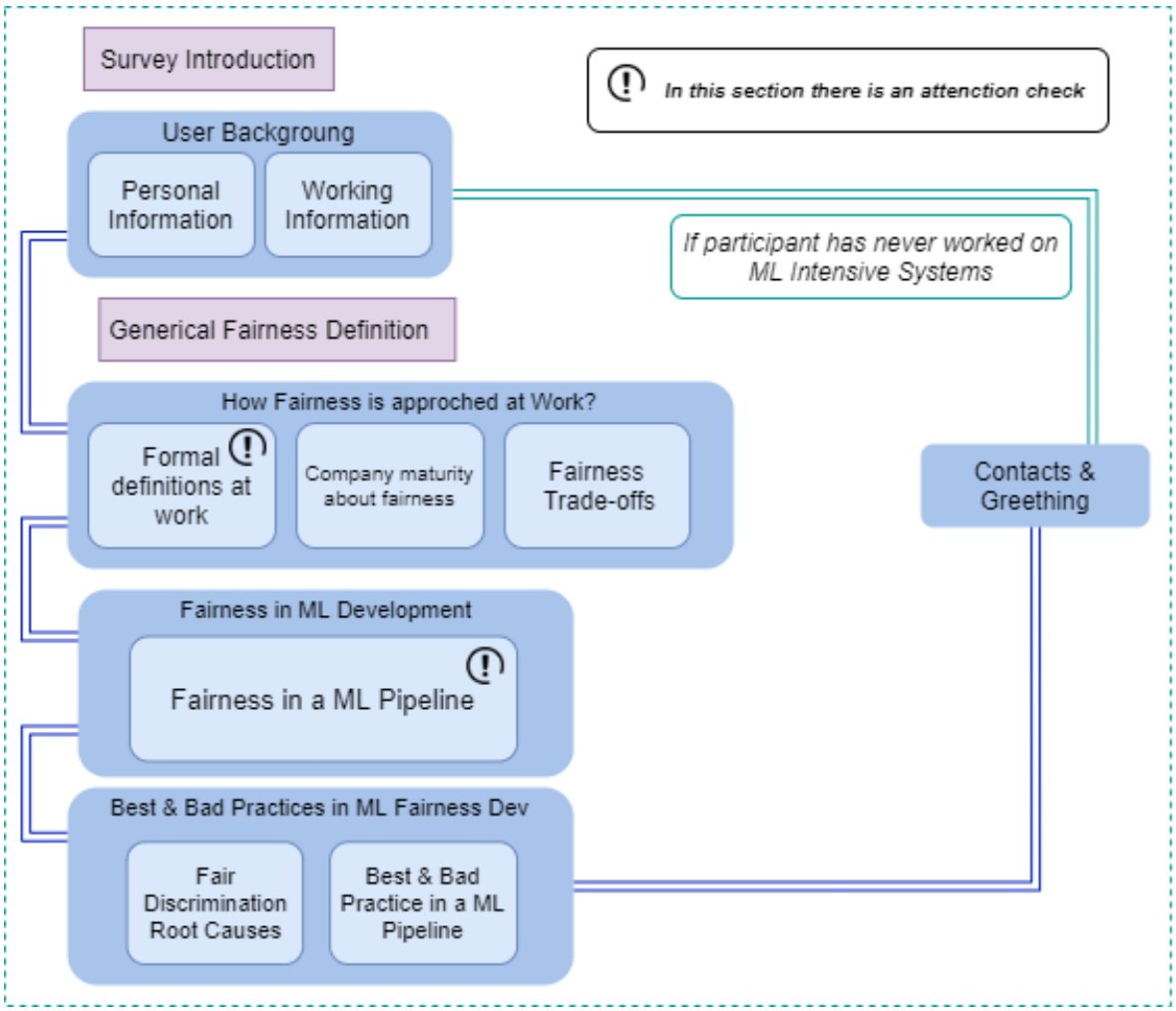

Figura 4.1: Diagramma strutturale circa le sezioni e i flussi alternativi del Survey

CAPITOLO 5

Conclusioni

BREVE SPIEGAZIONE CONTENUTO CAPITOLO

Ringraziamenti

INSERIRE RINGRAZIAMENTI QUI

Bibliografia

[1] Y. Brun and A. Meliou, "Software fairness," in *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pp. 754–759, 2018. (Citato alle pagine 2, 3, 17, 24, 28 e 31)

[2] B. Bruegge and A. Dutoit, "Object-oriented software engineering using uml, patterns, and java," 2009. (Citato alle pagine 2, 3 e 4)

[3] R. Mall, *Fundamentals of software engineering*. PHI Learning Pvt. Ltd., 2018. (Citato a pagina 2)

[4] P. Tripathy and K. Naik, *Software evolution and maintenance: a practitioner's approach*. John Wiley & Sons, 2014. (Citato a pagina 4)

[5] I. Sommerville, *Software Engineering*. Harlow, England: Addison-Wesley, 9 ed., 2010. (Citato a pagina 5)

[6] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016. (Citato a pagina 5)

[7] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017. (Citato a pagina 6)

[8] M. T. Islam, A. Fariha, and A. Meliou, "Through the data management lens: Experimental analysis and evaluation of fair classification," *arXiv preprint arXiv:2101.07361*, 2021. (Citato alle pagine 6, 16 e 21)

[9] L. Haldurai, T. Madhubala, and R. Rajalakshmi, "A study on genetic algorithm and its applications," *International journal of computer sciences and Engineering*, vol. 4, no. 10, p. 139, 2016. (Citato a pagina 7)

[10] J. Rech and K.-D. Althoff, "Artificial intelligence and software engineering: Status and future trends," *KI*, vol. 18, no. 3, pp. 5–11, 2004. (Citato alle pagine 8 e 9)

[11] J. Shaw, "Artificial intelligence and ethics," *Harvard Magazine*, vol. 30, 2019. (Citato alle pagine 8 e 9)

[12] F. Thung, S. Wang, D. Lo, and L. Jiang, "An empirical study of bugs in machine learning systems," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, pp. 271–280, 2012. (Citato a pagina 8)

[13] P. Jain, "Interaction between software engineering and artificial intelligence-a review," *International Journal on Computer Science and Engineering*, vol. 3, no. 12, p. 3774, 2011. (Citato a pagina 9)

[14] J. Horkoff, "Non-functional requirements for machine learning: Challenges and new directions," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 386–391, 2019. (Citato alle pagine 9, 10 e 16)

[15] H. Belani, M. Vukovic, and e. Car, "Requirements engineering challenges in building ai-based complex systems," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 252–255, 2019. (Citato a pagina 10)

[16] A. Burkov, *Machine learning engineering*, vol. 1. True Positive Incorporated, 2020. (Citato alle pagine 11 e 12)

[17] Y. Zhou, Y. Yu, and B. Ding, "Towards mlops: A case study of ml pipeline platform," in *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, pp. 494–500, IEEE, 2020. (Citato alle pagine 12 e 13)

[18] S. Alla and S. K. Adari, "Beginning mlops with mlflow: Deploy models in aws sagemaker, google cloud, and microsoft azure," *Beginning MLOps with MLFlow*, 2021. (Citato alle pagine 13 e 14)

[19] J. Dastin, "Amazon scraps secret ai recruiting tool that showed bias against women." `http://www.shorturl.at/agvCO`, 2018. (Citato a pagina 17)

[20] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019. (Citato a pagina 17)

[21] J. Angwin and J. Larson, "Machine bias - there's software used across the country to predict future criminals. and it's biased against blacks.." `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`, 2016. (Citato a pagina 18)

[22] R. Tatman, "Gender and dialect bias in YouTube's automatic captions," in *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, (Valencia, Spain), pp. 53–59, Association for Computational Linguistics, Apr. 2017. (Citato a pagina 18)

[23] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: why? how? what to do?," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 429–440, 2021. (Citato alle pagine 18, 24 e 26)

[24] S. Verma and J. Rubin, "Fairness definitions explained," in *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pp. 1–7, 2018. (Citato alle pagine 18, 19, 20, 21, 22, 23, 27 e 38)

[25] J. M. Zhang and M. Harman, ""ignorance and prejudice" in software fairness," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1436–1447, 2021. (Citato alle pagine 24, 25 e 27)

[26] Z. Moumoulidou, A. McGregor, and A. Meliou, "Diverse data selection under fairness constraints," *arXiv preprint arXiv:2010.09141*, 2020. (Citato alle pagine 24 e 28)

[27] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, ""fairness analysis" in requirements assignments," in *2008 16th IEEE International Requirements Engineering Conference*, pp. 115–124, IEEE, 2008. (Citato alle pagine 25 e 30)

[28] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, pp. 498–510, 2017. (Citato alle pagine 25 e 31)

[29] S. Vasudevan and K. Kenthapadi, "Lift: A scalable framework for measuring fairness in ml applications," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2773–2780, 2020. (Citato a pagina 25)

[30] D. Andrews, B. Nonnecke, and J. Preece, "Conducting research on the internet:: Online survey design, development and implementation guidelines," 2007. (Citato alle pagine 35 e 44)

[31] P. P. Kuver, "Sei cmm or iso 9000: Which is right for your organization?," in *New Directions in Project Management*, pp. 131–138, Auerbach Publications, 2001. (Citato a pagina 38)

[32] R. Sumbaly, J. Kreps, and S. Shah, "The big data ecosystem at linkedin," in *Proceedings of the 2013 acm sigmod international conference on management of data*, pp. 1125–1134, 2013. (Citato a pagina 46)

[33] B. Reid, M. Wagner, M. d'Amorim, and C. Treude, "Software engineering user study recruitment on prolific: An experience report," *arXiv preprint arXiv:2201.05348*, 2022. (Citato a pagina 47)