Certified Tester
Syllabus AI Testing (CT-AI)

Versione 1.0

International Software Testing Qualifications Board

Realizzato da

Alliance for Qualification, Artificial Intelligence United,
Chinese Software Testing Qualifications Board,
e il Korean Software Testing Qualifications Board

© International Software Testing Qualifications Board

Avviso di copyright

Copyright Notice © International Software Testing Qualifications Board (di seguito chiamato ISTQB®)

ISTQB® è un marchio registrato dell'International Software Testing Qualifications Board.

Copyright © 2021, gli autori Klaudia Dussa-Zieger (chair), Werner Henschelchen, Vipul Kocher, Qin Liu, Stuart Reid, Kyle Siemens e Adam Leon Smith.

Tutti i diritti riservati. Gli autori trasferiscono il copyright a ISTQB®. Gli autori (in qualità di attuali detentori del copyright) e ISTQB® (in qualità di futuro detentore del copyright) hanno concordato le seguenti condizioni di utilizzo:

Estratti, per uso non commerciale, da questo documento possono essere copiati se ne viene riconosciuta la fonte. Qualsiasi fornitore di formazione accreditato può usare questo syllabus come base per un corso di formazione se gli autori e ISTQB® sono riconosciuti come fonte e proprietari del copyright del syllabus e a condizione che qualsiasi pubblicità di tale corso di formazione possa menzionare il syllabus solo dopo che l'accreditamento ufficiale dei materiali di formazione sia stato ricevuto da un Member Board riconosciuto da ISTQB®.

Qualsiasi individuo o gruppo di individui può usare questo syllabus come base per articoli e libri, se gli autori e ISTQB® sono riconosciuti come fonte e proprietari del copyright del syllabus.

Qualsiasi altro uso di questo syllabus è proibito senza aver prima ottenuto l'approvazione scritta di ISTQB®.

Qualsiasi Member Board riconosciuto da ISTQB® può tradurre questo syllabus a condizione che riproduca la sopracitata nota di copyright nella versione tradotta del syllabus.

Nota importante relativa alla traduzione italiana

La traduzione italiana è stata realizzata a cura di ITA-STQB (www.ita-stqb.org).

L'indice finale non è stato tradotto ed è rimasto nella versione originale inglese.

Per segnalare eventuali imprecisioni della traduzione scrivere a exams-support@ita-stqb.org

© International Software Testing Qualifications Board

Cronologia delle revisioni

Versione	Data	Osservazioni
1.0	2021/10/01	Rilascio per GA

© International Software Testing Qualifications Board

Sommario

Avviso di copyright 2
Cronologia delle revisioni 3
Sommario 4
Ringraziamenti 8
0 Introduzione 9
0.1 Scopo di questo Syllabus 9
0.2 Il Certified Tester AI Testing 9
0.3 Obiettivi di apprendimento esaminabili e Livello Cognitivo di Conoscenza 9
0.4 Livelli di competenza pratica 10
0.5 L'esame Certified Tester AI Testing 10
0.6 Accreditamento 11
0.7 Livello di dettaglio 11
0.8 Come è organizzato questo Syllabus 11
1 Introduzione all'AI - 105 minuti 13
1.1 Definizione di AI e di effetto AI 14
1.2 AI Ristretta, AI Generale e Super AI 14
1.3 Sistemi basati su AI e sistemi convenzionali 14
1.4 Tecnologie di AI 15
1.5 Framework di sviluppo dell'AI 15
1.6 Hardware per sistemi basati su AI 16
1.7 AI as a Service (AIaaS) 17
1.7.1 Contratti per l'AI as a Service 17
1.7.2 Esempi di AIaaS 18
1.8 Modelli pre-addestrati 18
1.8.1 Introduzione ai modelli pre-addestrati 18
1.8.2 Transfer Learning 19
1.8.3 Rischi dell'utilizzo di modelli pre-addestrati e del transfer learning 19
1.9 Standard, regolamenti e AI 20
2 Caratteristiche di qualità per i sistemi basati su AI - 105 minuti 21
2.1 Flessibilità e adattabilità 22
2.2 Autonomia 22
2.3 Evoluzione 22

© International Software Testing Qualifications Board

2.4 Bias ... 23
2.5 Etica ... 23
2.6 Effetti collaterali e reward hacking ... 24
2.7 Trasparenza, interpretabilità e spiegabilità ... 25
2.8 Safety e AI ... 25
3 Machine Learning (ML) - Panoramica - 145 minuti ... 27
3.1 Forme di ML ... 28
3.1.1 Apprendimento supervisionato ... 28
3.1.2 Apprendimento non supervisionato ... 28
3.1.3 Apprendimento per rinforzo ... 29
3.2 Flusso di lavoro di ML ... 29
3.3 Selezionare una forma di ML ... 32
3.4 Fattori coinvolti nella selezione dell'algoritmo di ML ... 33
3.5 Overfitting e Underfitting ... 33
3.5.1 Overfitting ... 33
3.5.2 Underfitting ... 34
3.5.3 Esercizio pratico: Dimostrare l'overfitting e l'underfitting ... 34
4 ML - Dati - 230 minuti ... 35
4.1 Preparazione dei dati come parte del flusso di lavoro di ML ... 36
4.1.1 Sfide nella preparazione dei dati ... 37
4.1.2 Esercizio pratico: Preparazione dei dati per il ML ... 37
4.2 Set di dati di addestramento, validazione e test nel flusso di lavoro di ML ... 38
4.2.1 Esercizio pratico: Identificare i dati di addestramento e di test e creare un modello di ML 39
4.3 Problemi di qualità dei set di dati ... 39
4.4 La qualità dei dati e il suo effetto sul modello di ML ... 40
4.5 Etichettatura dei dati per l'apprendimento supervisionato ... 41
4.5.1 Approcci all'etichettatura dei dati ... 41
4.5.2 Dati etichettati erroneamente nei set di dati ... 41
5 Metriche delle prestazioni funzionali del ML - 120 minuti ... 43
5.1 Matrice di confusione ... 44
5.2 Ulteriori metriche delle prestazioni funzionali del ML per classificazione, regressione e clustering ... 45
5.3 Limitazioni delle metriche delle prestazioni funzionali del ML ... 45
5.4 Selezionare le metriche delle prestazioni funzionali del ML ... 46
5.4.1 Esercizio pratico: Valutare il modello di ML creato ... 47

© International Software Testing Qualifications Board

5.5 Suite di benchmark per il ML 47
6 ML - Reti neurali e testing - 65 minuti 48
6.1 Reti neurali 49
6.1.1 Esercizio pratico: Implementare un percettrone semplice 50
6.2 Misure di copertura per le reti neurali 51
7 Panoramica sul testing di sistemi basati su AI - 115 minuti 53
7.1 Specifiche dei sistemi basati su AI 54
7.2 Livelli di test per i sistemi basati su AI 54
7.2.1 Testing dei dati di input 55
7.2.2 Testing del modello di ML 55
7.2.3 Testing di componente 55
7.2.4 Testing di integrazione dei componenti 55
7.2.5 Testing di sistema 56
7.2.6 Testing di accettazione 56
7.3 Dati di test per il testing dei sistemi basati su AI 56
7.4 Testing per l'automation bias nei sistemi basati su AI 57
7.5 Documentare un componente AI 57
7.6 Testing per la deriva concettuale 58
7.7 Selezione di un approccio di test per un sistema di ML 58
8 Testing delle caratteristiche di qualità specifiche dell'AI - 150 minuti 61
8.1 Sfide nel testing dei sistemi di autoapprendimento 62
8.2 Testing dei sistemi autonomi basati su AI 63
8.3 Testing per bias algoritmici, di campionamento e inappropriati 63
8.4 Sfide nel testing dei sistemi basati su AI probabilistici e non deterministici 64
8.5 Sfide nel testing dei sistemi complessi basati su AI 64
8.6 Testare la trasparenza, l'interpretabilità e la spiegabilità dei sistemi basati su AI 65
8.6.1 Esercizio pratico: Spiegabilità del modello 66
8.7 Oracoli di test per sistemi basati su AI 66
8.8 Obiettivi dei test e criteri di accettazione 67
9 Metodi e tecniche per il testing di sistemi basati su AI - 245 minuti 69
9.1 Adversarial attacks e avvelenamento dei dati 70
9.1.1 Adversarial attacks 70
9.1.2 Avvelenamento dei dati 70
9.2 Testing pairwise 71
9.2.1 Esercizio pratico: Testing pairwise 71

© International Software Testing Qualifications Board

9.3 Testing back-to-back 72
9.4 Testing A/B 72
9.5 Testing metamorfico (MT) 73
9.5.1 Esercizio pratico: Testing metamorfico 74
9.6 Testing basato sull'esperienza di sistemi basati su AI 74
9.6.1 Esercizio pratico: Testing esplorativo e Exploratory Data Analysis (EDA) 76
9.7 Selezione delle tecniche di test per i sistemi basati su AI 76
10 Ambienti di test per sistemi basati su AI - 30 minuti 78
10.1 Ambienti di test per i sistemi basati su AI 79
10.2 Ambienti di test virtuali per il testing dei sistemi basati su AI 79
11 Usare l'AI per il testing - 195 minuti 81
11.1 Tecnologie di AI per il testing 82
11.1.1 Esercizio pratico: L'uso dell'AI nel testing 82
11.2 Utilizzare l'AI per analizzare i difetti segnalati 82
11.3 Utilizzare l'AI per la generazione di casi di test 83
11.4 Utilizzare l'AI per l'ottimizzazione delle suite di test di regressione 83
11.5 Utilizzare l'AI per la previsione dei difetti 84
11.5.1 Esercizio pratico: Costruire un sistema di previsione dei difetti 84
11.6 Utilizzare l'AI per il testing delle interfacce utente 84
11.6.1 Utilizzare l'AI per testare tramite la Graphical User Interface (GUI) 84
11.6.2 Utilizzare l'AI per testare la GUI 85
12 Riferimenti 86
12.1 Standard 86
12.2 Documenti ISTQB® 86
12.3 Libri e articoli 87
12.4 Altri riferimenti 90
13 Appendice A - Abbreviazioni 92
14 Appendice B - Termini specifici di AI e altri termini 93
15 Indice 104

© International Software Testing Qualifications Board

Ringraziamenti

Questo documento è stato formalmente rilasciato dall'Assemblea Generale di ISTQB® il 1° ottobre 2021.

È stato redatto da un team dell'International Software Testing Qualifications Board: Klaudia Dussa-Zieger (chair), Werner Henschelchen, Vipul Kocher, Qin Liu, Stuart Reid, Kyle Siemens e Adam Leon Smith.

Il team ringrazia gli autori dei tre syllabi che hanno contribuito;

- A4Q: Rex Black, Bruno Legeard, Jeremias Rößler, Adam Leon Smith, Stephan Goericke, Werner Henschelchen
- AiU: Autori principali Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni e Sonika Bengani e coautori Rik Marselis, José M. Diaz Delgado
- KSTQB & CSTQB AIT: Stuart Reid, Qin Liu, Yeoungjae Choi, Murian Song, Wonil Kwon

Il team ringrazia i gruppi di lavoro Esame, Glossario e Marketing per il loro supporto durante lo sviluppo del syllabus, Graham Bath per il suo editing tecnico e le Member Board per i loro suggerimenti e input.

Le seguenti persone hanno partecipato alla revisione e al commento di questo syllabus: Laura Albert, Reto Armuzzi, Árpád Beszédes, Armin Born, Géza Bujdosó, Renzo Cerquozzi, Sudeep Chatterjee, Seunghee Choi, Young-jae Choi, Piet de Roo, Myriam Christener, Jean-Baptiste Crouigneau, Guofu Ding, Erwin Engelsma, Hongfei Fan, Péter Földházi Jr., Tamás Gergely, Ferdinand Gramsamer, Attila Gyúri, Matthias Hamburg, Tobias Horn, Jarosław Hryszko, Beata Karpinska, Joan Killeen, Rik Kochuyt, Thomas Letzkus, Chunhui Li, Haiying Liu, Gary Mogyorodi, Rik Marselis, Imre Mészáros, Tetsu Nagata, Ingvar Nordström, Gábor Péterffy, Tal Pe'er, Ralph Pichler, Nishan Portoyan, Meile Posthuma, Adam Roman, Gerhard Runze, Andrew Rutz, Klaus Skafte, Mike Smith, Payal Sobti, Péter Sótér, Michael Stahl, Chris van Bael, Stephanie van Dijck, Robert Werkhoven, Paul Weymouth, Dong Xin, Ester Zabar, Claude Zhang.

Le seguenti persone hanno partecipato alla traduzione nella lingua italiana di questo syllabus: Salvatore Reale, Raffaele Bozzini, Giorgio Pisani

© International Software Testing Qualifications Board

0 Introduzione

0.1 Scopo di questo Syllabus

Questo syllabus costituisce la base per il Certified Tester AI Testing ISTQB ®. ISTQB ® fornisce questo syllabus come segue:

1. Ai member board, per tradurlo nella loro lingua locale e accreditare i fornitori di formazione. I Member board possono adattare il syllabus alle loro particolari esigenze linguistiche e modificare i riferimenti per adattarli alle loro pubblicazioni locali.
2. Agli organismi di certificazione, per derivare domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento di questo syllabus.
3. Ai fornitori di formazione, per produrre materiale didattico e determinare metodi di insegnamento appropriati.
4. Ai candidati alla certificazione, per prepararsi all'esame di certificazione (sia come parte di un corso di formazione che in modo indipendente).
5. Alla comunità internazionale di ingegneria del software e dei sistemi, per far progredire la professione del testing del software e dei sistemi, e come base per libri e articoli.

0.2 Il Certified Tester AI Testing

Il Certified Tester AI Testing è rivolto a chiunque sia coinvolto nel testing di sistemi basati su AI e/o l'uso di AI per il testing. È quindi indicato per chi ricopre il ruolo di tester, test analyst, data analyst, test engineer, test consultant, test manager, user acceptance tester e software developer. Questa certificazione è anche indicata per chiunque desideri una comprensione di base del testing di sistemi basati su AI e/o dell'uso di AI per il testing, come project manager, quality manager, software development manager, business analyst, membri del team operativo, direttori IT e management consultant.

Il Certified Tester AI Testing Overview [I03] è un documento separato che include le seguenti informazioni:

- Risultati aziendali per il syllabus
- Matrice dei risultati aziendali e connessione con gli obiettivi di apprendimento
- Riassunto del syllabus
- Relazioni tra i syllabi

0.3 Obiettivi di apprendimento esaminabili e Livello Cognitivo di Conoscenza

Gli obiettivi di apprendimento supportano i risultati aziendali e sono utilizzati per creare gli esami Certified Tester AI Testing.

Ai candidati può essere chiesto di riconoscere, ricordare o richiamare una parola chiave o un concetto menzionato in uno qualsiasi degli undici capitoli. I livelli specifici degli obiettivi di apprendimento sono indicati all'inizio di ogni capitolo e classificati come segue:

© International Software Testing Qualifications Board

- K1: Ricordare
- K2: Comprendere
- K3: Applicare
- K4: Analizzare

Tutti i termini elencati come parole chiave appena sotto i titoli dei capitoli dovranno essere ricordati (K1), anche se non esplicitamente menzionati negli obiettivi di apprendimento.

0.4 Livelli di competenza pratica

Certified Tester Specialist AI Testing include obiettivi pratici incentrati su abilità e competenze pratiche.

I seguenti livelli si applicano agli obiettivi pratici (come mostrato):

- H0: Demo live di un esercizio o video registrato.
- H1: Esercizio guidato. Gli studenti seguono una sequenza di passaggi eseguiti dal formatore.
- H2: Esercizio con suggerimenti. Allo studente viene dato un esercizio con suggerimenti pertinenti in modo che l'esercizio possa essere risolto entro il tempo stabilito, oppure gli studenti partecipano a una discussione.

Le competenze sono raggiunte eseguendo esercizi pratici, come riportato nella seguente lista:

- Dimostrare l'underfitting e l'overfitting (H0).
- Eseguire la preparazione dei dati a supporto della creazione di un modello di ML (H2).
- Identificare i dataset di formazione e di test e creare un modello di ML (H2).
- Valutare il modello di ML creato usando metriche di performance funzionali ML selezionate (H2).
- Esperienza di implementazione di un percettrone (H1).
- Uso di uno strumento per mostrare come la spiegabilità possa essere usata dai tester (H2).
- Applicare il testing pairwise per derivare ed eseguire test case per un sistema basato su AI (H2).
- Applicare il testing metamorfico per derivare ed eseguire test case per un dato scenario (H2).
- Applicare il testing esplorativo a un sistema basato su AI (H2).
- Discutere, usando esempi, quelle attività nel testing in cui è meno probabile che l'AI venga usata (H2).
- Implementare un semplice sistema di previsione dei difetti basato su AI (H2).

0.5 L'esame Certified Tester AI Testing

L'esame Certified Tester AI Testing sarà basato su questo syllabus. Le risposte alle domande d'esame possono richiedere l'uso di materiale basato su più di una sezione di questo syllabus. Tutte le sezioni del syllabus sono esaminabili, tranne l'introduzione e le appendici. Gli standard e i libri sono

© International Software Testing Qualifications Board

inclusi come riferimenti, ma il loro contenuto non è esaminabile, al di là di ciò che è riassunto nel syllabus stesso da tali standard e libri.

Si faccia riferimento al documento Certified Tester Specialist AI Testing "Overview" per ulteriori dettagli nella sezione "Exam Structure".

Nota sui requisiti di ingresso: Il certificato ISTQB® Certified Tester Foundation Level deve essere ottenuto prima di sostenere l'esame Certified Tester Specialist AI Testing.

0.6 Accreditamento

Un Member Board ISTQB® può accreditare i fornitori di formazione il cui materiale del corso segue questo syllabus. I fornitori di formazione dovrebbero ottenere le linee guida per l'accreditamento dal Member Board o dall'ente che effettua l'accreditamento. Un corso accreditato è riconosciuto come conforme a questo syllabus e può avere un esame ISTQB® come parte del corso.

Le linee guida di accreditamento per questo syllabus seguono le Linee guida generali di Accreditamento pubblicate dal Processes Management and Compliance Working Group.

0.7 Livello di dettaglio

Il livello di dettaglio di questo syllabus permette corsi ed esami coerenti a livello internazionale. Per raggiungere questo obiettivo, il syllabus è composto da:

- Obiettivi didattici generali che descrivono l'intenzione del Certified Tester AI Testing.
- Una lista di termini che gli studenti devono essere in grado di ricordare.
- Obiettivi di apprendimento e pratici per ogni area di conoscenza, che descrivono i risultati di apprendimento da raggiungere.
- Una descrizione dei concetti chiave, compresi i riferimenti a fonti come la letteratura accettata o gli standard.

Il contenuto del syllabus non è una descrizione dell'intera area di conoscenza per il testing dei sistemi basati su AI, bensì riflette il livello di dettaglio da coprire nei corsi di formazione Certified Tester Specialist AI Testing. Si concentra sull'introduzione dei concetti di base dell'intelligenza artificiale (AI) e del machine learning in particolare, e su come i sistemi basati su queste tecnologie possono essere testati.

0.8 Come è organizzato questo Syllabus

Ci sono undici capitoli con contenuti esaminabili. L'intestazione principale per ogni capitolo specifica il tempo per il capitolo; il tempo non è fornito per i sotto capitoli. Per i corsi di formazione accreditati, il syllabus richiede un minimo di 25.1 ore di istruzione, distribuite attraverso gli undici capitoli come segue:

- Capitolo 1: 105 minuti Introduzione all'AI
- Capitolo 2: 105 minuti Caratteristiche di qualità per i sistemi basati su AI
- Capitolo 3: 145 minuti Machine Learning (ML) - Panoramica
- Capitolo 4: 230 minuti ML - Dati
- Capitolo 5: 120 minuti Metriche delle prestazioni funzionali del ML

© International Software Testing Qualifications Board

- Capitolo 6: 65 minuti ML - Reti neurali e testing
- Capitolo 7: 115 minuti Panoramica sul testing di sistemi basati su AI
- Capitolo 8: 150 minuti Testare le caratteristiche di qualità specifiche dell'AI
- Capitolo 9: 245 minuti Metodi e tecniche per il testing di sistemi basati su AI
- Capitolo 10: 30 minuti Ambienti di test per sistemi basati su AI
- Capitolo 11: 195 minuti Usare l'AI per il testing

© International Software Testing Qualifications Board

1 Introduzione all'AI - 105 minuti

Parole chiave del testing

Nessuna

Parole chiave specifiche per l'AI

AI as a Service (AIaaS), framework di sviluppo dell'AI, effetto AI, sistema basato su AI, intelligenza artificiale (AI), rete neurale, deep learning (DL), rete neurale profonda, AI generale, General Data Protection Regulation (GDPR), machine learning (ML), AI ristretta, modello pre-addestrato, super AI, singolarità tecnologica, transfer learning

Obiettivi di apprendimento per il Capitolo 1:

1.1 Definizione di AI e di effetto AI

AI-1.1.1 K2 Descrivere l'effetto AI e come influenza la definizione di AI.

1.2 AI Ristretta, AI Generale e Super AI

AI-1.2.1 K2 Distinguere tra AI ristretta, AI generale e super AI.

1.3 Sistemi basati su AI e sistemi convenzionali.

AI-1.3.1 K2 Differenziare tra sistemi basati su AI e sistemi convenzionali.

1.4 Tecnologie di AI

AI-1.4.1 K1 Riconoscere le diverse tecnologie utilizzate per implementare l'AI.

1.5 Framework di sviluppo dell'AI

AI-1.5.1 K1 Identificare i framework di sviluppo dell'AI popolari.

1.6 Hardware per sistemi basati su AI

AI-1.6.1 K2 Confrontare le scelte disponibili per l'hardware per implementare sistemi basati su AI.

1.7 AI as a Service (AIaaS)

AI-1.7.1 K2 Spiegare il concetto di AI as a Service (AIaaS).

1.8 Modelli pre-addestrati

AI-1.8.1 K2 Spiegare l'uso di modelli AI pre-addestrati e i rischi ad essi associati.

1.9 Standard, regolamenti e AI

AI-1.9.1 K2 Descrivere come si applicano gli standard ai sistemi basati su AI.

© International Software Testing Qualifications Board

1.1 Definizione di AI e di effetto AI

Il termine intelligenza artificiale (AI) risale agli anni '50 e si riferisce all'obiettivo di costruire e programmare macchine "intelligenti" in grado di imitare gli esseri umani. La definizione oggi si è evoluta notevolmente e la seguente definizione ne cattura il concetto [S01]:

> *La capacità di un sistema ingegnerizzato di acquisire, elaborare e applicare conoscenze e competenze.*

Il modo in cui le persone comprendono il significato di AI dipende dalla loro percezione attuale. Negli anni '70 l'idea di un sistema informatico in grado di battere un umano a scacchi era una visione futuristica e la maggior parte delle persone consideravano questa idea come AI. Ora, più di vent'anni dopo che il sistema basato su computer Deep Blue ha battuto il campione mondiale di scacchi Garry Kasparov, l'approccio di "forza bruta" implementato in quel sistema non è considerato da molti come vera intelligenza artificiale (cioè, il sistema non imparava dai dati e non era capace di autoapprendimento). Allo stesso modo, i sistemi esperti degli anni '70 e '80 incorporavano la competenza umana come regole che potevano essere eseguite ripetutamente senza che l'esperto fosse presente. Questi sistemi all'epoca erano considerati AI, mentre oggi non è più così.

La percezione mutevole di ciò che costituisce l'AI è nota come "Effetto AI" [R01]. Man mano che la percezione dell'AI nella società cambia, così cambia anche la sua definizione. Di conseguenza, qualsiasi definizione fatta oggi è probabile che cambi in futuro e potrebbe non corrispondere a quelle del passato.

1.2 AI Ristretta, AI Generale e Super AI

Ad un livello elevato, l'AI può essere suddivisa in tre categorie:

- I sistemi di **AI ristretta** (conosciuta anche come AI debole) sono programmati per svolgere un compito specifico con un contesto limitato. Attualmente questa forma di AI è ampiamente disponibile. Per esempio, sistemi di gioco, filtri antispam, generatori di test case e assistenti vocali.
- I sistemi di **AI generale** (nota anche come AI forte) hanno capacità cognitive generali (ad ampio raggio) simili agli umani. Questi sistemi basati su AI possono ragionare e comprendere il loro ambiente come fanno gli umani e agire di conseguenza. Ad oggi (2021), nessun sistema di AI generale è stato realizzato.
- I sistemi di **super AI** sono in grado di replicare la cognizione umana (AI generale) e fanno uso di una massiccia potenza di elaborazione, una memoria praticamente illimitata e l'accesso a tutta la conoscenza umana (ad esempio, attraverso l'accesso al web). Si presume che i sistemi di super AI diventeranno rapidamente più saggi degli umani. Il punto in cui i sistemi basati su AI passano dall'AI generale alla super AI è comunemente noto come "singolarità tecnologica" [B01].

1.3 Sistemi basati su AI e sistemi convenzionali

In un tipico sistema informatico convenzionale, il software è programmato dagli umani usando un linguaggio imperativo, che include costrutti come if-then-else e loop. Questo rende relativamente semplice per gli umani capire come il sistema trasforma gli input in output. In un sistema basato su AI che usa il machine learning (ML), i pattern nei dati sono usati dal sistema per determinare come dovrebbe reagire in futuro a nuovi dati (vedi il Capitolo 3 per una spiegazione dettagliata del ML). Per esempio, un processore di immagini basato su AI progettato per identificare immagini di gatti viene

© International Software Testing Qualifications Board

addestrato con un insieme di immagini note per contenere gatti. L'AI determina da sola quali pattern o caratteristiche nei dati possono essere usati per identificare i gatti. Questi pattern e regole vengono poi applicati a nuove immagini per determinare se contengono o meno gatti. In molti sistemi basati su AI, ciò si traduce in una procedura di predizione meno facile da comprendere per gli esseri umani (vedi Sezione 2.7).

Nella pratica, i sistemi basati su AI possono essere implementati da una varietà di tecnologie (vedi Sezione 1.4), e l'"Effetto AI" (vedi Sezione 1.1) può far distinguere ciò che è attualmente considerato un sistema basato su AI da ciò che è considerato un sistema convenzionale.

1.4 Tecnologie di AI

L'AI può essere implementata utilizzando una vasta gamma di tecnologie (vedi [B02] per maggiori dettagli), come:

- Logica fuzzy
- Algoritmi di ricerca
- Tecniche di ragionamento
  - Motori delle regole
  - Classificatori deduttivi
  - Ragionamento basato sul caso
  - Ragionamento procedurale
- Tecniche di machine learning
  - Reti neurali
  - Modelli bayesiani
  - Alberi di decisione
  - Foresta casuale
  - Regressione lineare
  - Regressione logistica
  - Algoritmi di clustering
  - Algoritmi genetici
  - Macchina a vettore di supporto (SVM)

I sistemi basati su AI tipicamente implementano una o più di queste tecnologie.

1.5 Framework di sviluppo dell’AI

Ci sono molti framework di sviluppo dell’AI disponibili, alcuni dei quali sono focalizzati su domini specifici. Questi framework supportano una serie di attività, come la preparazione dei dati, la selezione degli algoritmi e la compilazione di modelli da eseguire su vari processori, come unità centrali di elaborazione (CPU), unità di elaborazione grafica (GPU) o Cloud Tensor Processing Unit (TPU). La scelta di un particolare framework può dipendere anche da aspetti particolari come il linguaggio di programmazione utilizzato per l'implementazione e la sua facilità d'uso. I seguenti framework sono alcuni dei più popolari (ad aprile 2021):

© International Software Testing Qualifications Board

- **Apache MxNet**: Un framework open-source di deep learning usato da Amazon per Amazon Web Services (AWS) [R02].
- **CNTK**: Il Microsoft Cognitive Toolkit (CNTK) è un toolkit open-source di deep learning [R03].
- **IBM Watson Studio**: Una suite di strumenti che supportano lo sviluppo di soluzioni AI [R04].
- **Keras**: Un'API open-source di alto livello, scritta in linguaggio Python, capace di funzionare su TensorFlow e CNTK [R06].
- **PyTorch**: Una libreria di ML open-source gestita da Facebook e utilizzata per applicazioni che applicano l'elaborazione delle immagini e del linguaggio naturale (NLP). Il supporto è fornito per entrambe le interfacce Python e C++ [R07].
- **Scikit-learn**: Una libreria open-source di ML per il linguaggio di programmazione Python [R08].
- **TensorFlow**: Un framework di ML open-source basato su grafici di flusso dati per il machine learning scalabile, fornito da Google [R05].

Si noti che questi framework di sviluppo sono in continua evoluzione. Questi possono combinarsi ed essere sostituiti da nuovi framework.

1.6 Hardware per sistemi basati su AI

Diverse tipologie di hardware vengono utilizzate per l'addestramento del modello di ML (vedi Capitolo 3) e per l'implementazione del modello. Per esempio, un modello che esegue il riconoscimento vocale può funzionare su uno smartphone di fascia bassa, anche se potrebbe essere necessario l'accesso alla potenza del cloud computing per addestrarlo. Un approccio comune utilizzato quando l'host device non è connesso a Internet è quello di addestrare il modello nel cloud e poi distribuirlo sull'host device.

Il ML beneficia tipicamente dell'hardware avente le seguenti caratteristiche:

- Aritmetica a bassa precisione: Questo usa meno bit per il calcolo (ad esempio, 8 bit invece di 32, che di solito è tutto ciò che è necessario per il ML).
- La capacità di lavorare con grandi strutture di dati (ad esempio, per supportare la moltiplicazione di matrici).
- Elaborazione massicciamente parallela (simultanea).

Le CPU general-purpose forniscono supporto per operazioni complesse che non sono tipicamente richieste per le applicazioni di ML e forniscono solo pochi core. Per questo motivo, la loro architettura è meno efficiente per l'addestramento e l'esecuzione di modelli di ML rispetto alle GPU, che hanno migliaia di core e che sono progettate per eseguire l'elaborazione massicciamente parallela ma relativamente semplice di immagini. Di conseguenza, le GPU in genere hanno prestazioni migliori delle CPU per le applicazioni di ML, anche se le CPU hanno tipicamente velocità di clock più elevate. Per il lavoro di ML su piccola scala, le GPU offrono generalmente l'opzione migliore.

Alcuni hardware sono appositamente concepiti per l'AI, come gli Application-Specific Integrated Circuits (ASIC) e i System on a Chip (SoC). Queste soluzioni specifiche per l'AI hanno caratteristiche ottimali quali core multipli, gestione speciale dei dati e la capacità di eseguire l'elaborazione in memoria. Sono più adatti per l'edge computing, mentre l'addestramento del modello di ML viene fatto nel cloud.

© International Software Testing Qualifications Board

Hardware con architetture di AI specifiche sono attualmente (ad aprile 2021) in fase di sviluppo. Ciò include i processori neuromorfici [B03], i quali, invece che utilizzare la tradizionale architettura von Neumann, implementano un'architettura che imita vagamente i neuroni del cervello.

Esempi di fornitori di hardware di AI e dei loro processori includono (ad aprile 2021):

- **NVIDIA**: Forniscono una gamma di GPU e processori specifici per l'AI, come Volta [R09].
- **Google**: Hanno sviluppato circuiti integrati specifici per applicazioni sia per l'addestramento che per l'inferenza. Le TPU di Google (Cloud Tensor Processing Units) [R10] sono accessibili agli utenti su Google Cloud, mentre la Edge TPU [R11] è un ASIC appositamente costruito per eseguire l'AI sui singoli dispositivi.
- **Intel**: Forniscono i processori per reti neurali Nervana [R12] per il deep learning (sia l'addestramento che l'inferenza) e le unità di elaborazione visiva Movidius Myriad per l'inferenza nelle applicazioni di computer vision e reti neurali.
- **Mobileye**: Producono la famiglia EyeQ di dispositivi SoC [R13] per supportare l'elaborazione visiva complessa e computazionalmente intensa. Hanno un basso consumo energetico per l'uso nei veicoli.
- **Apple**: Producono il chip Bionic per l'AI on-device negli iPhone [B04].
- **Huawei**: Il loro chip Kirin 970 per smartphone ha un'elaborazione di rete neurale integrata per l'AI [B05].

1.7 AI as a Service (AIaaS)

I componenti AI, come i modelli ML, possono essere creati all'interno di un'organizzazione, scaricati da una terza parte, o usati come servizio sul web (AIaaS). È anche possibile un approccio ibrido in cui alcune delle funzionalità AI sono fornite dall'interno del sistema e altre sono fornite come servizio.

Quando il ML è usato come servizio, viene fornito l'accesso a un modello di ML attraverso il web e un supporto può anche essere fornito per la preparazione e l'archiviazione dei dati, l'addestramento del modello, la valutazione, il tuning, il testing e la distribuzione.

I fornitori di terze parti (ad esempio, AWS, Microsoft) offrono servizi di AI specifici, come il riconoscimento facciale e vocale. Ciò permette agli individui e alle organizzazioni di implementare l'AI usando servizi basati sul cloud anche quando non hanno risorse e competenze sufficienti per costruire i propri servizi di AI. Inoltre, i modelli di ML forniti come parte di un servizio di terze parti sono verosimilmente stati addestrati su un set di dati di formazione più ampio e diversificato di quello che è nell'immediato disponibile per molti stakeholder, come quelli che hanno approcciato da poco il mercato dell'AI.

1.7.1 Contratti per l'AI as a Service

Questi servizi AI sono tipicamente forniti con contratti simili a quelli per il Software as a Service (SaaS) basato su cloud non-AI. Un contratto per AIaaS tipicamente include un accordo sul livello di servizio (SLA) che definisce la disponibilità e la sicurezza dei commitment. Tali SLA coprono tipicamente un uptime per il servizio (ad esempio, 99,99% di uptime) e un tempo di risposta per correggere i difetti, ma raramente definiscono metriche di performance funzionali ML, (come l'accuratezza), in modo simile (vedi Capitolo 5). AIaaS è spesso pagato sotto forma di abbonamento e, se la disponibilità e/o i tempi di risposta non sono soddisfatti, allora il fornitore di servizi tipicamente fornisce crediti per servizi futuri. Oltre a questi crediti, la maggior parte dei contratti AIaaS forniscono una responsabilità limitata (oltre che in termini di tariffe pagate), il che significa che i sistemi basati su

© International Software Testing Qualifications Board

AI che dipendono da AIaaS sono solitamente riservati ad applicazioni a rischio relativamente basso, dove la perdita del servizio non sarebbe troppo dannosa.

I servizi spesso vengono forniti con un periodo iniziale di prova gratuita invece che di un periodo di accettazione. Durante questo periodo ci si aspetta che il consumatore dell'AIaaS verifichi se il servizio fornito soddisfa le proprie esigenze in termini di funzionalità e prestazioni richieste (ad esempio, l'accuratezza). Questo è generalmente necessario per coprire qualsiasi mancanza di trasparenza sul servizio fornito (vedi Sezione 7.5).

1.7.2 Esempi di AIaaS

I seguenti sono esempi di AIaaS (ad aprile 2021):

- **IBM Watson Assistant**: È un chatbot AI che ha un prezzo diverso in base al numero di utenti attivi mensili.
- **Google Cloud AI e prodotti di ML**: Forniscono un'AI basata su documenti che include un parser di moduli e un OCR di documenti. I prezzi si basano sul numero di pagine inviate per l'elaborazione.
- **Amazon CodeGuru**: Fornisce una revisione del codice di ML Java che fornisce agli sviluppatori raccomandazioni per migliorare la qualità del loro codice. I prezzi si basano sul numero di linee di codice sorgente analizzato.
- **Microsoft Azure Cognitive Search**: Fornisce la ricerca cloud AI. I prezzi sono basati su unità di ricerca (definite in termini di storage e throughput utilizzati).

1.8 Modelli pre-addestrati

1.8.1 Introduzione ai modelli pre-addestrati

Addestrare modelli di ML può essere costoso (vedi Capitolo 3). In primo luogo, i dati devono essere preparati e quindi il modello deve essere addestrato. La prima attività può consumare grandi quantità di risorse umane, mentre la seconda attività può consumare molte risorse di calcolo. Molte organizzazioni non hanno disponibilità di queste risorse.

Un'alternativa più economica e spesso più efficace consiste nell'utilizzare un modello pre-addestrato. Esso fornisce funzionalità simili al modello richiesto e viene utilizzato come base per la creazione di un nuovo modello che estende e/o focalizza le funzionalità del modello pre-addestrato. Tali modelli sono disponibili solo per un numero limitato di tecnologie, come le reti neurali e le foreste casuali.

Se è necessario un classificatore di immagini, questo potrebbe essere addestrato utilizzando il set di dati ImageNet disponibile al pubblico, che contiene oltre 14 milioni di immagini classificate in oltre 1000 categorie. Ciò riduce il rischio di consumare risorse significative senza alcuna garanzia di successo. In alternativa, si potrebbe riutilizzare un modello esistente che è già stato addestrato su questo set di dati. Utilizzando un tale modello pre-addestrato, i costi di addestramento vengono risparmiati e il rischio che questo non funzioni è in gran parte eliminato.

Quando un modello pre-addestrato viene utilizzato senza modifiche, può essere semplicemente integrato nel sistema basato su AI, oppure può essere utilizzato in modalità As a Service (vedi Sezione 1.7).

© International Software Testing Qualifications Board

1.8.2 Transfer Learning

È anche possibile prendere un modello pre-addestrato e modificarlo per soddisfare un secondo, diverso requisito. Questo è noto come transfer learning e viene utilizzato su reti neurali profonde in cui i primi strati (vedi Capitolo 6) della rete neurale svolgono in genere compiti abbastanza basilari (ad esempio, identificare la differenza tra linee rette e curve in un classificatore di immagini), mentre gli strati successivi svolgono compiti più specializzati (ad esempio, differenziare tra tipi architettonici di edifici). In questo esempio, tutti i livelli tranne gli ultimi di un classificatore di immagini possono essere riusati, eliminando così la necessità di addestrare i primi livelli. I livelli successivi vengono poi riaddestrati per gestire i requisiti unici per un nuovo classificatore. Nella pratica, il modello pre-addestrato può essere perfezionato con un ulteriore addestramento su nuovi dati specifici del problema.

L'efficacia di questo approccio dipende in gran parte dalla somiglianza tra la funzione svolta dal modello originale e quella richiesta dal nuovo modello. Ad esempio, modificare un classificatore di immagini che identifica le specie di gatti affinchè identifichi le razze canine sarebbe molto più efficace che modificarlo affichè identifichi gli accenti delle persone.

Ci sono molti modelli pre-addestrati disponibili, forniti specialmente da ricercatori accademici. Alcuni esempi di tali modelli pre-addestrati sono i modelli ImageNet [R41] come Inception, VGG, AlexNet, e MobileNet per la classificazione delle immagini e modelli NLP pre-addestrati come BERT di Google [R51].

1.8.3 Rischi dell'utilizzo di modelli pre-addestrati e del transfer learning

L'utilizzo di modelli pre-addestrati e del transfer learning sono entrambi approcci diffusi alla creazione di sistemi basati su AI, ci sono però alcuni rischi associati. Questi includono:

- Un modello pre-addestrato potrebbe mancare di **trasparenza** rispetto a un modello generato internamente.
- Il **livello di somiglianza** tra la funzione svolta dal modello pre-addestrato e la funzionalità richiesta potrebbe non essere sufficiente. Inoltre, questa differenza potrebbe non essere compresa dal data scientist.
- Le **differenze tra le fasi di preparazione dei dati** (vedi Sezione 4.1), utilizzate per il modello pre-addestrato quando originariamente sviluppato, e le fasi di preparazione dei dati utilizzate quando questo modello viene poi utilizzato in un nuovo sistema, possono influire sulle prestazioni funzionali risultanti.
- È probabile che le **carenze** di un modello pre-addestrato vengano ereditate da coloro che lo riusano e potrebbero non essere documentate. Per esempio, i bias ereditati (vedi Sezione 2.4) potrebbero non essere evidenti se manca la documentazione sui dati utilizzati per addestrare il modello. Inoltre, se il modello pre-addestrato non è largamente utilizzato, è probabile che vi siano più difetti sconosciuti (o non documentati) e potrebbe essere necessario un testing più rigoroso per mitigare questo rischio.
- È molto probabile che i modelli creati attraverso il transfer learning siano sensibili alle stesse **vulnerabilità** del modello pre-addestrato su cui si basa (ad esempio, adversarial attacks come spiegato in 9.1.1). Inoltre, se è noto che un sistema basato su AI contiene un modello pre-addestrato specifico (o si basa su un modello pre-addestrato specifico), le vulnerabilità ad esso associate possono essere già note ai potenziali attacker.

Si noti che molti dei rischi di cui sopra possono essere mitigati più facilmente avendo a disposizione una documentazione accurata per il modello pre-addestrato (vedi Sezione 7.5).

© International Software Testing Qualifications Board

1.9 Standard, regolamenti e AI

Il Joint Technical Committee di IEC e ISO sulla tecnologia dell'informazione (ISO/IEC JTC1) prepara standard internazionali che contribuiscono all'AI. Ad esempio, nel 2017 è stato istituito un sottocomitato sull'AI (ISO/IEC JTC 1/SC42). Inoltre, ISO/IEC JTC1/SC7, che si occupa di ingegneria del software e dei sistemi, ha pubblicato un report tecnico sul "Testing of AI-based systems" [S01].

Gli standard sull'AI sono pubblicati anche a livello regionale (ad esempio, standard europei) e a livello nazionale.

Il General Data Protection Regulation (GDPR) a livello dell'UE è entrato in vigore nel maggio 2018 e stabilisce obblighi per i responsabili del trattamento dei dati in relazione ai dati personali e al processo decisionale automatizzato [B06]. Include requisiti per valutare e migliorare le prestazioni funzionali del sistema di AI, compresa la mitigazione di potenziali discriminazioni, e per garantire i diritti delle persone a non essere soggetti a processi decisionali automatizzati. L'aspetto più importante del GDPR, dal punto di vista del testing, è che i dati personali (comprese le previsioni) dovrebbero essere accurati. Ciò non significa che ogni singola previsione fatta dal sistema debba essere accurata, ma che il sistema dovrebbe essere sufficientemente accurato per gli scopi per cui viene utilizzato.

L'ente nazionale tedesco per gli standard (DIN) ha anche sviluppato l'AI Quality Metamodel ([S02], [S03]).

Gli standard sull'AI sono pubblicati anche da organismi del settore. Ad esempio, l'Institute of Electrical and Electronics Engineers (IEEE) sta lavorando su una serie di standard sull'etica e sull'AI (The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems). Molti di questi standard sono ancora in sviluppo al momento della stesura.

Laddove l'AI viene utilizzata nei sistemi legati alla sicurezza, sono applicabili gli standard normativi pertinenti, come ISO 26262 [S04] e ISO/PAS 21448 (SOTIF) [S05] per i sistemi automotive. Tali standard normativi sono in genere imposti dagli enti governativi, e per alcuni paesi risulta illegale commercializzare un'auto operante con software non conforme alla ISO 26262. Gli standard isolati sono documenti volontari, per cui il loro utilizzo è normalmente reso obbligatorio solo dalla legislazione o dal contratto. Tuttavia, chi si adegua agli standard lo fa per beneficiare dell'esperienza degli autori e quindi per creare prodotti di qualità superiore.

© International Software Testing Qualifications Board

2 Caratteristiche di qualità per i sistemi basati su AI - 105 minuti

Parole chiave

Nessuna

Parole chiave specifiche per l'AI

Adattabilità, bias algoritmico, autonomia, bias, evoluzione, spiegabilità, explainable AI (XAI), flessibilità, bias inappropriato, interpretabilità, sistema di ML, machine learning, reward hacking, robustezza, bias di campionamento, sistema di autoapprendimento, effetti collaterali, trasparenza

Obiettivi di apprendimento per il Capitolo 2:

2.1 Flessibilità e adattabilità

AI-2.1.1 K2 Spiegare l'importanza della flessibilità e dell'adattabilità come caratteristiche dei sistemi basati su AI.

2.2 Autonomia

AI-2.2.1 K2 Spiegare la relazione tra autonomia e sistemi basati su AI.

2.3 Evoluzione

AI-2.3.1 K2 Spiegare l'importanza di gestire l'evoluzione per i sistemi basati su AI.

2.4 Bias

AI-2.4.1 K2 Descrivere le diverse cause e tipi di bias che si trovano nei sistemi basati su AI.

2.5 Etica

AI-2.5.1 K2 Discutere i principi etici che dovrebbero essere rispettati nello sviluppo, nella distribuzione e nell'uso di sistemi basati su AI.

2.6 Effetti collaterali e reward hacking

AI-2.6.1 K2 Spiegare il verificarsi di effetti collaterali e di reward hacking nei sistemi basati su AI.

2.7 Trasparenza, interpretabilità e spiegabilità

AI-2.7.1 K2 Spiegare come la trasparenza, l'interpretabilità e la spiegabilità si applicano ai sistemi basati su AI.

2.8 Salvaguardia della salute (safety) e AI

AI-2.8.1 K1 Ricordare le caratteristiche che rendono difficile l'uso di sistemi basati su AI in applicazioni legate alla safety.

© International Software Testing Qualifications Board

2.1 Flessibilità e adattabilità

Flessibilità e adattabilità sono caratteristiche di qualità strettamente correlate. In questo syllabus, la **flessibilità** è considerata la capacità del sistema di essere utilizzato in situazioni che non facevano parte dei requisiti originali del sistema, mentre l'**adattabilità** è considerata la facilità con cui il sistema può essere modificato per nuove situazioni, come hardware diversi e ambienti operativi mutevoli.

Sia la flessibilità che l'adattabilità sono utili se:

- l'ambiente operativo non è completamente noto quando il sistema viene distribuito.
- il sistema dovrebbe far fronte a nuovi ambienti operativi.
- il sistema dovrebbe adattarsi a nuove situazioni.
- il sistema deve determinare quando dovrebbe cambiare il suo comportamento.

Ci si aspetta che i sistemi di autoapprendimento basati su AI dimostrino tutte le caratteristiche di cui sopra. Di conseguenza, devono sia essere adattabili che possedere il potenziale per essere flessibili.

I requisiti di flessibilità e adattabilità di un sistema basato su AI dovrebbero dettagliare le eventuali modifiche ambientali a cui il sistema dovrebbe adattarsi. Questi requisiti dovrebbero anche specificare i vincoli sul tempo e sulle risorse che il sistema può utilizzare per adattarsi (ad esempio, quanto tempo può impiegare per adattarsi al riconoscimento di un nuovo tipo di oggetto).

2.2 Autonomia

Quando si definisce l'autonomia, è importante riconoscere innanzitutto che un sistema completamente autonomo sarebbe completamente indipendente dalla supervisione e dal controllo umano. Nella pratica, spesso non si desidera la piena autonomia. Ad esempio, le auto a guida completamente autonoma, comunemente denominate "autonome", sono ufficialmente classificate come dotate di "guida autonoma completa" [B07].

Molti considerano i sistemi autonomi come "intelligenti" o "smart", il che suggerisce che includano componenti basati su AI per svolgere determinate funzioni. Ad esempio, i veicoli autonomi chiamati ad essere consapevoli della situazione che li circonda, in genere utilizzano diversi sensori e l'elaborazione delle immagini per raccogliere informazioni sull'ambiente circostante il veicolo. Il machine learning, e in particolare il deep learning (vedi Sezione 6.1), si è rivelato l'approccio più efficace per svolgere questa funzione. I sistemi autonomi possono comprendere anche funzioni decisionali e di controllo. Entrambi possono essere eseguiti in modo efficace utilizzando componenti basati su AI.

Anche se alcuni sistemi basati su AI sono considerati autonomi, questo non è vero per tutti i sistemi basati su AI. In questo syllabus, l'autonomia è considerata la capacità del sistema di funzionare indipendentemente dalla supervisione e dal controllo umano per periodi di tempo prolungati. Questo può aiutare a identificare le caratteristiche di un sistema autonomo che devono essere specificate e testate. Ad esempio, è necessario conoscere il periodo di tempo in cui un sistema autonomo dovrebbe funzionare in modo soddisfacente senza l'intervento umano. Inoltre, è importante identificare gli eventi per i quali il sistema autonomo deve restituire il controllo ai suoi controllori umani.

2.3 Evoluzione

In questo syllabus, l'evoluzione è considerata la capacità del sistema di migliorarsi in risposta a vincoli esterni mutevoli. Alcuni sistemi di AI possono essere classificati come di autoapprendimento e i

© International Software Testing Qualifications Board

sistemi di autoapprendimento basati su AI di successo devono includere questa capacità di evoluzione.

I sistemi basati su AI operano spesso in un ambiente in evoluzione. Come con altre forme di sistemi IT, un sistema basato su AI deve essere sufficientemente flessibile e adattabile per far fronte ai cambiamenti nel suo ambiente operativo.

I sistemi di autoapprendimento basati su AI, in genere, devono gestire due **forme di cambiamento**:

- una dove il sistema impara dalle proprie decisioni e dalle sue interazioni con il suo ambiente.
- un'altra in cui il sistema impara dalle modifiche apportate all'ambiente operativo del sistema.

In entrambi i casi il sistema si evolverà idealmente per migliorarne l'efficacia e l'efficienza. Tuttavia, questa evoluzione deve essere vincolata per evitare che il sistema sviluppi caratteristiche indesiderate. Qualsiasi evoluzione deve continuare a soddisfare i requisiti e i vincoli originali del sistema. Laddove questi manchino, il sistema deve essere gestito per assicurare che qualsiasi evoluzione rimanga entro i limiti e che rimanga sempre allineato con i valori umani. La Sezione 2.6 fornisce esempi relativi all'impatto degli effetti collaterali e del reward hacking sui sistemi di autoapprendimento basati su AI.

2.4 Bias

Nel contesto dei sistemi basati su AI, il bias è una misura statistica della distanza tra gli output forniti dal sistema e quelli che sono considerati "output equi" che non mostrano favoritismi per un particolare gruppo. I bias inappropriati possono essere collegati ad attributi come sesso, razza, etnia, orientamento sessuale, livello di reddito ed età. Sono stati segnalati casi di bias inappropriati nei sistemi basati su AI, ad esempio, nei sistemi utilizzati per formulare raccomandazioni per i prestiti bancari, nei sistemi di reclutamento e nei sistemi di monitoraggio giudiziario.

I bias possono essere introdotti in molti tipi di sistemi basati su AI. Ad esempio, è difficile evitare che l'expert bias (condizionamento degli esperti) sia incorporato nelle regole applicate da un sistema esperto. Tuttavia, la prevalenza dei sistemi di ML significa che gran parte della discussione relativa ai bias avviene nel contesto di questi sistemi.

I sistemi di ML vengono utilizzati per prendere decisioni e previsioni, utilizzando algoritmi che fanno uso di dati raccolti.Queste due componenti possono introdurre bias nei risultati:

- Il **bias algoritmico** può verificarsi quando l'algoritmo di apprendimento è configurato in modo errato (Es. sopravvalutazione di alcuni dati rispetto ad altri). Questa fonte di bias può essere causata e gestita dal tuning degli iperparametri degli algoritmi di ML (vedi Sezione 3.2).
- Il **bias di campionamento** può verificarsi quando i dati di addestramento non sono completamente rappresentativi dello spazio dati a cui viene applicato il ML.

Il bias inappropriato è spesso causato da un bias di campionamento, ma occasionalmente può anche essere causato da un bias algoritmico.

2.5 Etica

L'etica è definita nel Cambridge Dictionary come:

> *"Un sistema di credenze accettate che controllano il comportamento, in particolare un tipo di sistema basato sulla morale"*

© International Software Testing Qualifications Board

I sistemi basati su AI con capacità avanzate, stanno avendo un effetto ampiamente positivo sulla vita delle persone. Poiché questi sistemi sono diventati più diffusi, sono state sollevate preoccupazioni circa l'utilizzo delle stesse in modo etico.

Ciò che è considerato etico può cambiare nel tempo e può anche variare tra località e culture. È necessario prestare attenzione affinché l'implementazione di un sistema basato su AI da una località all'altra tenga conto delle differenze nei valori degli stakeholder.

Le politiche nazionali e internazionali sull'etica dell'AI possono essere trovate in molti paesi e regioni. La Organisation for Economic Co-operation and Development ha emesso nel 2019 i suoi principi per l'AI, ovvero i primi standard internazionali concordati dai governi per lo sviluppo responsabile dell'AI, [B08]. Questi principi, quando sono stati emanati. sono stati adottati da 42 paesi e sono sostenuti anche dalla Commissione Europea. I principi includono raccomandazioni politiche pratiche e principi basati sui valori per la "gestione responsabile di un'AI affidabile". Questi sono riassunti come segue:

- L'AI dovrebbe dare benefici alle persone e al pianeta favorendo: la crescita inclusiva, lo sviluppo sostenibile e il benessere.
- I sistemi di AI dovrebbero rispettare lo stato di diritto, i diritti umani, i valori democratici e la diversità, e dovrebbero includere garanzie adeguate a assicurare una società equa.
- Ci dovrebbe essere trasparenza sull'AI al fine di garantire che le persone comprendano i risultati e possano quindi contestarli.
- I sistemi di AI devono funzionare in modo solido, sicuro e protetto per tutto il loro ciclo di vita .I rischi dovrebbero essere valutati continuamente.
- Le organizzazioni e gli individui che sviluppano, distribuiscono o gestiscono sistemi di AI dovrebbero essere ritenuti responsabili.

2.6 Effetti collaterali e reward hacking

Gli effetti collaterali e il reward hacking possono far sì che i sistemi basati su AI generino risultati inaspettati e persino dannosi quando il sistema tenta di raggiungere i suoi obiettivi [B09].

Gli effetti collaterali negativi possono verificarsi quando il progettista di un sistema basato su AI specifica un obiettivo che "si concentra sull'esecuzione di alcuni compiti specifici nell'ambiente ma ignora altri aspetti dello stesso (potenzialmente molto ampio), e quindi esprime implicitamente indifferenza per le variabili ambientali che potrebbero effettivamente essere dannose da cambiare" [B09]. Ad esempio, un'auto a guida autonoma con l'obiettivo di viaggiare verso la sua destinazione nel "modo più efficiente in termini di consumo di carburante e sicuro possibile" può raggiungere l'obiettivo, ma con l'effetto collaterale di infastidire i passeggeri a causa dell'eccessivo tempo impiegato per il viaggio.

Il reward hacking può verificarsi in un sistema basato su AI che raggiunge un obiettivo specifico utilizzando una soluzione "intelligente" o "facile" la quale "perverte lo spirito dell'intento del progettista". In effetti, l'obiettivo può essere aggirato. Un esempio ampiamente utilizzato di reward hacking è quando un sistema basato su AI sta imparando a giocare ad un gioco arcade per computer. Viene presentato con l'obiettivo di ottenere il "punteggio più alto", e per farlo hackera semplicemente il record di dati che memorizza il punteggio più alto, piuttosto che giocare per raggiungerlo.

© International Software Testing Qualifications Board

2.7 Trasparenza, interpretabilità e spiegabilità

I sistemi basati su AI vengono in genere applicati in aree in cui gli utenti devono fidarsi di tali sistemi. Questo può avvenire per motivi di sicurezza, ma anche quando è necessaria la privacy e dove potrebbero fornire previsioni e decisioni potenzialmente in grado di cambiare la vita.

I sistemi basati su AI vengono presentati alla maggior parte degli utenti come "scatole nere" per cui hanno poca consapevolezza di come questi sistemi arrivino ai loro risultati. In alcuni casi, questa ignoranza può sussistere anche per i data scientist che hanno costruito i sistemi. Occasionalmente, gli utenti potrebbero non essere nemmeno consapevoli di interagire con un sistema basato su AI.

La complessità intrinseca dei sistemi basati su AI ha portato al campo della "Explainable AI" (XAI). L'obiettivo della XAI è che gli utenti siano in grado di capire come i sistemi basati su AI ottengono i loro risultati, aumentando così la fiducia degli utenti in essi.

Secondo The Royal Society [B10], ci sono diversi motivi per volere la XAI, tra cui:

- dare agli utenti fiducia nel sistema
- salvaguardia contro i bias
- soddisfare gli standard normativi o i requisiti delle linee di condotta
- migliorare la progettazione del sistema
- valutare il rischio, la robustezza e la vulnerabilità
- comprendere e verificare gli output di un sistema
- autonomia, rappresentanza (far sentire l'utente autorizzato) e soddisfare i valori sociali

Ciò porta alle seguenti tre caratteristiche XAI desiderabili di base per i sistemi basati su AI dal punto di vista di uno stakeholder (vedi anche Sezione 8.6):

- **Trasparenza**: È considerata la facilità con cui è possibile determinare l'algoritmo e i dati di addestramento utilizzati per generare il modello.
- **Interpretabilità**: È considerata la comprensibilità della tecnologia di AI da parte di vari stakeholder, inclusi gli utenti.
- **Spiegabilità**: È considerata la facilità con cui gli utenti possono determinare in che modo il sistema basato su AI ottiene un particolare risultato.

2.8 Safety e AI

In questo syllabus, la safety (salvaguardia della salute) è considerata l'aspettativa che un sistema basato su AI non causi danni a persone, proprietà o ambiente. I sistemi basati su AI possono essere utilizzati per prendere decisioni che influiscono sulla safety. Ad esempio, i sistemi basati su AI che operano nei settori della medicina, della produzione, della difesa, della sicurezza e dei trasporti hanno il potenziale per influire sulla safety.

Le caratteristiche dei sistemi basati su AI che rendono più difficile garantire che siano sicuri (ad esempio, che non danneggino gli esseri umani) includono:

- complessità
- non determinismo
- natura probabilistica

© International Software Testing Qualifications Board

- autoapprendimento
- mancanza di trasparenza, interpretabilità e spiegabilità
- mancanza di robustezza

Le sfide del testing di molte di queste caratteristiche sono trattate nel Capitolo 8.

© International Software Testing Qualifications Board

3 Machine Learning (ML) - Panoramica - 145 minuti

Parole chiave

Nessuna

Parole chiave specifiche per l'AI

Associazione, classificazione, clustering, preparazione dei dati, algoritmo di ML, framework di ML, criteri delle prestazioni funzionali del ML, modello di ML, dati di addestramento ML, flusso di lavoro di ML, valutazione del modello, tuning del modello, valore anomalo, overfitting, regressione, apprendimento per rinforzo, apprendimento supervisionato, underfitting, apprendimento non supervisionato

Obiettivi di apprendimento per il Capitolo 3:

3.1 Forme di ML

AI-3.1.1 K2 Descrivere la classificazione e la regressione come parte dell'apprendimento supervisionato.

AI-3.1.2 K2 Descrivere il clustering e l'associazione come parte dell'apprendimento non supervisionato.

AI-3.1.3 K2 Descrivere l'apprendimento per rinforzo.

3.2 Flusso di lavoro di ML

AI-3.2.1 K2 Riassumere il flusso di lavoro utilizzato per creare un sistema di ML.

3.3 Selezionare una forma di ML

AI-3.3.1 K3 Dato uno scenario di progetto, identificare una forma appropriata di ML (tra classificazione, regressione, clustering, associazione o apprendimento per rinforzo).

3.4 Fattori coinvolti nella selezione dell'algoritmo di ML

AI-3.4.1 K2 Spiegare i fattori coinvolti nella selezione degli algoritmi di ML.

3.5 Overfitting e Underfitting

AI-3.5.1 K2 Riassumere i concetti di underfitting e overfitting.

HO-3.5.1 H0 Dimostrare l'underfitting e l'overfitting.

© International Software Testing Qualifications Board

3.1 Forme di ML

Gli algoritmi di ML possono essere categorizzati come:

- apprendimento **supervisionato**,
- apprendimento **non supervisionato**
- apprendimento **per rinforzo**.

3.1.1 Apprendimento supervisionato

In questo tipo di apprendimento, l'algoritmo crea il modello di ML dai dati etichettati durante la fase di addestramento. I dati etichettati, che in genere comprendono coppie di input (ad esempio, un'immagine di un cane e l'etichetta "cane"), vengono utilizzati dall'algoritmo per dedurre la relazione tra i dati di input (ad esempio, immagini di cani) e le etichette di output (ad esempio, "cane" e "gatto") durante l'**addestramento**.
Durante la fase di **testing** del modello di ML, al modello addestrato viene applicato un nuovo insieme di dati invisibili per prevedere l'output.
Il modello viene **distribuito** una volta che il livello di accuratezza dell'output è considerato soddisfacente.

I problemi risolti dall'apprendimento supervisionato si dividono in due categorie:

- **Classificazione**: Quando il problema richiede che un input sia classificato in una delle poche classi predefinite, si usa la classificazione. Il riconoscimento facciale o il rilevamento di oggetti in un'immagine sono esempi di problemi che utilizzano la classificazione.
- **Regressione**: È quando il problema richiede al modello di ML di prevedere un output numerico utilizzando la regressione. La previsione dell'età di una persona in base ai dati di input sulle sue abitudini o la previsione dei prezzi futuri delle azioni sono esempi di problemi che utilizzano la regressione.

Si noti che il termine regressione, usato nel contesto di un problema di ML, è diverso dal suo uso in altri sillabi ISTQB®, come [I01], dove la regressione è usata per descrivere il problema delle modifiche al software che causano difetti legati al cambiamento.

3.1.2 Apprendimento non supervisionato

In questo tipo di apprendimento, l'algoritmo crea il modello di ML da dati non etichettati durante la fase di addestramento. I dati senza etichetta vengono utilizzati dall'algoritmo per dedurre i modelli nei dati di input durante l'**addestramento** e assegna gli input a classi diverse, in base alle loro caratteristiche comuni.
Durante la fase di **testing**, il modello addestrato viene applicato a un nuovo insieme di dati invisibili per prevedere a quali classi dovrebbero essere assegnati i dati di input.
Il modello viene **distribuito** una volta che il livello di accuratezza dell'output è considerato soddisfacente.

I problemi risolti dall'apprendimento non supervisionato si dividono in due categorie:

- **Clustering**: È quando il problema richiede l'identificazione di somiglianze nei punti di dati di input che consentono di raggrupparli in base a caratteristiche o attributi comuni. Ad esempio, il clustering viene utilizzato per classificare diversi tipi di clienti ai fini di marketing.

© International Software Testing Qualifications Board

- **Associazione**: È quando il problema richiede l'identificazione di relazioni o dipendenze interessanti tra gli attributi dei dati. Ad esempio, un sistema di raccomandazione dei prodotti può identificare associazioni basate sul comportamento di acquisto dei clienti.

3.1.3 Apprendimento per rinforzo

L'apprendimento per rinforzo è un approccio in cui il sistema (un "agente intelligente") apprende interagendo con l'ambiente in modo iterativo e quindi impara dall'esperienza. L'apprendimento per rinforzo non utilizza i dati di **addestramento**. L'agente viene premiato quando prende una decisione corretta e penalizzato quando prende una decisione sbagliata.

La creazione dell'ambiente, la scelta della giusta strategia per l'agente per raggiungere l'obiettivo desiderato e la progettazione di una funzione di ricompensa sono sfide chiave quando si implementa l'apprendimento per rinforzo. Robotica, veicoli autonomi e chatbot sono esempi di applicazioni che utilizzano l'apprendimento per rinforzo.

3.2 Flusso di lavoro di ML

Le attività nel flusso di lavoro di machine learning sono:

Comprendere gli obiettivi

Lo scopo del modello di ML da implementare deve essere compreso e concordato con gli stakeholder per garantire l'allineamento con le priorità aziendali. I criteri di accettazione (comprese le metriche delle prestazioni funzionali del ML - vedi Capitolo 5) dovrebbero essere definiti per il modello sviluppato.

Selezionare un framework

Dovrebbe essere selezionato un framework di sviluppo dell'AI adatto in base agli obiettivi, ai criteri di accettazione e alle priorità aziendali (vedi Sezione 1.5).

Selezionare e costruire l'algoritmo

Un algoritmo di ML viene selezionato in base a vari fattori inclusi gli obiettivi, i criteri di accettazione e i dati disponibili (vedi Sezione 3.4). L'algoritmo può essere codificato manualmente, ma spesso viene recuperato da una libreria di codice pre-scritto. L'algoritmo viene quindi compilato per preparare l'addestramento del modello, se necessario.

Preparare e testare i dati

La preparazione dei dati (vedi Sezione 4.1) comprende l'acquisizione dei dati, la pre-elaborazione dei dati e l'ingegneria delle caratteristiche. Parallelamente a queste attività può essere eseguita l'Exploratory Data Analysis (EDA).

I dati utilizzati dall'algoritmo e dal modello saranno basati sugli obiettivi e vengono utilizzati da tutte le attività nell'attività di "generazione e testing del modello" mostrata nella Figura 1. Ad esempio, se il sistema è un sistema di trading in tempo reale, i dati proverranno dal mercato del trading.

I dati utilizzati per addestrare, regolare e testare il modello devono essere rappresentativi dei dati operativi che verranno utilizzati dal modello. In alcuni casi, è possibile utilizzare set di dati pre-raccolti per l'addestramento iniziale del modello (ad esempio, vedi i set di dati Kaggle [R61]). In caso contrario, i dati grezzi richiedono in genere una fase minima di pre-elaborazione e di ingegneria delle caratteristiche.

© International Software Testing Qualifications Board

È necessario eseguire il testing dei dati e di tutte le fasi di preparazione dei dati automatizzate. Vedi la Sezione 7.2.1 per maggiori dettagli sul testing dei dati di input.

Addestrare il modello

L'algoritmo di ML selezionato utilizza i dati di addestramento per addestrare il modello.

Alcuni algoritmi, come quelli che generano una rete neurale, leggono più volte il set di dati di addestramento. Ogni iterazione di addestramento sul set di dati di addestramento viene definita “epoch”.

I parametri che definiscono la struttura del modello (ad esempio, il numero di layers di una rete neurale o la profondità di un albero delle decisioni) vengono passati all'algoritmo. Questi parametri sono noti come iperparametri del modello.

Anche i parametri che controllano l'addestramento (ad esempio, quante epoch utilizzare durante l’addestramento di una rete neurale) vengono passati all'algoritmo. Questi parametri sono noti come iperparametri dell'algoritmo.

Valutare il modello

Il modello viene valutato rispetto alle metriche delle prestazioni funzionali del ML concordate, utilizzando il set di dati di validazione, con i risultati poi utilizzati per migliorare (regolare) il modello. La valutazione e il tuning del modello dovrebbero assomigliare a un esperimento scientifico che deve essere condotto con attenzione in condizioni controllate con l’ausilio di una documentazione chiara. Nella pratica, diversi modelli vengono in genere creati e addestrati utilizzando algoritmi diversi (ad esempio, foreste casuali, SVM e reti neurali), e viene scelto il migliore, in base ai risultati della valutazione e del tuning.

Regolare il modello

I risultati della valutazione del modello rispetto alle metriche delle prestazioni funzionali del ML concordate, vengono utilizzati per regolare le impostazioni del modello per adattarle ai dati e quindi migliorarne le prestazioni. Il modello può essere regolato mediante:
- il tuning degli iperparametri in cui l'attività di addestramento viene modificata (ad esempio, modificando il numero di fasi di addestramento o modificando la quantità di dati utilizzati per l'addestramento),
- oppure vengono aggiornati gli attributi del modello (ad esempio, il numero di neuroni in una rete neurale o la profondità di un albero delle decisioni).

Le tre attività di addestramento, valutazione e tuning possono essere considerate come comprendenti la **Generazione del Modello**, come mostrato nella Figura 1.

Testare il modello

Una volta che un modello è stato generato (ovvero, è stato addestrato, valutato e ottimizzato) dovrebbe essere testato rispetto a un set di dati di test indipendente, per garantire che i criteri delle prestazioni funzionali del ML concordati siano soddisfatti (vedi Sezione 7.2.2). Le misure delle prestazioni funzionali del testing vengono anche confrontate con quelle della valutazione e se le prestazioni del modello con dati indipendenti sono significativamente inferiori rispetto alla valutazione, potrebbe essere necessario selezionare un modello diverso.

Oltre ai test delle prestazioni funzionali, devono essere eseguiti anche test non funzionali come il tempo per addestrare il modello e il tempo e l'utilizzo delle risorse necessari per fornire una previsione. Tipicamente, questi test vengono eseguiti dal data engineer/scientist, ma anche i tester con sufficiente conoscenza del dominio e accesso alle risorse rilevanti possono eseguire questi test.

© International Software Testing Qualifications Board

Distribuire il modello

Una volta completato lo sviluppo del modello, come mostrato nella Figura 1, il modello ottimizzato in genere deve essere riprogettato per la distribuzione insieme alle relative risorse, inclusa la pipeline di dati pertinente. Ciò si ottiene normalmente attraverso il framework. Gli obiettivi potrebbero includere i sistemi embedded e il cloud, in cui è possibile accedere al modello tramite Web API.

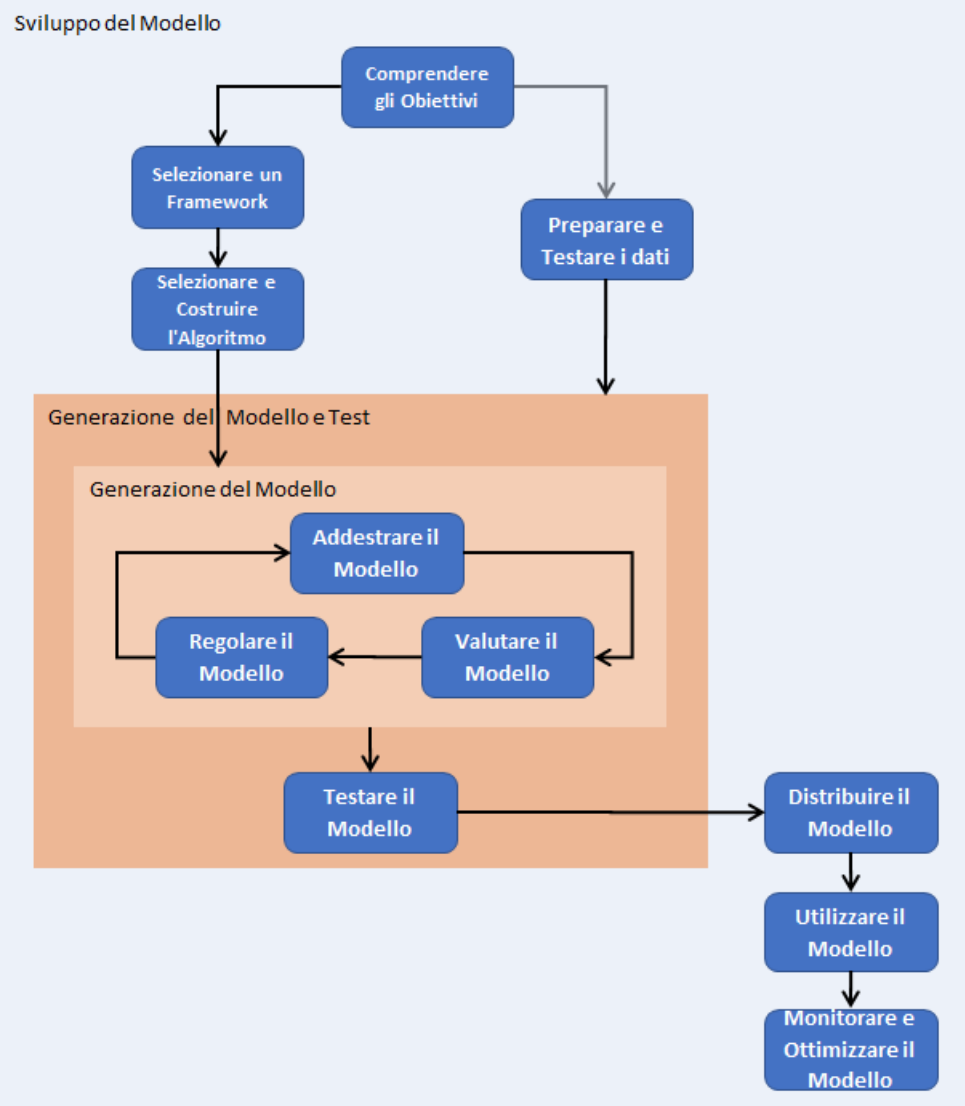

Figura 1: Flusso di lavoro di ML

Utilizzare il modello

© International Software Testing Qualifications Board

Una volta distribuito, il modello può essere utilizzato operativamente ed è in genere parte di un più ampio sistema basato su AI . I modelli possono eseguire previsioni in batch programmate a intervalli di tempo prestabiliti o possono essere eseguiti su richiesta in tempo reale.

Monitorare e ottimizzare il modello

Il modello, durante l'uso, può evolvere allontanandosi dalle prestazioni previste (vedi Sezioni 2.3 e 7.6). Per garantire che ogni deriva sia identificata e gestita, il modello operativo dovrebbe essere regolarmente valutato rispetto ai suoi criteri di accettazione.

Potrebbe essere necessario aggiornare le impostazioni del modello per affrontare il problema della deriva, oppure si potrebbe decidere che sia necessario un riaddestramento con nuovi dati per creare un modello più accurato o più robusto. In questo caso è possibile creare e addestrare un nuovo modello con dati di addestramento aggiornati. Il nuovo modello può quindi essere confrontato con il modello esistente utilizzando una forma di testing A/B (vedi Sezione 9.4).

Il flusso di lavoro di ML mostrato nella Figura 1 è una sequenza logica. Nella pratica, il flusso di lavoro viene applicato in maniera tale per cui le fasi vengono ripetute in modo iterativo (ad esempio, quando il modello viene valutato, è spesso necessario tornare alla fase di addestramento, e talvolta alla preparazione dei dati).

Le fasi mostrate nella Figura 1 non includono l'integrazione del modello di ML con le parti non-ML del sistema complessivo. In genere, i modelli di ML non possono essere distribuiti in modo isolato bensì devono essere integrati con le parti non-ML. Ad esempio, nelle applicazioni di visione artificiale, esiste una pipeline di dati che pulisce e modifica i dati prima di inviarli al modello di ML. Laddove il modello faccia parte di un più ampio sistema basato su AI, dovrà essere integrato in questo sistema prima della distribuzione. In questo caso, possono essere eseguiti livelli di test di integrazione, di sistema e di accettazione, come descritto nella Sezione 7.2.

3.3 Selezionare una forma di ML

Quando si seleziona un approccio di ML appropriato, si applicano le seguenti linee guida:

- Dovrebbero essere disponibili dati di addestramento e di test sufficienti per l'approccio di ML selezionato.
- Per l'**apprendimento supervisionato** è necessario disporre di dati adeguatamente etichettati.
- Se è presente un'etichetta di output, potrebbe trattarsi di **apprendimento supervisionato**.
- Se l'output è discreto e categorico, potrebbe trattarsi di **classificazione**.
- Se l'output è di natura numerica e continua, potrebbe trattarsi di **regressione**.
- Se non viene fornito alcun output nel set di dati fornito, potrebbe trattarsi di **apprendimento non supervisionato**.
- Se il problema riguarda il raggruppamento di dati simili, potrebbe trattarsi di **clustering**.
- Se il problema riguarda la ricerca di elementi di dati che si verificano contemporaneamente, potrebbe trattarsi di **associazione**.
- L'**apprendimento per rinforzo** è più adatto a contesti in cui vi è interazione con l'ambiente.

© International Software Testing Qualifications Board

- Se il problema riguarda la nozione di stati multipli e implica decisioni in ciascuno stato, allora può essere applicabile l'**apprendimento per rinforzo**.

3.4 Fattori coinvolti nella selezione dell'algoritmo di ML

Non esiste un approccio definitivo alla selezione dell'algoritmo di ML ottimale, delle impostazioni del modello di ML e degli iperparametri del modello di ML. Nella pratica, questo set viene scelto in base a un mix dei seguenti fattori:

- La **funzionalità** richiesta (ad esempio, se la funzionalità è la classificazione o la previsione di un valore discreto)
- Le **caratteristiche di qualità** richieste; come ad esempio
  - accuratezza (ad esempio, alcuni modelli possono essere più accurati, ma più lenti)
  - vincoli sulla memoria disponibile (ad esempio, per un sistema embedded)
  - velocità di addestramento (e riaddestramento) del modello
  - velocità di previsione (ad esempio, per i sistemi in tempo reale)
  - requisiti di trasparenza, interpretabilità e spiegabilità
- Il **tipo di dati** disponibili per l'addestramento del modello (ad esempio, alcuni modelli potrebbero funzionare solo con dati di immagini)
- La **quantità di dati** disponibili per l'addestramento e il testing del modello (alcuni modelli potrebbero, ad esempio, avere la tendenza ad un overfit con una quantità limitata di dati, in misura maggiore rispetto ad altri modelli)
- Il numero di **caratteristiche nei dati** di input che dovrebbero essere utilizzate dal modello (ad esempio, è probabile che altri fattori, come la velocità e l'accuratezza, siano direttamente influenzati dal numero di caratteristiche)
- Il numero previsto di **classi per il clustering** (ad esempio, alcuni modelli potrebbero non essere adatti a problemi con più di una classe)
- **Esperienza precedente**
- **Trial** e **errore**

3.5 Overfitting e Underfitting

3.5.1 Overfitting

L'overfitting si verifica quando il modello si adatta troppo strettamente a un insieme di punti di dati e non riesce a generalizzare correttamente.
Un tale modello funziona molto bene con i dati utilizzati per addestrarlo, ma può avere difficoltà a fornire previsioni accurate per nuovi dati. L'overfitting può verificarsi quando il modello cerca di adattarsi ad ogni punto di dati, inclusi quei punti di dati che possono essere descritti come rumore o valori anomali. Può verificarsi anche quando vengono forniti dati insufficienti nel set di dati di addestramento.

© International Software Testing Qualifications Board

3.5.2 Underfitting

L'underfitting si verifica quando il modello non è sufficientemente sofisticato per adattarsi accuratamente ai modelli nei dati di addestramento.
I modelli underfitting tendono ad essere troppo semplicistici e possono avere difficoltà a fornire previsioni accurate sia per i nuovi dati che per dati molto simili a quelli di addestramento. Una delle cause dell'underfitting può essere un set di dati di addestramento che non contiene caratteristiche che riflettono importanti relazioni tra input e output. Può anche verificarsi quando l'algoritmo non si adatta correttamente ai dati (ad esempio, creando un modello lineare per dati non lineari).

3.5.3 Esercizio pratico: Dimostrare l'overfitting e l'underfitting

Dimostrate i concetti di overfitting e underfitting su un modello. Ciò potrebbe essere dimostrato utilizzando un set di dati che contiene pochissimi dati (overfitting), e un set di dati con scarse correlazioni delle caratteristiche (underfitting).

© International Software Testing Qualifications Board

4 ML - Dati - 230 minuti

Parole chiave

Nessuna

Parole chiave specifiche per l'AI

Annotazione, aumento, modello di classificazione, etichettatura dei dati, preparazione dei dati, dati di addestramento ML, apprendimento supervisionato, set di dati di test, set di dati di validazione

Obiettivi di apprendimento per il Capitolo 4:

4.1 Preparazione dei dati come parte del flusso di lavoro di ML

AI-4.1.1 K2 Descrivere le attività e le sfide relative alla preparazione dei dati.

HO-4.1.1 H2 Eseguire la preparazione dei dati a supporto della creazione di un modello di ML.

4.2 Set di dati di addestramento, validazione e test nel flusso di lavoro di ML

AI-4.2.1 K2 Contrastare l'uso di set di dati di addestramento, validazione e test nello sviluppo di un modello di ML.

HO-4.2.1 H2 Identificare i set di dati di addestramento e di test e creare un modello di ML.

4.3 Problemi di qualità dei set di dati

AI-4.3.1 K2 Descrivere i problemi tipici di qualità dei set di dati.

4.4 La qualità dei dati e il suo effetto sul modello di ML

AI-4.4.1 K2 Riconoscere come una scarsa qualità dei dati può causare problemi con il modello di ML risultante.

4.5 Etichettatura dei dati per l'apprendimento supervisionato

AI-4.5.1 K1 Ricordare i diversi approcci all'etichettatura dei dati nei set di dati per l'apprendimento supervisionato.

AI-4.5.2 K1 Ricordare i motivi per cui i dati nei set di dati sono stati etichettati erroneamente.

© International Software Testing Qualifications Board

4.1 Preparazione dei dati come parte del flusso di lavoro di ML

La **preparazione dei dati** utilizza in media il 43% dell'effort del flusso di lavoro di ML ed è probabilmente l'attività che richiede più risorse nel flusso di lavoro di ML. In confronto, la **selezione e la costruzione del modello** utilizzano solo il 17% [R71]. La preparazione dei dati fa parte della pipeline di dati, che acquisisce dati grezzi e produce dati in un formato che può essere utilizzato sia per addestrare un modello di ML sia per la previsione da parte di un modello di ML addestrato.

La preparazione dei dati può essere considerata come comprendente le seguenti attività:

Acquisizione dei dati

- **Identificazione**: Vengono identificati i tipi di dati da utilizzare per l'addestramento e le previsioni. Ad esempio, per un'auto a guida autonoma, potrebbe includere l'identificazione della necessità di dati radar, video e laser imaging, detection and ranging (LiDAR).
- **Raccolta**: Viene identificata la fonte dei dati e vengono determinati i mezzi per raccogliere i dati. Ad esempio, ciò potrebbe includere l'identificazione del Fondo Monetario Internazionale (FMI) come fonte di dati finanziari e i canali che verranno utilizzati per inviare i dati al sistema basato su AI.
- **Etichettatura**: Vedere la Sezione 4.5.

I dati acquisiti possono essere in varie forme (ad esempio, numerico, categorico, immagine, tabellare, testo, serie temporale, sensore, geospaziale, video e audio).

Pre-elaborazione dei dati

- **Pulizia**: Se vengono identificati dati errati, dati duplicati o valori anomali, questi vengono rimossi o corretti. Inoltre, l'imputazione dei dati può essere utilizzata per sostituire i valori dei dati mancanti con valori stimati o ipotizzati (ad esempio, utilizzando valori medi, mediani e modali). Può anche essere eseguita la rimozione o l'anonimizzazione delle informazioni personali.
- **Trasformazione**: Il formato dei dati forniti viene modificato (ad esempio, scomponendo un indirizzo contenuto come stringa nelle sue parti costituenti, eliminando un campo contenente un identificatore casuale, convertendo i dati categorici in dati numerici, modificando i formati delle immagini).
  Alcune delle trasformazioni applicate ai dati numerici includono il ridimensionamento per garantire che venga utilizzato lo stesso intervallo. La standardizzazione, ad esempio, ridimensiona i dati per garantire che assumano una media pari a zero e una deviazione standard di uno. Questa normalizzazione garantisce che i dati abbiano un intervallo compreso tra zero e uno.
- **Aumento**: Viene utilizzato per aumentare il numero di campioni in un set di dati. L'aumento può essere utilizzato anche per includere esempi contraddittori nei dati di addestramento, fornendo robustezza contro gli adversarial attacks (vedi 9.1).
- **Campionamento**: Comporta la selezione di una parte del set di dati totale disponibile in modo da poter osservare i modelli nel set di dati più ampio. Questo viene in genere fatto per ridurre i costi e il tempo necessario per creare il modello di ML.

Si noti che tutta la pre-elaborazione comporta il rischio che possa modificare dati validi utili o aggiungere dati non validi.

© International Software Testing Qualifications Board

Feature engineering

- **Selezione delle feature**: Una feature è un attributo/proprietà riflessa nei dati. La selezione delle feature implica la selezione di quelle che con maggiore probabilità contribuiranno all'addestramento e alla previsione del modello. Nella pratica, spesso include la rimozione delle feature che non si prevede (o non si desidera) abbiano alcun effetto sul modello risultante. Rimuovendo le informazioni irrilevanti (rumore), la selezione delle feature può ridurre i tempi complessivi di addestramento, prevenire l'overfitting (vedi Sezione 3.5.1), aumentare l'accuratezza e rendere i modelli più generalizzabili.
- **Estrazione delle feature**: Implica la derivazione di caratteristiche informative e non ridondanti dalle feature esistenti. Il set di dati risultante è in genere più piccolo e può essere utilizzato per generare un modello di ML di accuratezza equivalente in modo più economico e rapido.

Parallelamente a queste attività di preparazione dei dati, viene generalmente eseguita anche l'**Exploratory Data Analysis (EDA)** per supportare l'attività complessiva di preparazione dei dati. Ciò include l'esecuzione dell'analisi dei dati finalizzata a scoprire gli andamenti inerenti ai dati e l'utilizzo della visualizzazione dei dati per rappresentarli in un formato visivo tracciando l'andamento nei dati.

Sebbene le attività e le sotto-attività di preparazione dei dati di cui sopra siano state mostrate in un ordine logico, progetti diversi possono riordinarle o utilizzarne solo un sottoinsieme. Alcune delle fasi di preparazione dei dati, come l'identificazione della fonte dei dati, vengono eseguite una sola volta e possono essere eseguite manualmente. Altre fasi possono far parte della pipeline di dati operativi e normalmente funzionano su dati in tempo reale. Queste attività dovrebbero essere automatizzate.

4.1.1 Sfide nella preparazione dei dati

Alcune delle sfide relative alla preparazione dei dati includono:

- La necessità di conoscenza di:
  - dominio dell'applicazione.
  - dati e le loro proprietà.
  - varie tecniche associate alla preparazione dei dati.
- La difficoltà di ottenere dati di alta qualità da più fonti.
- La difficoltà di automatizzare la pipeline di dati e garantire che la pipeline di dati di produzione sia scalabile e abbia una ragionevole efficienza delle prestazioni (ad esempio, il tempo necessario per completare l'elaborazione di un elemento di dati).
- I costi associati alla preparazione dei dati.
- Non dare sufficiente priorità al controllo dei difetti introdotti nella pipeline di dati durante la preparazione dei dati.
- L'introduzione di bias di campionamento (vedi Sezione 2.4).

4.1.2 Esercizio pratico: Preparazione dei dati per il ML

Per un determinato insieme di dati grezzi, eseguite le fasi di preparazione dei dati applicabili come descritto nella Sezione 4.1 per produrre un set di dati che verrà utilizzato per creare un modello di classificazione utilizzando l'apprendimento supervisionato.

© International Software Testing Qualifications Board

Questa attività costituisce il primo passo nella creazione di un modello di ML che verrà utilizzato per gli esercizi futuri.

Per svolgere questa attività, agli studenti verranno forniti materiali appropriati (e specifici per la lingua), tra cui:

- Librerie
- Framework di ML
- Strumenti
- Un ambiente di sviluppo

4.2 Set di dati di addestramento, validazione e test nel flusso di lavoro di ML

Logicamente, per sviluppare un modello di ML sono necessari **tre set di dati equivalenti** (cioè selezionati casualmente da un singolo set di dati iniziale):

- Un set di **dati di addestramento** che viene utilizzato per addestrare il modello.
- Un set di **dati di validazione** utilizzato per la valutazione e il successivo tuning del modello.
- Un set di **dati di test** (noto anche come set di dati di resistenza), che viene utilizzato per testare il modello ottimizzato.

Se sono disponibili dati idonei illimitati, la **quantità di dati** utilizzata nel flusso di lavoro di ML per l'addestramento, la valutazione e il testing dipende in genere dai seguenti fattori:

- L'algoritmo utilizzato per addestrare il modello.
- La disponibilità di risorse, come RAM, spazio su disco, potenza di calcolo, larghezza di banda della rete e tempo disponibile.

Nella pratica, a causa della sfida di acquisire un numero sufficiente di dati adeguati, i set di dati di addestramento e di validazione sono spesso derivati da un unico set di dati combinato. Il set di dati di test viene mantenuto separato e non viene utilizzato durante l'addestramento. Questo per garantire che il modello sviluppato non sia influenzato dai dati di test e in modo che i risultati dei test riflettano fedelmente la qualità del modello.

Non esiste un rapporto ottimale per suddividere il set di dati combinato nei tre set di dati singoli, ma i rapporti tipici che possono essere utilizzati come linee guida vanno da 60:20:20 a 80:10:10 (addestramento: validazione: test). La suddivisione dei dati in questi set di dati viene spesso eseguita in modo casuale, a meno che il set di dati non sia piccolo o se vi sia il rischio che i set di dati risultanti non siano rappresentativi dei dati operativi previsti.

Se sono disponibili dati limitati, la suddivisione dei dati disponibili in tre set di dati può comportare una insufficiente disponibilità di dati per un addestramento efficace. Per ovviare a questo problema, i set di dati di addestramento e di validazione possono essere combinati (mantenendo separato il set di dati di test), e quindi utilizzati per creare più combinazioni divise di questo set di dati (ad esempio, 80% di addestramento / 20% di validazione). I dati vengono quindi assegnati in modo casuale ai set di dati di addestramento e di validazione. L'addestramento, la validazione e il tuning vengono eseguiti utilizzando queste combinazioni multiple di suddivisione per creare più modelli ottimizzati e le prestazioni complessive del modello possono essere calcolate come media di tutte le esecuzioni. Ci sono vari metodi usati per creare combinazioni multiple di split, che includono split-test, bootstrap, K-fold cross validation e leave-one-out cross validation (vedere [B02] per maggiori dettagli).

© International Software Testing Qualifications Board

4.2.1 Esercizio pratico: Identificare i dati di addestramento e di test e creare un modello di ML

Dividete i dati preparati in precedenza (vedi Sezione 4.1.2) in set di dati di addestramento, validazione e test.

Addestrate e testate un modello di classificazione utilizzando l'apprendimento supervisionato con questi set di dati.

Spiegate la differenza tra valutazione/tuning e testing confrontando l'accuratezza raggiunta con i set di dati di validazione e di test.

4.3 Problemi di qualità dei set di dati

Tipici problemi di qualità relativi ai dati in un set di dati includono, ma non sono limitati a, quelli mostrati nella tabella seguente:

Aspetto della qualità	**Descrizione**
Dati errati	I dati acquisiti erano errati (ad esempio, a causa di un sensore difettoso) o inseriti in modo errato (ad esempio, errori di copia-incolla).
Dati incompleti	I valori dei dati potrebbero mancare (ad esempio, un campo in un record potrebbe essere vuoto o i dati per un particolare intervallo di tempo potrebbero essere stati omessi). Possono esserci vari motivi per l'incompletezza dei dati, quali problemi di sicurezza, problemi hardware ed errori umani.
Dati etichettati erroneamente	Esistono diversi possibili motivi per cui i dati vengono etichettati in modo errato (vedi Sezione 4.5.2).
Dati insufficienti	Non sono disponibili dati sufficienti affinché i modelli possano essere riconosciuti dagli algoritmi di apprendimento in uso (si noti che la quantità minima di dati richiesta varierà per algoritmi diversi).
Dati non pre-elaborati	I dati dovrebbero essere pre-elaborati per garantire che siano puliti, in un formato coerente e non contengano valori anomali indesiderati (vedi Sezione 4.1).
Dati obsoleti	I dati utilizzati sia per l'apprendimento che per la previsione dovrebbero essere il più aggiornati possibile (ad esempio, l'utilizzo di dati finanziari di diversi anni fa potrebbe portare a risultati imprecisi).
Dati sbilanciati	Dati sbilanciati possono derivare da: - bias inappropriati (ad esempio, basati su razza, sesso o etnia) - posizionamento inadeguato dei sensori (ad esempio, telecamere di riconoscimento facciale posizionate all'altezza del soffitto)

© International Software Testing Qualifications Board

Aspetto della qualità	Descrizione
	- variabilità nella disponibilità dei set di dati e motivazioni diverse dei fornitori di dati.
Dati iniqui	L'equità è una caratteristica di qualità soggettiva, ma spesso può essere identificata. Ad esempio, per sostenere la diversità o l'equilibrio di genere, i dati selezionati possono essere orientati positivamente verso minoranze o gruppi svantaggiati (si noti che tali dati possono essere considerati equi ma potrebbero non essere bilanciati).
Dati duplicati	Record di dati ripetuti possono influenzare indebitamente il modello di ML risultante.
Dati irrilevanti	I dati non rilevanti per il problema affrontato possono influenzare negativamente i risultati e possono portare a uno spreco di risorse.
Problemi di privacy	Qualsiasi utilizzo dei dati deve rispettare le leggi pertinenti sulla privacy dei dati (ad esempio, il GDPR in relazione alle informazioni personali degli individui nell'Unione Europea).
Problemi di sicurezza	Dati fraudolenti o fuorvianti inseriti deliberatamente nei dati di addestramento possono causare imprecisioni nel modello addestrato.

4.4 La qualità dei dati e il suo effetto sul modello di ML

La **qualità del modello di ML** dipende fortemente dalla qualità del set di dati da cui viene creato. Dati di scarsa qualità possono dare luogo a modelli imperfetti e previsioni errate.

Le seguenti categorie di difetti derivano da problemi di qualità dei dati:

- **Accuratezza ridotta**: Questi difetti sono causati da dati errati, incompleti, etichettati erroneamente, insufficienti, obsoleti, irrilevanti, e da dati non pre-elaborati.
  Ad esempio: se i dati fossero usati per costruire un modello dei prezzi attesi delle case, ma i dati di addestramento contenessero pochi o nessun dato sulle "case unifamiliari con una serra", i prezzi previsti per questo specifico tipo di casa sarebbero probabilmente imprecisi.
- **Modello orientato**: Questi difetti sono causati da dati incompleti, sbilanciati, iniqui, privi di diversità o duplicati.
  Ad esempio: se mancano i dati di una particolare caratteristica (ad esempio, tutti i dati medici per la previsione delle malattie sono raccolti da soggetti di un particolare genere), è probabile che ciò abbia un effetto negativo sul modello risultante (a meno che il modello non sia solo da utilizzare per fare previsioni operative per quel genere).
- **Modello compromesso**: Questi difetti sono dovuti alla privacy dei dati e alle restrizioni di sicurezza.
  Ad esempio: i problemi di privacy nei dati possono portare a vulnerabilità della sicurezza, che consentirebbero agli attacker di decodificare le informazioni dai modelli e potrebbero successivamente causare la fuga di informazioni personali.

© International Software Testing Qualifications Board

4.5 Etichettatura dei dati per l'apprendimento supervisionato

L'etichettatura dei dati è l'arricchimento di dati non etichettati (o scarsamente etichettati) con l'aggiunta di etichette, in modo da renderli adatti all'uso nell'apprendimento supervisionato. L'etichettatura dei dati è un'attività ad alta intensità di risorse che è stata segnalata per utilizzare, in media, il 25% del tempo nei progetti di ML [B11].

Nella sua forma più semplice, l'etichettatura dei dati può consistere nel mettere immagini o file di testo in varie cartelle, in base alle loro classi.
Ad esempio: inserire tutti i file di testo delle recensioni positive sui prodotti in una cartella e tutte le recensioni negative in un'altra cartella. L'etichettatura degli oggetti nelle immagini disegnando rettangoli attorno ad essi è un'altra tecnica di etichettatura comune, spesso conosciuta come annotazione. Annotazioni più complesse potrebbero essere necessarie per etichettare oggetti 3D o per disegnare bounding box attorno a oggetti irregolari. L'etichettatura e l'annotazione dei dati sono in genere supportate da strumenti.

4.5.1 Approcci all'etichettatura dei dati

L'etichettatura può essere eseguita in diversi modi:

- **Interna**: L'etichettatura viene eseguita da sviluppatori, tester o un team all'interno dell'organizzazione che è stata impostata per l'etichettatura.
- **In outsourcing**: L'etichettatura viene eseguita da un'organizzazione specializzata esterna.
- **Crowdsourcing**: L'etichettatura viene eseguita da un grande gruppo di individui. A causa della difficoltà di gestire la qualità dell'etichettatura, a più annotatori può essere chiesto di etichettare gli stessi dati e quindi si decide sull'etichetta da utilizzare.
- **Assistita da AI**: Vengono utilizzati strumenti basati su AI per riconoscere e annotare i dati o per raggruppare dati simili. I risultati vengono quindi confermati o forse integrati (ad esempio, modificando il bounding box) da un umano, come parte di un processo in due fasi.
- **Ibrida**: Potrebbe essere utilizzata una combinazione degli approcci di etichettatura di cui sopra. Ad esempio, l'etichettatura in crowdsourcing è in genere gestita da un'organizzazione esterna che ha accesso a strumenti specializzati di crowd management basati su AI.

Dove applicabile, potrebbe essere possibile **riutilizzare** un set di dati pre-etichettato, evitando così del tutto la necessità di etichettare i dati. Molti di questi set di dati sono disponibili pubblicamente, ad esempio, da Kaggle [R61].

4.5.2 Dati etichettati erroneamente nei set di dati

L'apprendimento supervisionato presuppone che i dati siano etichettati correttamente dagli annotatori dei dati. Tuttavia, nella pratica è raro che tutti gli elementi in un set di dati vengano etichettati correttamente. I dati sono etichettati erroneamente per i seguenti motivi:

- Gli annotatori possono commettere **errori casuali** (da esempio, premendo il pulsante sbagliato).
- Possono essere commessi **errori sistemici** (ad esempio, quando gli etichettatori ricevono istruzioni sbagliate o una formazione insufficiente).
- Possono essere commessi **errori intenzionali** da annotatori di dati malintenzionati.

© International Software Testing Qualifications Board

- Gli **errori di traduzione** possono prendere dati etichettati correttamente in una lingua ed etichettarli erroneamente in un'altra.
- Laddove la scelta è aperta all'interpretazione, i **giudizi soggettivi** espressi dagli annotatori dei dati possono portare a etichette di dati contrastanti da parte di diversi annotatori.
- La **mancanza della conoscenza del dominio** richiesta può portare a un'etichettatura errata.
- Compiti di **classificazione complessi** possono causare più errori.
- Gli **strumenti** utilizzati per supportare l'etichettatura dei dati presentano difetti che portano a etichette errate.
- Gli approcci all'etichettatura basati sul ML sono **probabilistici** e ciò può portare a etichette errate.

© International Software Testing Qualifications Board

5 Metriche delle prestazioni funzionali del ML - 120 minuti

Parole chiave

Nessuna

Parole chiave specifiche per l'AI

Accuratezza, area sotto la curva (AUC), matrice di confusione, punteggio F1, metriche inter-cluster, metriche intra-cluster, errore quadratico medio (MSE), suite di benchmark per il ML, metriche delle prestazioni funzionali del ML, precisione, richiamo, curva ROC (receiver operating characteristic), modello di regressione, R al quadrato, coefficiente di silhouette

Obiettivi di apprendimento per il Capitolo 5:

5.1 Matrice di confusione

AI-5.1.1 K3 Calcolare le metriche delle prestazioni funzionali del ML da un dato set di dati della matrice di confusione.

5.2 Ulteriori metriche delle prestazioni funzionali del ML per classificazione, regressione e clustering

AI-5.2.1 K2 Contrastare e confrontare i concetti alla base delle metriche delle prestazioni funzionali del ML per i metodi di classificazione, regressione e clustering.

5.3 Limitazioni delle metriche delle prestazioni funzionali del ML

AI-5.3.1 K2 Riassumere le limitazioni dell'utilizzo delle metriche delle prestazioni funzionali del ML per determinare la qualità del sistema di ML.

5.4 Selezionare le metriche delle prestazioni funzionali del ML

AI-5.4.1 K4 Selezionare le metriche delle prestazioni funzionali del ML appropriate e/o i relativi valori per un determinato modello e scenario di ML.

HO-5.4.1 H2 Valutare il modello di ML creato utilizzando le metriche delle prestazioni funzionali del ML selezionate

5.5 Suite di benchmark per il ML

AI-5.5.1 K2 Spiegare l'uso delle suite di benchmark nel contesto del ML

© International Software Testing Qualifications Board

5.1 Matrice di confusione

In un problema di classificazione, un modello raramente predice sempre i risultati correttamente. Per qualsiasi problema di questo tipo, è possibile creare una matrice di confusione con le seguenti combinazioni:

		Effettivo	
		Positivo	**Negativo**
Predetto	**Positivo**	**Vero positivo (TP)**	**Falso positivo (FP)**
	Negativo	**Falso negativo (FN)**	**Vero negativo (TN)**

Figura 2: Matrice di confusione

Si noti che la matrice di confusione mostrata nella Figura 2 può essere presentata in modo diverso ma genererà sempre valori per le quattro possibili situazioni di Vero Positivo (TP), Vero Negativo (TN), Falso Positivo (FP) e Falso Negativo (FN).

Sulla base della matrice di confusione, vengono definite le seguenti metriche:

- **Accuratezza**

  *Accuratezza = (TP + TN) / (TP +TN + FP + FN) * 100%*

  L'accuratezza misura la percentuale di tutte le classificazioni corrette.

- **Precisione**

  *Precisione = TP / (TP + FP) * 100%*

  La precisione misura la proporzione di positivi che sono stati predetti correttamente.
  È una misura di quanto si può essere sicuri delle previsioni positive.

- **Richiamo**

  *Richiamo = TP / (TP + FN) * 100%*

  Il richiamo (noto anche come sensibilità) misura la proporzione di effettivi positivi che sono stati predetti correttamente.
  È una misura di quanto si può essere sicuri di non perdere nessun positivo.

- **Punteggio F1**

  Punteggio F1 = 2 (precisione * richiamo) / (precisione + richiamo)*

  Il punteggio F1 viene calcolato come media armonica di precisione e richiamo.
  Avrà un valore compreso tra zero e uno. Un punteggio vicino a uno indica che i dati falsi hanno poca influenza sul risultato. Un basso punteggio F1 suggerisce che il modello è scarso nel rilevare i positivi.

© International Software Testing Qualifications Board

5.2 Ulteriori metriche delle prestazioni funzionali del ML per classificazione, regressione e clustering

Esistono numerose metriche per diversi tipi di problemi del ML (oltre a quelle relative alla classificazione descritte nella Sezione 5.1). Di seguito sono descritte alcune delle metriche più comunemente utilizzate.

Metriche di classificazione supervisionata

- **La curva ROC** (Receiver Operating Characteristic) è un grafico che illustra la capacità di un classificatore binario al variare della sua soglia di discriminazione. Il metodo è stato originariamente sviluppato per i radar militari, motivo per cui è così chiamato.
  La curva ROC è tracciata con la *Frazione di Vero Positivo (TPR)* (nota anche come richiamo) rispetto alla *Frazione di Falso Positivo (FPR = FP / (TN + FP))*, con TPR sull'asse y e FPR sull'asse x.
- **L'area sotto la curva (AUC)** è l'area sotto la curva ROC.
  Rappresenta il grado di separabilità di un classificatore, mostrando quanto bene il modello distingue tra le classi. Con un AUC più alta, le previsioni del modello sono migliori.

Metriche di regressione supervisionata

Per i modelli di regressione supervisionati, le metriche rappresentano quanto bene la retta di regressione si adatta ai punti di dati effettivi.

- L'**Errore Quadratico Medio (MSE)** è la media delle differenze al quadrato tra il valore effettivo e quello previsto.
  Il valore di MSE è sempre positivo e un valore più vicino a zero suggerisce un modello di regressione migliore. Elevando al quadrato la differenza, si garantisce che gli errori positivi e negativi non si annullino a vicenda.
- L'**R al quadrato** (noto anche come coefficiente di determinazione) è una misura di quanto bene il modello di regressione si adatta alle variabili dipendenti.

Metriche di clustering non supervisionato

Per il clustering non supervisionato, sono disponibili diverse metriche che rappresentano le distanze tra i vari cluster e la vicinanza dei punti di dati all'interno di un determinato cluster.

- Le **metriche intra-cluster** misurano la somiglianza dei punti di dati all'interno di un cluster.
- Le **metriche inter-cluster** misurano la somiglianza dei punti di dati in diversi cluster.
- Il **coefficiente di silhouette** (noto anche come punteggio di silhouette) è una misura (tra -1 e +1) basata sulla media delle distanze inter-cluster e intra-cluster.
  Un punteggio di:
  · +1 significa che i cluster sono ben separati
  · zero implica un raggruppamento casuale
  · -1 significa che i cluster sono assegnati in modo errato

5.3 Limitazioni delle metriche delle prestazioni funzionali del ML

Le metriche delle prestazioni funzionali del ML si limitano alla misurazione della funzionalità del modello, ad esempio, in termini di accuratezza, precisione, richiamo, MSE, AUC e coefficiente di silhouette.

© International Software Testing Qualifications Board

Non misurano altre caratteristiche di qualità non funzionali, come quelle definite nella ISO 25010 [S06] (ad esempio, efficienza delle prestazioni) e quelle descritte nel Capitolo 2, (ad esempio, spiegabilità, flessibilità e autonomia).

In questo syllabus, il termine "metriche delle prestazioni funzionali del ML" viene utilizzato a causa dell'uso diffuso del termine "metriche delle prestazioni" per riferirsi a queste metriche funzionali. L'aggiunta di "funzionali del ML" evidenzia che queste metriche sono specifiche del machine learning e non hanno alcuna relazione con le metriche di efficienza delle prestazioni.

Le metriche delle prestazioni funzionali del ML sono limitate da diversi altri fattori:

- Per l'apprendimento supervisionato, le metriche delle prestazioni funzionali del ML sono calcolate sulla base di dati etichettati e l'accuratezza delle metriche risultanti dipende dalla **corretta etichettatura** (vedi Sezione 4.5).
- I dati utilizzati per la misurazione potrebbero **non essere rappresentativi** (ad esempio, possono essere distorti) e le metriche delle prestazioni funzionali del ML generate dipendono da questi dati (vedi Sezione 2.4).
- Il sistema può comprendere diversi componenti, ma le metriche delle prestazioni funzionali del ML si applicano **solo al modello di ML**. Ad esempio, la pipeline di dati non viene considerata dalle metriche delle prestazioni funzionali del ML per valutare il modello.
- La maggior parte delle metriche delle prestazioni funzionali del ML può essere misurata solo con il **supporto degli strumenti**.

5.4 Selezionare le metriche delle prestazioni funzionali del ML

Normalmente non è possibile costruire un modello di ML che ottenga il punteggio più alto per tutte le metriche delle prestazioni funzionali del ML generate da una matrice di confusione.
Al contrario, le metriche delle prestazioni funzionali del ML più appropriate vengono selezionate come criteri di accettazione, in base all'uso previsto del modello (ad esempio, per ridurre al minimo i falsi positivi, è richiesto un alto valore di precisione, mentre per ridurre al minimo i falsi negativi, la metrica di richiamo dovrebbe essere alta).
I seguenti **criteri** possono essere utilizzati quando si selezionano le metriche delle prestazioni funzionali del ML descritte nelle Sezioni 5.1 e 5.2:

- **Accuratezza**: È probabile che questa metrica sia applicabile se i set di dati sono **simmetrici** (ad esempio, i conteggi dei falsi positivi e dei falsi negativi e i costi sono simili).
  Questa metrica diventa una scelta sbagliata se una classe di dati domina sulle altre, nel qual caso dovrebbe essere considerato il Punteggio F1.
- **Precisione**: Questa può essere una metrica adatta quando il **costo dei Falsi Positivi è elevato** e la fiducia nei risultati positivi deve essere elevata.
  Un filtro antispam, (in cui classificare un'email come spam è considerato positivo), è un esempio in cui è richiesta un'elevata precisione, poiché inserire nella cartella spam troppe email che non sono effettivamente spam non sarà accettabile per la maggior parte degli utenti.
  Quando il classificatore si occupa di situazioni in cui una percentuale molto ampia di casi è positiva, è improbabile che l'utilizzo della sola precisione sia una buona scelta.
- **Richiamo**: Quando è fondamentale **non perdere i positivi**, allora è importante un punteggio di richiamo elevato.
  Ad esempio, perdere tutti i risultati veri positivi nel rilevamento di un cancro e contrassegnarli come negativi (cioè, nessun cancro rilevato) è probabilmente inaccettabile.

© International Software Testing Qualifications Board

- **Punteggio F1**: È più utile quando c'è uno **squilibrio nelle classi previste** e quando **la Precisione** e il **Richiamo** hanno un'importanza simile.

Oltre alle metriche di cui sopra, nella Sezione 5.2 sono descritte diverse metriche. Queste possono essere applicabili per determinati problemi di ML, ad esempio:

- L'**AUC** della curva ROC può essere utilizzata per problemi di Classificazione Supervisionata.
- **MSE** e **R al Quadrato** possono essere utilizzati per problemi di Regressione Supervisionata.
- Le **Metriche inter-cluster**, le **Metriche intra-cluster** e il **Coefficiente di Silhouette** possono essere utilizzati per problemi di clustering non supervisionato.

5.4.1 Esercizio pratico: Valutare il modello di ML creato

Utilizzando il modello di classificazione addestrato nell'esercizio precedente, calcolate ed mostrate i valori di accuratezza, precisione, richiamo e punteggio F1. Dove applicabile, utilizzate le funzioni di libreria fornite dal vostro framework di sviluppo per eseguire i calcoli.

5.5 Suite di benchmark per il ML

Nuove tecnologie di AI come nuovi set di dati, algoritmi, modelli e hardware vengono rilasciati regolarmente e può essere difficile determinare l'efficacia relativa di ogni nuova tecnologia.

Per fornire confronti oggettivi tra queste diverse tecnologie, sono disponibili suite di benchmark per il ML come standard di settore. Queste coprono un'ampia gamma di aree di applicazione e forniscono strumenti per valutare piattaforme hardware, framework software e piattaforme cloud per le prestazioni di AI e ML.

Le suite di benchmark per il ML possono fornire varie **misure**, inclusi i tempi di addestramento (ad esempio, la velocità con cui un framework può addestrare un modello di ML utilizzando un set di dati di addestramento definito per una metrica di qualità specificata, come un'accuratezza del 75%) e i tempi di inferenza (ad esempio, quanto velocemente un modello di ML addestrato può eseguire l'inferenza).

Le suite di benchmark per il ML sono fornite da diverse organizzazioni, come ad esempio:

- **MLCommons** [R18]: È un'organizzazione no-profit costituita nel 2020 e precedentemente denominata ML Perf, che fornisce benchmark per framework software, processori specifici per l'AI e piattaforme cloud di ML.
- **DAWNBench** [R19]: È una suite di benchmark per il ML della Stanford University.
- **MLMark** [R20]: È una suite di benchmark per il ML progettata per misurare le prestazioni e l'accuratezza dell'inferenza embedded, dall'Embedded Microprocessor Benchmark Consortium.

© International Software Testing Qualifications Board

6 ML - Reti neurali e testing - 65 minuti

Parole chiave

Nessuna

Parole chiave specifiche per l'AI

Valore di attivazione, rete neurale profonda (DNN), dati di addestramento ML, percettrone multistrato, rete neurale, copertura dei neuroni, percettrone, copertura del cambio di segno, copertura segno-segno, apprendimento supervisionato, copertura della soglia, dati di addestramento, copertura del cambio di valore

Obiettivi di apprendimento per il Capitolo 6:

6.1 Reti neurali

AI-6.1.1 K2 Spiegare la struttura e la funzione di una rete neurale, inclusa una DNN.

HO-6.1.1 H1 Sperimentare l'implementazione di un percettrone.

6.2 Misure di copertura per le reti neurali

AI-6.2.1 K2 Descrivere le diverse misure di copertura per le reti neurali.

© International Software Testing Qualifications Board

6.1 Reti neurali

Inizialmente le reti neurali artificiali erano destinate a imitare il funzionamento del cervello umano, che può essere pensato come tanti neuroni biologici collegati. Il **percettrone monostrato** è uno dei primi esempi di implementazione di una rete neurale artificiale e comprende una rete neurale con un solo strato (cioè, un singolo neurone). Può essere utilizzato per l'apprendimento supervisionato dei classificatori, che decidono se un input appartiene o meno a una classe specifica.

La maggior parte delle reti neurali attuali sono considerate reti neurali profonde perché comprendono diversi strati e possono essere considerate come percettroni multistrato (vedere la Figura 3).

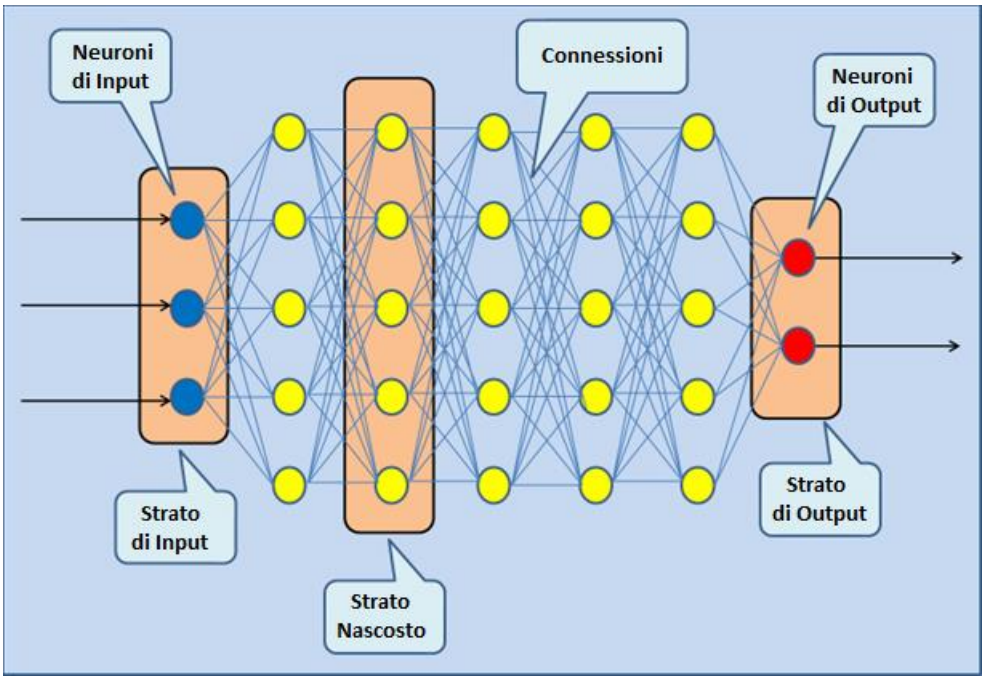

Figura 3 Struttura di una rete neurale profonda

Una rete neurale profonda comprende tre tipi di strati. Lo **strato di input** riceve gli input, ad esempio valori di pixel da una fotocamera. Lo **strato di output** fornisce risultati al mondo esterno. Questo potrebbe essere, ad esempio, un valore che indica la probabilità che l'immagine di input sia un gatto. Tra gli strati di input e di output ci sono **strati nascosti** costituiti da neuroni artificiali, noti anche come nodi. I neuroni in uno strato sono collegati a ciascuno dei neuroni nello strato successivo e possono esserci numeri diversi di neuroni in ogni strato successivo. I neuroni eseguono calcoli e trasmettono informazioni attraverso la rete dai neuroni di input ai neuroni di output.

© International Software Testing Qualifications Board

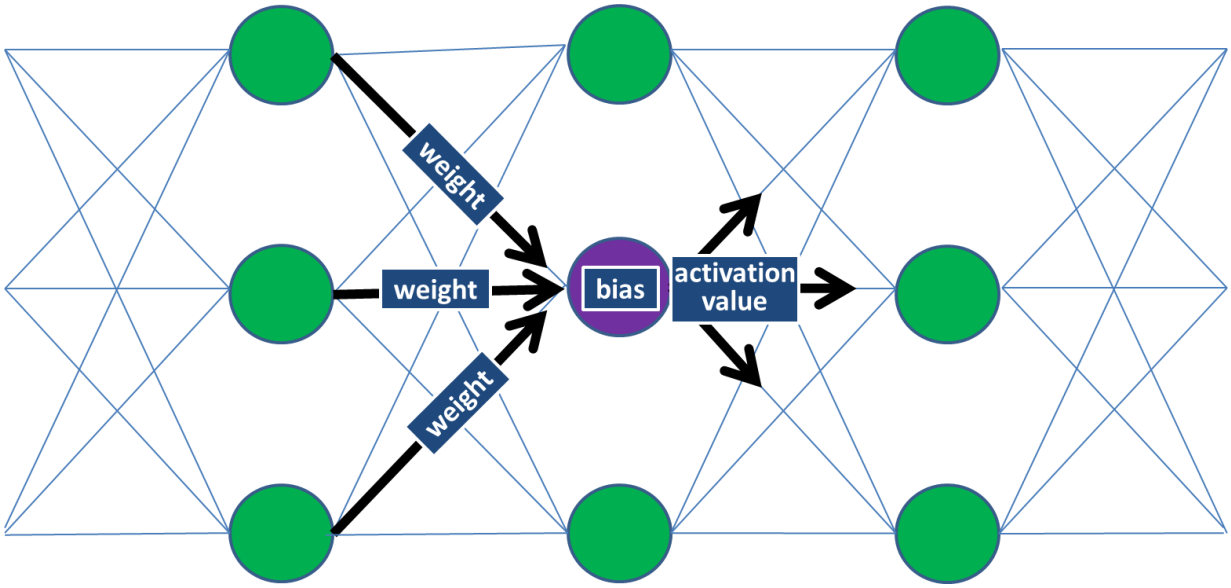

Figura 4 Calcolo eseguito da ogni neurone

Come mostrato nella Figura 4, il calcolo eseguito da ogni neurone (eccetto quelli nello strato di input) genera quello che è noto come **valore di attivazione**. Questo valore viene calcolato eseguendo una formula (la **funzione di attivazione**) che riceve come input i valori di attivazione da tutti i neuroni nello strato precedente, i **pesi** assegnati alle connessioni tra i neuroni (questi pesi cambiano man mano che la rete apprende) e il **bias** individuale di ogni neurone. Si noti che questo bias è un valore costante preimpostato e non è correlato ai bias considerati in precedenza nella Sezione 2.4). L'esecuzione di diverse funzioni di attivazione può comportare il calcolo di diversi valori di attivazione. Questi valori sono tipicamente centrati intorno allo zero e hanno un intervallo compreso tra -1 (significa che il neurone è "disinteressato") e +1 (significa che il neurone è "molto interessato").

Durante l'**addestramento** della rete neurale, ogni neurone è preimpostato su un valore di bias e i dati di addestramento vengono passati attraverso la rete, con ogni neurone che esegue la funzione di attivazione, per generare infine un output. L'output generato viene quindi confrontato con il risultato corretto conosciuto (in questo esempio di apprendimento supervisionato vengono utilizzati dati etichettati). La differenza tra l'output effettivo e il risultato corretto conosciuto viene quindi reimmessa attraverso la rete per modificare i valori dei pesi sulle connessioni tra i neuroni al fine di ridurre al minimo questa differenza. Man mano che più dati di addestramento vengono alimentati attraverso la rete, i pesi vengono gradualmente regolati man mano che la rete apprende. Alla fine, gli output prodotti sono considerati sufficientemente buoni per terminare l'addestramento.

6.1.1 Esercizio pratico: Implementare un percettrone semplice

Gli studenti saranno guidati attraverso un esercizio che mostra un Percettrone che impara una funzione semplice, come una funzione AND.

L'esercizio dovrebbe coprire il modo in cui un Percettrone impara modificando i pesi in un certo numero di epoch fino a quando l'errore non è zero. Per questa attività si possono utilizzare vari meccanismi (ad esempio, foglio di calcolo, simulazione).

© International Software Testing Qualifications Board

6.2 Misure di copertura per le reti neurali

Il raggiungimento dei criteri di copertura dei test white-box (ad esempio, copertura di istruzioni, rami, decisioni/condizioni modificate (MC/DC) [I01] è obbligatorio per la conformità ad alcuni standard relativi alla safety [S07] quando si utilizza il codice sorgente imperativo tradizionale ed è consigliato da molti esperti di test per altre applicazioni critiche. Il monitoraggio e il miglioramento della copertura supportano la progettazione di nuovi casi di test, portando ad una maggiore fiducia nell'oggetto di test.

L'uso di tali misure per misurare la copertura delle reti neurali fornisce scarso valore poiché lo stesso codice tende ad essere eseguito ogni volta che viene eseguita la rete neurale. Sono state invece proposte misure di copertura basate sulla copertura della struttura della rete neurale stessa e, più specificamente, dei neuroni al suo interno. La maggior parte di queste misure si basa sui valori di attivazione dei neuroni.

La copertura per le reti neurali è una nuova area di ricerca. I documenti accademici sono stati pubblicati solo dal 2017 e, in quanto tali, sono disponibili poche prove oggettive (ad esempio, risultati di ricerca duplicati) che dimostrino che le misure proposte sono efficaci. Tuttavia, va notato che, nonostante la copertura delle istruzioni e delle decisioni sia stata utilizzata per oltre 50 anni, ci sono poche prove oggettive anche della loro efficacia relativa (questo nonostante siano state imposte per misurare la copertura del software nelle applicazioni legate alla safety, come dispositivi medici e sistemi avionici).

I seguenti **criteri di copertura** per le reti neurali sono stati proposti e applicati dai ricercatori a una varietà di ambiti:

- **Copertura dei neuroni**: La copertura dei neuroni completa richiede che ogni neurone nella rete neurale raggiunga un valore di attivazione maggiore di zero [B12].
  Questo è molto facile da ottenere nella pratica e la ricerca ha dimostrato che si ottiene quasi il 100% di copertura con pochissimi casi di test su una varietà di reti neurali profonde.
  Questa misura di copertura può essere molto utile come segnale di allarme qualora non venisse raggiunta.
- **Copertura della soglia**: La copertura della soglia completa richiede che ogni neurone nella rete neurale raggiunga un valore di attivazione maggiore di una soglia specificata.
  I ricercatori che hanno creato il framework DeepXplore hanno effettivamente suggerito che la copertura dei neuroni dovrebbe essere misurata in base al valore di attivazione che supera una soglia che cambierebbe in base alla situazione. Hanno eseguito la loro ricerca con una soglia di 0,75 quando hanno riferito di aver trovato in modo efficiente migliaia di comportamenti errati dei casi d'angolo utilizzando questo approccio white-box.
  Questo tipo di copertura è stato qui rinominato per distinguerlo più facilmente dalla copertura dei neuroni con una soglia impostata su zero, poiché alcuni altri ricercatori usano il termine "copertura dei neuroni" per indicare la copertura dei neuroni con una soglia pari a zero.
- **Copertura del cambio di segno**: Per ottenere una copertura del cambio di segno completa, i casi di test devono far sì che ogni neurone raggiunga valori di attivazione sia positivi che negativi [B13].
- **Copertura del cambio di valore**: Per ottenere una copertura del cambiamento di valore completa, i casi di test devono far sì che ogni neurone raggiunga due valori di attivazione, dove la differenza tra i due valori supera un valore scelto [B13].
- **Copertura segno-segno**: Questa copertura considera coppie di neuroni in strati adiacenti e il segno assunto dai loro valori di attivazione.
  Affinché una coppia di neuroni sia considerata coperta, un caso di test deve dimostrare che la

© International Software Testing Qualifications Board

modifica del segno di un neurone nel primo strato fa sì che il neurone nel secondo strato cambi il suo segno, mentre i segni di tutti gli altri neuroni nel secondo strato rimangono invariati [B13]. Questo è un concetto simile alla copertura MC/DC per il codice sorgente imperativo.

I ricercatori hanno riportato ulteriori misure di **copertura basate sugli strati** (sebbene più semplici della Copertura segno-segno) e nello strumento TensorFuzz [B14] è stato implementato un approccio di successo che utilizza algoritmi nearest neighbor per identificare cambiamenti significativi negli insiemi di neuroni vicini.

© International Software Testing Qualifications Board

7 Panoramica sul testing di sistemi basati su AI - 115 minuti

Parole chiave

Testing dei dati di input, testing del modello di ML

Parole chiave specifiche per l'AI

Componente AI, automation bias (condizionamento dell'automazione), big data, deriva concettuale, pipeline di dati, metriche delle prestazioni funzionali del ML, dati di addestramento

Obiettivi di apprendimento per il Capitolo 7:

7.1 Specifiche dei sistemi basati su AI

AI-7.1.1 K2 Spiegare come le specifiche di sistema per i sistemi basati su AI possono creare sfide nel testing.

7.2 Livelli di test per i sistemi basati su AI

AI-7.2.1 K2 Descrivere come vengono testati i sistemi basati su AI ad ogni livello di test

7.3 Dati di test per il testing dei sistemi basati su AI

AI-7.3.1 K1 Richiamare quei fattori associati ai dati di test che possono rendere difficile il testing dei sistemi basati su AI.

7.4 Testing per l'automation bias nei sistemi basati su AI

AI-7.4.1 K2 Spiegare l'automation bias e come questo influisce sul testing.

7.5 Documentare un modello di ML

AI-7.5.1 K2 Descrivere la documentazione di un componente AI e comprendere in che modo la documentazione supporta il testing dei sistemi basati su AI.

7.6 Testing per la deriva concettuale

AI-7.6.1 K2 Spiegare la necessità di testare frequentemente il modello addestrato per gestire la deriva concettuale.

7.7 Selezione di un approccio di test per un sistema di ML

AI-7.7.1 K4 Per un determinato scenario, determinare un approccio di test da seguire durante lo sviluppo di un sistema di ML.

© International Software Testing Qualifications Board

7.1 Specifiche dei sistemi basati su AI

I requisiti di sistema e le specifiche di progettazione sono ugualmente importanti sia per i sistemi basati su AI che per i sistemi convenzionali. Queste specifiche forniscono la base ai tester per verificare se il comportamento effettivo del sistema è in linea con i requisiti specificati. Tuttavia, se le specifiche sono incomplete e mancano di testabilità, ciò introduce un problema dell'oracolo del test (vedi Sezione 8.7).

Ci sono diversi motivi per cui le specifiche dei sistemi basati su AI possono essere particolarmente impegnative:

- In molti progetti di sistemi basati su AI, i **requisiti sono specificati solo in termini di obiettivi di business di alto livello e previsioni richieste**.
  Una ragione di ciò è la natura esplorativa dello sviluppo di sistemi basati su AI. Spesso, i progetti di sistemi basati su AI iniziano con un set di dati e l'obiettivo è determinare quali previsioni possono essere ottenute da quei dati. Ciò è in contrasto con lo specificare la logica richiesta dall'inizio di un progetto convenzionale.
- **L'accuratezza del sistema basato su AI è spesso sconosciuta** fino a quando non sono disponibili i risultati del testing indipendente. Insieme all'approccio di sviluppo esplorativo, ciò porta spesso a specifiche inadeguate poiché l'implementazione è già in corso nel momento in cui vengono determinati i criteri di accettazione desiderati.
- La natura **probabilistica** di molti sistemi basati su AI può rendere necessario specificare le tolleranze per alcuni dei requisiti di qualità previsti, come l'accuratezza delle previsioni.
- Laddove gli obiettivi del sistema richiedano di **replicare il comportamento umano**, piuttosto che fornire funzionalità specifiche, ciò spesso porta a requisiti di comportamento scarsamente specificati basati sul fatto che il sistema sia altrettanto efficace, o anche migliore, delle attività umane che mira a sostituire. Ciò può rendere difficile definire un oracolo del test, specialmente quando gli esseri umani che sta sostituendo variano ampiamente nelle loro capacità.
- Laddove l'AI viene utilizzata per implementare le **interfacce utente**, come il riconoscimento del linguaggio naturale, la visione artificiale o l'interazione fisica con gli esseri umani, i sistemi devono dimostrare una maggiore flessibilità. Tuttavia, tale flessibilità può anche creare difficoltà nell'identificare e documentare tutti i diversi modi in cui tali interazioni potrebbero verificarsi.
- Le caratteristiche di qualità specifiche dei sistemi basati su AI, come adattabilità, flessibilità, evoluzione e autonomia, devono essere considerate e definite come parte della specifica dei requisiti (vedi Capitolo 2). La **novità di queste caratteristiche** può renderle difficili da definire e testare.

7.2 Livelli di test per i sistemi basati su AI

I sistemi basati su AI in genere comprendono sia componenti AI che non-AI. I componenti non-AI possono essere testati utilizzando approcci convenzionali [I01], mentre i componenti AI e i sistemi che contengono componenti AI potrebbero dover essere testati in modo diverso sotto alcuni aspetti, come descritto di seguito. Per tutti i livelli di test che includono il testing di componenti AI, è importante che il testing sia strettamente supportato da data engineers/scientists ed esperti di dominio.

Una delle principali differenze rispetto ai livelli di test utilizzati per il software convenzionale è l'inclusione di due nuovi livelli di test specializzati per gestire esplicitamente il testing dei dati di input e

© International Software Testing Qualifications Board

dei modelli utilizzati nei sistemi basati su AI [B15]. La maggior parte di questa sezione è applicabile a tutti i sistemi basati su AI, sebbene alcune parti siano specificamente incentrate sul ML.

7.2.1 Testing dei dati di input

L'obiettivo del testing dei dati di input è garantire che i **dati utilizzati dal sistema per l'addestramento e la previsione siano della massima qualità** (vedi Sezione 4.3). Include quanto segue:

- Reviews
- Tecniche statistiche (ad esempio, testing dei dati per i bias)
- EDA dei dati di addestramento
- Testing statico e dinamico della pipeline di dati

La pipeline di dati comprende -in genere- diversi componenti che eseguono la preparazione dei dati (vedi Sezione 4.1), e il testing di questi componenti include sia il Testing di Componente che il Testing di Integrazione. La pipeline di dati per l'addestramento potrebbe essere molto diversa dalla pipeline di dati utilizzata per supportare la previsione operativa. Per l'addestramento, la pipeline di dati può essere considerata un prototipo, rispetto alla versione completamente ingegnerizzata e automatizzata utilizzata a livello operativo. Per questo motivo, il testing di queste due versioni della pipeline di dati potrebbe essere abbastanza diverso. Tuttavia, dovrebbe essere considerato anche il testing dell'Equivalenza Funzionale delle due versioni.

7.2.2 Testing del modello di ML

L'obiettivo del testing del modello di ML è garantire che il modello selezionato soddisfi tutti i criteri delle prestazioni che potrebbero essere stati specificati. Questo include:

- Criteri delle **prestazioni funzionali** del ML (vedere Sezioni 5.1 e 5.2)
- Criteri di **accettazione non funzionali** del ML appropriati per il modello in sé, come velocità di addestramento, velocità di previsione, risorse di calcolo utilizzate, adattabilità e trasparenza.

Il testing del modello di ML mira anche a determinare che la scelta del framework, dell'algoritmo, del modello, delle impostazioni del modello e degli iperparametri del ML sia il più vicino possibile all'ottimale. Dove appropriato, il testing del modello di ML può anche includere il testing per ottenere i criteri di copertura white-box (vedi Sezione 6.2). Il modello selezionato viene successivamente integrato con altri componenti, AI e non-AI.

7.2.3 Testing di componente

Il testing di componente è un livello di test convenzionale applicabile a qualsiasi componente non del modello, come interfacce utente e componenti di comunicazione.

7.2.4 Testing di integrazione dei componenti

Il testing di integrazione dei componenti è un livello di test convenzionale condotto per garantire che i componenti del sistema (sia AI che non-AI) interagiscano correttamente. Verifica che gli input dalla pipeline di dati siano ricevuti come previsto dal modello e che tutte le previsioni prodotta dal modello siano scambiate con i componenti di sistema pertinenti (ad esempio, l'interfaccia utente) e utilizzate

© International Software Testing Qualifications Board

correttamente da questi ultimi. Quando l'AI è fornita come servizio (vedi Sezione 1.7), è normale eseguire il testing delle API del servizio fornito come parte del testing di integrazione dei componenti.

7.2.5 Testing di sistema

Il testing di sistema è un livello di test convenzionale che viene condotto per garantire che il sistema completo di componenti integrati (sia AI che non-AI) funzioni come previsto, sia dal punto di vista funzionale che non funzionale, in un ambiente di test strettamente rappresentativo dell'ambiente operativo. A seconda del sistema, questo testing può assumere la forma di prove sul campo nell'ambiente operativo previsto o di testing all'interno di un simulatore (ad esempio, se gli scenari di test sono pericolosi o difficili da replicare in un ambiente operativo).

Durante il testing di sistema, i criteri delle prestazioni funzionali del ML vengono ri-testati per garantire che i risultati dei test del modello di ML iniziale non siano influenzati negativamente quando il modello è integrato in un sistema completo. Questo testing è particolarmente importante quando il componente AI è stato deliberatamente modificato (ad esempio, comprimendo un DNN per ridurne le dimensioni).

Il testing di sistema è anche il livello di test in cui vengono testati molti dei requisiti non funzionali per il sistema. Ad esempio, è possibile eseguire l'Adversarial Testing per testare la robustezza e il sistema può essere testato per la spiegabilità. Dove appropriato, le interfacce con i componenti hardware (ad esempio, i sensori) possono essere testate come parte del testing di sistema.

7.2.6 Testing di accettazione

Il testing di accettazione è un livello di test convenzionale e viene utilizzato per determinare se il sistema completo è accettabile per il cliente. Per i sistemi basati su AI, la definizione dei criteri di accettazione può essere difficile (vedi Sezione 8.8).
Quando l'AI viene fornita come servizio (vedi Sezione 1.7), può essere necessario il testing di accettazione per determinare l'idoneità del servizio per il sistema previsto e se, ad esempio, i criteri delle prestazioni funzionali del ML sono stati sufficientemente raggiunti.

7.3 Dati di test per il testing dei sistemi basati su AI

A seconda della situazione e del System Under Test (SUT), l'acquisizione dei dati di test potrebbe rappresentare una sfida. Esistono diverse potenziali sfide nella gestione dei dati di test per i sistemi basati su AI, tra cui:

- I **big data** (dati ad alto volume, ad alta velocità e ad alta varietà) possono essere difficili da creare e gestire. Ad esempio, potrebbe essere difficile creare dati di test rappresentativi per un sistema che consuma grandi volumi di immagini e audio ad alta velocità.
- Potrebbe essere necessario **modificare i dati** di input nel tempo, in particolare se rappresentano eventi nel mondo reale. Ad esempio, le fotografie registrate per testare un sistema di riconoscimento facciale potrebbero dover essere "invecchiate" per rappresentare l'invecchiamento delle persone nel corso di diversi anni nella vita reale.
- I dati personali o altrimenti riservati potrebbero richiedere tecniche speciali per la **sanificazione**, la **crittografia** o la **redazione**. Potrebbe essere richiesta anche l'approvazione legale per l'uso.
- Quando i tester utilizzano la **stessa implementazione** dei data scientist per l'acquisizione e la pre-elaborazione dei dati, i difetti in questi passaggi possono essere mascherati.

© International Software Testing Qualifications Board

7.4 Testing per l'automation bias nei sistemi basati su AI

Una categoria di sistemi basati su AI aiuta gli umani nel processo decisionale. Tuttavia, a volte c'è la tendenza per gli umani a fidarsi troppo di questi sistemi. Questa fiducia mal riposta può essere chiamata **Automation Bias** o **Complacency Bias** e assume due forme.

- La prima forma di automation/complacency bias è quando l'umano **accetta le raccomandazioni fornite dal sistema** e non considera gli input provenienti da altre fonti (incluso sé stesso).
  Ad esempio, una procedura in cui un umano inserisce i dati in un modulo potrebbe essere migliorata utilizzando il machine learning per precompilare il modulo, e l'umano quindi convalida questi dati. È stato dimostrato che questa forma di Automation Bias in genere riduce la qualità delle decisioni prese del 5%, ma questa potrebbe essere molto maggiore a seconda del contesto del sistema [B16]. Allo stesso modo, la correzione automatica del testo digitato (ad esempio, nei messaggi del cellulare) è spesso difettosa e potrebbe cambiare il significato. Gli utenti spesso non se ne accorgono, e non annullano l'errore.
- La seconda forma di automation/complacency bias è quando l'umano **non si accorge di un errore del sistema** perché non lo monitora adeguatamente.
  Ad esempio, i veicoli semi-autonomi stanno diventando sempre più a guida autonoma, ma fanno ancora affidamento su un umano che prenda il controllo in caso di un incidente imminente. In genere, l'occupante umano del veicolo diventa gradualmente troppo fiducioso delle capacità del sistema di controllare il veicolo e inizia a prestare meno attenzione. Questo può portare a una situazione in cui non è in grado di reagire in modo appropriato quando necessario.

In entrambi gli scenari, i tester dovrebbero capire in che modo il processo decisionale umano può essere compromesso e testare sia la qualità delle raccomandazioni del sistema che la qualità del corrispondente input umano fornito da utenti rappresentativi.

7.5 Documentare un componente AI

Il contenuto tipico della **documentazione di un componente AI** include:

- **Generale**: Identificatori, descrizione, dettagli dello sviluppatore, requisiti hardware, dettagli della licenza, versione, data e punto di contatto.
- **Progettazione**: Ipotesi e decisioni tecniche.
- **Utilizzo**: Use case primari e secondari, utenti tipici, approccio all'autoapprendimento, bias noti, problemi etici, problemi di sicurezza, trasparenza, soglie di decisione, piattaforma e deriva concettuale.
- **Set di dati**: Caratteristiche, raccolta, disponibilità, requisiti di pre-elaborazione, uso, contenuto, etichettatura, dimensioni, privacy, sicurezza, bias/equità e restrizioni/vincoli.
- **Testing**: Set di dati di test (descrizione e disponibilità), indipendenza del testing, risultati dei test, approccio al testing per robustezza, spiegabilità, deriva concettuale e portabilità.
- **Addestramento e prestazioni funzionali del ML**: Algoritmo di ML, pesi, set di dati di validazione, selezione delle metriche delle prestazioni funzionali del ML, soglie per le metriche delle prestazioni funzionali del ML e metriche effettive delle prestazioni funzionali del ML.

© International Software Testing Qualifications Board

Una documentazione chiara aiuta a migliorare il testing fornendo trasparenza sull'implementazione del sistema basato su AI. Le **aree chiave della documentazione** che sono importanti per il testing sono:

- Lo **scopo del sistema** e la **specifica dei requisiti funzionali** e **non funzionali**.
  Questi tipi di documentazione in genere fanno parte della base di test.
- **Informazioni architettoniche** e di **progettazione**, che descrivono come interagiscono i diversi componenti AI e non-AI. Supporta l'identificazione degli obiettivi del testing di integrazione e può fornire una base per il testing white-box della struttura del sistema.
- La **specifica dell'ambiente operativo**.
  È necessaria quando si testano l'autonomia, la flessibilità e l'adattabilità del sistema.
- La **fonte di tutti i dati di input**, inclusi i metadati associati.
  Deve essere compresa chiaramente quando si testano i seguenti aspetti:
  - Correttezza funzionale di input inaffidabili
  - Bias di campionamento esplicito o implicito
  - Flessibilità, compreso l'apprendimento errato da scarsi input di dati per i sistemi di autoapprendimento
- Il modo in cui ci si aspetta che il sistema si **adatti ai cambiamenti** nel suo ambiente operativo. È necessario come base di test quando si testa l'adattabilità.
- Dettagli degli **utenti** previsti del sistema. Sono necessari per garantire che il testing possa essere rappresentativo.

7.6 Testing per la deriva concettuale

L'ambiente operativo può cambiare nel tempo senza che il modello addestrato cambi di conseguenza. Questo fenomeno è noto come **deriva concettuale** e in genere fa sì che gli output del modello diventino sempre meno accurati e meno utili.
Ad esempio, l'impatto delle campagne di marketing può comportare un cambiamento nel comportamento dei potenziali clienti per un periodo di tempo. Tali cambiamenti potrebbero essere stagionali o improvvisi dovuti a cambiamenti culturali, morali o sociali esterni al sistema. Un esempio un cambiamento così improvviso è l'impatto della pandemia di COVID-19 e il suo effetto sull'accuratezza dei modelli utilizzati per le proiezioni di vendita e i mercati azionari.

I sistemi che potrebbero essere soggetti a deriva concettuale dovrebbero essere regolarmente testati rispetto ai criteri delle prestazioni funzionali del ML concordati, per garantire che eventuali occorrenze di deriva concettuale vengano rilevate abbastanza presto da mitigare il problema. Le **mitigazioni** tipiche possono includere il ritiro del sistema o il riaddestramento del sistema. In caso di riaddestramento, questo verrebbe eseguito con dati di addestramento aggiornati e seguito da **testing** confermativo, testing di regressione ed eventualmente una forma di testing A/B (vedi Sezione 9.4), in cui il sistema B aggiornato deve superare il sistema A originale.

7.7 Selezione di un approccio di test per un sistema di ML

Un sistema basato su AI includerà in genere sia componenti AI che non-AI. L'**approccio di test** si basa su un'analisi del rischio per tale sistema e includerà sia testing convenzionale che testing più specializzato per affrontare quei fattori specifici dei componenti AI e dei sistemi basati su AI.

© International Software Testing Qualifications Board

L'elenco seguente fornisce alcuni rischi tipici e le relative mitigazioni, specifici per i sistemi di ML. Si noti che questo elenco fornisce solo una serie limitata di esempi e che esistono molti più rischi specifici per i sistemi di ML che richiedono una mitigazione tramite testing.

Aspetto del rischio	**Descrizione e possibili mitigazioni**
La qualità dei dati potrebbe essere inferiore al previsto	Questo rischio può diventare un problema in diverse maniere, ognuna delle quali può essere prevenuto in modo specifico (vedi Sezione 4.4). Le mitigazioni comuni includono l'uso di **review**, **EDA** e **testing dinamico**.
La pipeline di dati operativi potrebbe essere difettosa	Questo rischio può essere in parte mitigato dal **testing dinamico dei singoli componenti** della pipeline e dal **testing di integrazione della pipeline completa**.
Il flusso di lavoro di ML utilizzato per sviluppare il modello potrebbe non essere ottimale (Vedi Sezione 3.2)	Questo rischio potrebbe essere dovuto a: • Una mancanza di un accordo iniziale sul flusso di lavoro di ML da seguire • Una scelta sbagliata del flusso di lavoro • Data engineers che non seguono il flusso di lavoro Le **review con esperti** possono mitigare la possibilità di scegliere il flusso di lavoro sbagliato, mentre una **gestione più pratica** o **audit** potrebbero risolvere i problemi di accordo e implementazione del flusso di lavoro.
La scelta del framework, dell'algoritmo, del modello, delle impostazioni del modello e/o degli iperparametri del ML potrebbe non essere ottimale	Questo rischio potrebbe essere dovuto alla mancanza di competenza dei decisori, o alla scarsa implementazione delle fasi di valutazione e tuning (o fase di test) del flusso di lavoro di ML. Le **review con esperti** possono mitigare la possibilità di prendere decisioni sbagliate e una **migliore gestione** può garantire che le fasi di valutazione e tuning (e test) del flusso di lavoro siano seguite.
I criteri delle prestazioni funzionali del ML desiderati potrebbero non essere forniti operativamente, nonostante il componente ML soddisfi tali criteri in modo isolato	Questo rischio potrebbe essere dovuto al fatto che i set di dati utilizzati per l'addestramento e il testing del modello in isolamento non sono rappresentativi dei dati rilevati operativamente. Le **review** dei set di dati selezionati da parte di esperti (o utenti) possono mitigare la possibilità che i dati selezionati non siano rappresentativi.
I criteri delle prestazioni funzionali del ML desiderati sono soddisfatti, ma gli utenti potrebbero non essere soddisfatti dei risultati forniti	Questo rischio potrebbe essere dovuto alla selezione di criteri delle prestazioni errati (ad esempio, è stato selezionato un richiamo elevato quando era necessaria una precisione elevata). Le **review con esperti** possono mitigare la possibilità di scegliere le metriche delle prestazioni funzionali del ML sbagliate, o anche il **testing basato sull'esperienza** potrebbe identificare criteri

© International Software Testing Qualifications Board

Aspetto del rischio	Descrizione e possibili mitigazioni
	inappropriati. Il rischio potrebbe anche essere dovuto alla deriva concettuale, nel qual caso un **testing più frequente del sistema operativo** potrebbero mitigare il rischio.
I criteri delle prestazioni funzionali del ML desiderati sono soddisfatti, ma gli utenti potrebbero non essere soddisfatti del servizio fornito	Questo rischio potrebbe essere dovuto alla mancanza di attenzione ai requisiti non funzionali del sistema. Si noti che la gamma di caratteristiche di qualità per i sistemi basati su AI si estende oltre quelle elencate nella ISO/IEC 25010 (vedi Capitolo 2). L'utilizzo di un **approccio basato sul rischio** per prioritizzare le caratteristiche di qualità e l'esecuzione del relativo testing non funzionale potrebbe mitigare questo rischio. In alternativa, il problema potrebbe essere dovuto a una combinazione di fattori che potrebbero essere identificati tramite il **testing basato sull'esperienza**, come parte del testing di sistema. Il Capitolo 8 fornisce indicazioni su come testare queste caratteristiche.
Il sistema di autoapprendimento potrebbe non fornire il servizio previsto dagli utenti	Questo rischio potrebbe essere dovuto a vari motivi, ad esempio: • I dati utilizzati dal sistema per l'autoapprendimento possono essere inappropriati. In questo caso, le **review degli esperti** potrebbero identificare i dati problematici. • Il sistema potrebbe non funzionare a causa dell'inaccettabilità della nuova funzionalità auto-appresa. Questo potrebbe essere mitigato dal **testing di regressione automatizzato**, incluso il confronto delle prestazioni con la funzionalità precedente. • Il sistema potrebbe apprendere in un modo non previsto dagli utenti, che potrebbe essere rilevato dal **testing basato sull'esperienza**.
Gli utenti potrebbero essere frustrati dal non capire come il sistema determina le sue decisioni	Questo rischio potrebbe essere dovuto a una mancanza di spiegabilità, interpretabilità e/o trasparenza. Vedere la Sezione 8.6 per i dettagli su come testare queste caratteristiche.
Gli utenti potrebbero scoprire che il modello fornisce previsioni eccellenti quando i dati sono simili a quelli di addestramento, ma fornisce risultati scadenti in caso contrario	Questo rischio può essere dovuto all'overfitting (vedi Sezione 3.5.1), che può essere rilevato testando il modello con un set di dati che è completamente **indipendente** dal set di dati di addestramento o eseguendo il **testing basato sull'esperienza**.

© International Software Testing Qualifications Board

8 Testing delle caratteristiche di qualità specifiche dell'AI - 150 minuti

Parole chiave

Oracolo di test

Parole chiave specifiche per l'AI

Bias algoritmico, sistema autonomo, autonomia, sistema esperto, spiegabilità, bias inappropriato, interpretabilità, metodo LIME, dati di addestramento ML, sistema non deterministico, sistema probabilistico, bias di campionamento, sistema di autoapprendimento, trasparenza

Obiettivi di apprendimento per il Capitolo 8:

8.1 Sfide nel testing dei sistemi di auto-apprendimento

AI-8.1.1 K2 Spiegare le sfide nel testing create dall'autoapprendimento dei sistemi basati su AI.

8.2 Testing dei sistemi autonomi basati su AI

AI-8.2.1 K2 Descrivere come vengono testati i sistemi autonomi basati su AI

8.3 Testing per bias algoritmici, di campionamento e inappropriati

AI-8.3.1 K2 Spiegare come testare la presenza di bias in un sistema basato su AI.

8.4 Sfide nel testing dei sistemi basati su AI probabilistici e non deterministici

AI-8.4.1 K2 Spiegare le sfide nel testing create dalla natura probabilistica e non deterministica dei sistemi basati su AI.

8.5 Sfide nel testing dei sistemi complessi basati su AI

AI-8.5.1 K2 Spiegare le sfide nel testing create dalla complessità dei sistemi basati su AI.

8.6 Testare la trasparenza, l'interpretabilità e la spiegabilità dei sistemi basati su AI

AI-8.6.1 K2 Descrivere come è possibile testare la trasparenza, l'interpretabilità e la spiegabilità dei sistemi basati su AI.

HO-8.6.1 H2 Utilizzare uno strumento per mostrare come la spiegabilità può essere utilizzata dai tester.

8.7 Oracoli di test per sistemi basati su AI

AI-8.7.1 K2 Spiegare le sfide nella creazione di oracoli di test derivanti dalle caratteristiche specifiche dei sistemi basati su AI.

8.8 Obiettivi dei test e criteri di accettazione

AI-8.8.1 K4 Selezionare obiettivi di test e criteri di accettazione appropriati per le caratteristiche di qualità specifiche dell'AI di un determinato sistema basato su AI.

© International Software Testing Qualifications Board

8.1 Sfide nel testing dei sistemi di autoapprendimento

Ci sono diverse potenziali sfide da superare quando si testano i sistemi di autoapprendimento (vedi il Capitolo 2 per maggiori dettagli su questi sistemi), tra cui:

- **Modifica imprevista**: I requisiti originali e i vincoli entro i quali il sistema dovrebbe funzionare sono generalmente noti, di contro poche o nessuna informazione sulle modifiche apportate dal sistema stesso potrebbero essere disponibili. Normalmente è possibile eseguire test rispetto ai requisiti e al progetto originali (ed eventuali vincoli specificati), ma se il sistema ha ideato un'implementazione innovativa o ha messo in atto una soluzione completamente nuova (la cui implementazione non è visibile), potrebbe essere difficile progettare test che siano appropriati per questa nuova implementazione. Inoltre, quando i sistemi cambiano sé stessi (e i loro output), i risultati dei test -che prima venivano superati- possono cambiare. Questa è una sfida di progettazione dei test. Può essere affrontata progettando test appropriati che rimangano rilevanti quando il sistema cambia il suo comportamento, prevenendo così un potenziale problema di testing di regressione. Tuttavia, ciò può anche richiedere la progettazione di nuovi test in base ai nuovi comportamenti osservati del sistema.
- **Criteri di accettazione complessi**: Può essere necessario definire le aspettative di miglioramento da parte del sistema quando questo autoapprende. Ad esempio, si può presumere che se il sistema si modificasse da solo, le sue prestazioni funzionali complessive dovrebbero migliorare. Inoltre, specificare qualcosa di diverso dal semplice "miglioramento" può diventare rapidamente complesso. Ad esempio, potrebbe essere previsto un miglioramento minimo (piuttosto che semplicemente qualsiasi miglioramento), oppure il miglioramento richiesto potrebbe essere collegato a fattori ambientali (ad esempio, è necessario un miglioramento minimo del 10% della funzionalità X se il fattore ambientale F cambia più di Y). Questi problemi possono essere affrontati attraverso la specifica e il testing rispetto ai criteri di accettazione più complessi, e mantenendo una registrazione continua delle attuali prestazioni funzionali di base del sistema corrente.
- **Tempo di testing insufficiente**: Potrebbe essere necessario sapere quanto velocemente si prevede che il sistema apprenda e si adatti, dati diversi scenari. Questi criteri di accettazione possono essere difficili da specificare e acquisire. Se un sistema si adatta velocemente, potrebbe non esserci tempo sufficiente per eseguire manualmente nuovi test dopo ogni modifica, quindi potrebbe essere necessario scrivere test che possono essere eseguiti automaticamente quando il sistema cambia. Queste sfide possono essere affrontate attraverso la specifica di criteri di accettazione appropriati (vedi Sezione 8.8) e Continuous Testing automatizzato.
- **Requisiti delle risorse**: I requisiti di sistema potrebbero includere criteri di accettazione per le risorse che il sistema può utilizzare durante l'autoapprendimento o l'adattamento. Possono includere, ad esempio, la quantità di tempo di elaborazione e di memoria che può utilizzare per migliorarsi. Inoltre, è necessario considerare se questo utilizzo delle risorse debba essere collegato a un miglioramento misurabile della funzionalità o dell’accuratezza. Questa sfida influenza la specifica dei criteri di accettazione.
- **Specifiche insufficienti dell'ambiente operativo**: Un sistema di autoapprendimento può cambiare se gli input ambientali che riceve sono al di fuori degli intervalli previsti o se questi non si riflettono nei dati di addestramento. Questi input possono essere attacchi sotto forma di avvelenamento dei dati (vedi Sezione 9.1.2). Può essere difficile prevedere l'intera gamma di ambienti operativi e cambiamenti ambientali e quindi identificare l'intera serie di casi di test rappresentativi e requisiti ambientali. Idealmente, l'intera portata dei possibili cambiamenti nell'ambiente operativo a cui ci si aspetta che il sistema risponda sarà definita come criterio di accettazione.

© International Software Testing Qualifications Board

- **Ambiente di test complesso**: La gestione dell'ambiente di test per garantire che possa simulare tutte le potenziali modifiche dell'ambiente operativo ad alto rischio è una sfida e può comportare l'uso di strumenti di test (ad esempio, uno strumento di fault injection). A seconda della natura dell'ambiente operativo, questo può essere testato manipolando input e sensori o ottenendo l'accesso a diversi ambienti fisici in cui il sistema può essere testato.
- **Modifiche indesiderate del comportamento**: Un sistema di autoapprendimento modifica il proprio comportamento in base ai propri input e potrebbe non essere possibile per i tester impedire che ciò accada. Questo può verificarsi, ad esempio, se viene utilizzato un sistema di terze parti o se viene testato il sistema di produzione. Ripetendo gli stessi test, un sistema di autoapprendimento può diventare più efficace nel rispondere a tali test, che possono quindi influenzare il comportamento a lungo termine del sistema. È quindi importante prevenire una situazione in cui il testing induca un sistema di autoapprendimento a modificare negativamente il proprio comportamento. Questa è una sfida per la progettazione dei casi di test e per la gestione dei test.

8.2 Testing dei sistemi autonomi basati su AI

I sistemi autonomi devono essere in grado di determinare quando necessitano dell'intervento umano e quando no. Pertanto, testare l'autonomia dei sistemi basati su AI richiede che vengano ricreate le condizioni affinché il sistema possa esercitare questo processo decisionale.

Il testing per l'autonomia può richiedere:

- Testare se il sistema richiede l'intervento umano per uno **scenario specifico** quando il sistema dovrebbe rinunciare al controllo. Tali scenari potrebbero includere una modifica dell'ambiente operativo o il superamento del sistema dei limiti della sua autonomia.
- Testare se il sistema richiede l'intervento umano quando il sistema dovrebbe rinunciare al controllo dopo un determinato **periodo di tempo**.
- Testare se il sistema **richiede inutilmente** l'intervento umano quando dovrebbe ancora funzionare in modo autonomo.

L'Analisi dei Valori Limite applicata all'ambiente operativo può essere utile al fine di generare le condizioni necessarie per questo testing. Può essere difficile definire come i parametri che determinano l'autonomia si manifestino nell'ambiente operativo, così come la creazione di scenari di test che dipendono dalla natura dell'autonomia.

8.3 Testing per bias algoritmici, di campionamento e inappropriati

Un sistema di ML dovrebbe essere valutato rispetto ai diversi bias e alle azioni intraprese per rimuovere i bias inappropriati. Ciò può comportare l'introduzione deliberata di bias positivi per contrastare i bias inappropriati.

Il testing con un set di dati indipendente può spesso rilevare i bias. Tuttavia, può essere difficile identificare tutti i dati che causano bias perché l'algoritmo di ML può utilizzare combinazioni di caratteristiche apparentemente non correlate per creare bias indesiderati.

I sistemi basati su AI dovrebbero essere testati per bias algoritmici, bias di campionamento e bias inappropriati (vedi Sezione 2.4). Questo può comportare:

- Analisi durante le attività di addestramento, valutazione e tuning del modello per identificare se sono presenti bias algoritmici.

© International Software Testing Qualifications Board

- Revisionare la fonte dei dati di addestramento e i processi utilizzati per acquisirli, in modo da poter identificare la presenza di bias di campionamento.
- Revisionare la pre-elaborazione dei dati come parte del flusso di lavoro di ML per identificare se i dati sono stati influenzati in un modo che potrebbe causare bias di campionamento.
- Misurare come i cambiamenti negli input del sistema influiscono sugli output del sistema su un gran numero di interazioni ed esaminare i risultati in base ai gruppi di persone o oggetti verso i quali il sistema potrebbe presentare bias inappropriati o contrari. Questo è simile al metodo LIME (Local Interpretable Model-Agnostic Explanations) discusso in 8.6 e può essere eseguito in un ambiente di produzione così come parte del testing prima del rilascio.
- Ottenere ulteriori informazioni sugli attributi dei dati di input potenzialmente legati a bias e correlarli ai risultati. Questo potrebbe riguardare i dati demografici, ad esempio, che potrebbero essere appropriati quando si testa un bias inappropriato che colpisce gruppi di persone, in cui l'appartenenza a un gruppo è rilevante per valutare il bias ma non è un input del modello. Questo perché il bias può essere basato su variabili "nascoste" che non sono esplicitamente presenti nei dati di input, ma sono dedotte dall'algoritmo.

8.4 Sfide nel testing dei sistemi basati su AI probabilistici e non deterministici

La maggior parte dei sistemi probabilistici sono anche non-deterministici; quindi, il seguente elenco di sfide del testing si applica in genere ai sistemi basati su AI con uno qualsiasi di questi attributi:

- Potrebbero esserci più esiti validi di un test con lo stesso insieme di precondizioni e input. Questo rende più impegnativa la definizione dei risultati attesi e può causare difficoltà:
    - quando i test vengono riutilizzati per il testing confermativo.
    - quando i test vengono riutilizzati per il testing di regressione.
    - dove la riproducibilità del testing è importante.
    - quando i test sono automatizzati.
- Il tester in genere richiede una conoscenza più approfondita del comportamento richiesto del sistema in modo da poter fornire controlli ragionevoli per verificare se il test è stato superato, piuttosto che semplicemente indicare un valore esatto per il risultato atteso del test. Ad esempio, i tester potrebbero dover definire risultati attesi più sofisticati rispetto ai sistemi convenzionali. Questi risultati di test attesi possono includere tolleranze (ad esempio, "il risultato effettivo è entro il 2% della soluzione ottimale?").
- Laddove un unico output definitivo da un test non è possibile a causa della natura probabilistica del sistema, è spesso necessario che il tester esegua più volte un test per generare un risultato del test statisticamente valido.

8.5 Sfide nel testing dei sistemi complessi basati su AI

I sistemi basati su AI vengono spesso utilizzati per implementare compiti troppo complessi per essere eseguiti dagli esseri umani. Questo può portare ad un problema con l'oracolo di test perché i tester non sono in grado di determinare i risultati attesi come farebbero normalmente (vedi Sezione 8.7). Ad esempio, i sistemi basati su AI vengono spesso utilizzati per identificare i patterns in grandi volumi di dati. Tali sistemi vengono utilizzati perché possono trovare patterns che gli umani, anche dopo molte analisi, semplicemente non possono trovare manualmente. Comprendere il comportamento richiesto

© International Software Testing Qualifications Board

di tali sistemi in modo sufficientemente approfondito per essere in grado di generare i risultati attesi può essere difficile.

Un problema simile sorge quando la struttura interna di un sistema basato su AI è generata da un software, rendendolo troppo complesso per essere compreso dagli umani. Questo porta alla situazione in cui il sistema basato su AI può essere testato solo come black box. Anche quando la struttura interna è visibile, ciò non fornisce ulteriori informazioni utili per aiutare con il testing.

La complessità dei sistemi basati su AI aumenta quando forniscono risultati probabilistici e sono di natura non deterministica (vedi Sezione 8.4).

I problemi con i sistemi non deterministici sono acutizzati quando un sistema basato su AI è costituito da diversi componenti che interagiscono ed ognuno di questi fornisce risultati probabilistici. Ad esempio, è probabile che un sistema di riconoscimento facciale utilizzi un modello per identificare un volto all'interno di un'immagine e un secondo modello per riconoscere quale volto è stato identificato. Le interazioni tra i componenti AI possono essere complesse e difficili da comprendere, rendendo difficile identificare tutti i rischi e progettare test che verifichino adeguatamente il sistema.

8.6 Testare la trasparenza, l'interpretabilità e la spiegabilità dei sistemi basati su AI

Le informazioni su come è stato implementato il sistema possono essere fornite dagli sviluppatori del sistema. Queste possono includere le fonti dei dati di addestramento, come è stata condotta l'etichettatura e come sono stati progettati i componenti del sistema. Quando queste informazioni non sono disponibili, possono rendere difficile la progettazione dei test. Ad esempio, se le informazioni sui dati di addestramento non sono disponibili, diventa difficile identificare potenziali lacune in tali dati e testare l'impatto di quelle lacune. Questa situazione può essere paragonata al testing black-box e white-box e presenta vantaggi e svantaggi simili. La **trasparenza** può essere testata confrontando le informazioni documentate sui dati e sull'algoritmo con l'implementazione effettiva e determinando quanto strettamente corrispondano.

Con il ML, spesso può essere più difficile spiegare il collegamento tra un input specifico e un output specifico, rispetto ai sistemi convenzionali. Questo basso livello di **spiegabilità** è principalmente dovuto al fatto che il modello che genera l'output è esso stesso generato dal codice (l'algoritmo) e non riflette il modo in cui gli esseri umani pensano a un problema. Diversi modelli di ML forniscono diversi livelli di spiegabilità e dovrebbero essere selezionati in base ai requisiti del sistema, che possono includere spiegabilità e testabilità.

Un metodo per comprendere la spiegabilità è attraverso il testing dinamico del modello di ML quando si applicano perturbazioni ai dati di test. Esistono metodi per quantificare la spiegabilità in questo modo e per fornire spiegazioni visive di essa. Alcuni di questi metodi sono indipendenti dal modello, mentre altri sono specifici di un particolare tipo di modello e richiedono l'accesso ad esso. Il testing esplorativo può anche essere utilizzato per comprendere meglio la relazione tra gli input e gli output di un modello.

Il metodo LIME è indipendente dal modello e utilizza perturbazioni di input iniettate dinamicamente e l'analisi degli output per fornire ai tester una visione della relazione tra input e output. Può essere un metodo efficace per fornire la spiegabilità del modello. Tuttavia, si limita a fornire possibili ragioni per gli output, piuttosto che una ragione definitiva, e non è applicabile a tutti i tipi di algoritmi.

L'**interpretabilità** di un sistema basato su AI dipende fortemente su chi questo viene applicato. Diversi stakeholder possono avere diversi requisiti in termini di quanto bene devono comprendere la tecnologia sottostante.

© International Software Testing Qualifications Board

Misurare e testare il livello di comprensione -sia per l'interpretabilità che per la spiegabilità- può essere difficile poiché gli stakeholder variano nei loro livelli di abilità e potrebbero quindi non essere d'accordo. Inoltre, l'identificazione del profilo degli stakeholder tipici può essere difficile per molti tipi di sistemi. Laddove eseguito, questo testing assume generalmente la forma di sondaggi e/o questionari per gli utenti.

8.6.1 Esercizio pratico: Spiegabilità del modello

Utilizzate uno strumento appropriato per fornire spiegabilità sulla base del modello creato in precedenza. Ad esempio, per un modello di classificazione delle immagini o un modello di classificazione del testo, può essere appropriato un metodo indipendente dal modello, come LIME.

Gli studenti dovrebbero utilizzare lo strumento per generare spiegazioni delle decisioni del modello; in particolare, come le caratteristiche negli input influenzano gli output.

8.7 Oracoli di test per sistemi basati su AI

Un grosso problema con il testing dei sistemi basati su AI può essere la specifica dei risultati attesi. Un **oracolo di test** è la fonte utilizzata per determinare il risultato atteso di un test [I01]. Una sfida nel determinare i risultati attesi è nota come il problema dell'oracolo di test.

Con sistemi complessi, non deterministici o probabilistici, può essere difficile stabilire un oracolo di test senza conoscere la **"verità di base"** (cioè, il risultato effettivo nel mondo reale che il sistema basato su AI sta cercando di prevedere). Questa "verità di base" è distinta da un oracolo di test, in quanto un oracolo di test potrebbe non fornire necessariamente un valore atteso, ma solo un meccanismo con cui determinare se il sistema sta funzionando correttamente o meno.

I sistemi basati su AI possono **evolversi** (vedi Sezione 2.3), e anche il testing dei sistemi di autoapprendimento (vedi Sezione 8.1) può soffrire di problemi dell'oracolo di test, poiché si modifica da solo e può quindi rendere necessario aggiornare frequentemente le aspettative funzionali del sistema.

Un'ulteriore causa di difficoltà nell'ottenere un oracolo di test efficace è che in molti casi la correttezza del comportamento del software è **soggettiva**. Gli assistenti virtuali (ad esempio, Siri e Alexa) sono un esempio di questo problema in quanto utenti diversi hanno spesso aspettative abbastanza diverse e possono ottenere risultati diversi a seconda della scelta delle parole e della chiarezza del discorso.

In alcune situazioni, può essere possibile definire il risultato atteso con **limiti o tolleranze**. Ad esempio, il punto di arresto per un'auto autonoma potrebbe essere definito entro una distanza massima da un punto specifico. Nell'ambito dei sistemi esperti, la determinazione dei risultati attesi può essere ottenuta consultando un **esperto** (tenendo presente che il parere dell'esperto potrebbe comunque essere errato). Ci sono diversi fattori importanti da considerare in tali circostanze:

- Gli esperti umani variano nei loro livelli di competenza. Gli esperti coinvolti devono essere almeno tanto competenti quanto gli esperti che il sistema intende sostituire.
- Gli esperti potrebbero non essere d'accordo tra loro, anche quando vengono presentate le stesse informazioni.
- Gli esperti umani potrebbero non approvare l'automazione del loro giudizio. In questi casi la loro valutazione dei potenziali output dovrebbe essere in doppio cieco (cioè, né gli esperti né i valutatori degli output dovrebbero sapere quali valutazioni sono state automatizzate).

© International Software Testing Qualifications Board

- È più probabile che gli esseri umani si mettano in guardia con le risposte (ad esempio, con frasi come "Non ne sono sicuro, ma..."). Se questo tipo di avvertimento non fosse disponibile per il sistema basato su AI, se ne dovrebbe tenere conto quando si confrontano le risposte.

Esistono **tecniche di test** che possono alleviare il problema dell'oracolo di test, come il testing A/B (vedi Sezione 9.4), il testing back-to-back (vedi Sezione 9.3) e il testing metamorfico (vedi Sezione 9.5).

8.8 Obiettivi dei test e criteri di accettazione

Gli obiettivi dei test e i criteri di accettazione di un sistema devono essere basati sui rischi percepiti del prodotto. Questi rischi possono essere spesso identificati da un'analisi delle caratteristiche di qualità richieste. Le caratteristiche di qualità per un sistema basato su AI includono quelle tradizionalmente considerate nella ISO/IEC 25010 [S06] (cioè, adeguatezza funzionale, efficienza delle prestazioni, compatibilità, usabilità, affidabilità, sicurezza, manutenibilità e portabilità) ma dovrebbero anche includere una considerazione dei seguenti aspetti:

Aspetto	**Criteri di accettazione**
Adattabilità	• Verificare che il sistema funzioni ancora correttamente e soddisfi i requisiti non funzionali quando si adatta a un cambiamento nel suo ambiente. Questo può essere implementato come una forma di testing di regressione automatizzato. • Verificare il tempo impiegato dal sistema per adattarsi a un cambiamento nel suo ambiente. • Verificare le risorse utilizzate quando il sistema si adatta a un cambiamento nel suo ambiente.
Flessibilità	• Considerare come il sistema si comporta in contesti al di fuori della specifica iniziale. Questo può essere implementato come una forma di testing di regressione automatizzato eseguito nell'ambiente operativo modificato. • Verificare il tempo impiegato dal sistema e/o le risorse utilizzate per modificarsi per gestire un nuovo contesto.
Evoluzione	• Verificare quanto bene il sistema apprende dalla propria esperienza. • Verificare quanto bene si comporta il sistema quando il profilo dei dati cambia (cioè, la deriva concettuale).
Autonomia	• Verificare come risponde il sistema quando viene forzato al di fuori dell'ambito operativo in cui si prevede che sia completamente autonomo. • Verificare se il sistema può essere "persuaso" a richiedere l'intervento umano quando dovrebbe essere completamente autonomo.
Trasparenza, interpretabilità e spiegabilità	• Verificare la trasparenza revisionando la facilità di accesso all'algoritmo e al set di dati.

© International Software Testing Qualifications Board

Aspetto	Criteri di accettazione
	• Verificare l'interpretabilità e la spiegabilità interrogando gli utenti del sistema o, se gli utenti effettivi del sistema non sono disponibili, persone con un background simile.
Libertà da bias inappropriati	• Laddove è probabile che i sistemi siano interessati da bias, questo può essere testato utilizzando una suite di test indipendente e priva di bias, o utilizzando revisori esperti. • Confrontare i risultati di test utilizzando dati esterni come i dati del censimento per verificare la presenza di bias indesiderati sulle variabili dedotte (testing di validità esterna).
Etica	• Verificare il sistema rispetto a una checklist adeguata, come la EC Assessment List for Trustworthy Artificial Intelligence [R21], che supporta i requisiti chiave delineati dalle Ethics Guidelines for Trustworthy Artificial Intelligence (AI) [R22].
Sistemi probabilistici e sistemi non deterministici	• Non può essere valutato con criteri di accettazione precisi. Se funzionasse correttamente, il sistema potrebbe restituire risultati leggermente diversi per gli stessi test.
Effetti collaterali	• Identificare gli effetti collaterali potenzialmente dannosi e tentare di generare test che inducano il sistema a mostrare questi effetti collaterali.
Reward hacking	• I test indipendenti possono identificare il reward hacking quando questi test utilizzano un metodo di misurazione del successo diverso rispetto all'agente intelligente in fase di test.
Safety	• Deve essere valutato attentamente, magari in un ambiente di test virtuale (vedi Sezione 10.2). Questo potrebbe includere tentativi di forzare un sistema a causare a sé stesso dei danni.

Per i sistemi di ML, devono essere specificate le metriche delle prestazioni funzionali di ML richieste per il modello di ML (vedi Capitolo 5).

© International Software Testing Qualifications Board

9 Metodi e tecniche per il testing di sistemi basati su AI - 245 minuti

Parole chiave

Testing A/B, adversarial testing, testing back-to-back, error guessing, testing basato sull'esperienza, testing esplorativo, relazione metamorfica (MR), testing metamorfico (MT), testing pairwise, pseudo-oracolo, problema dell'oracolo di test, tour

Parole chiave specifiche per l'AI

Adversarial attack, adversarial example, avvelenamento dei dati, sistema di ML, modello addestrato

Obiettivi di apprendimento per il Capitolo 9:

9.1 Advrsarial attacks e avvelenamento dei dati

AI-9.1.1 K2 Spiegare come il testing dei sistemi di ML può aiutare a prevenire gli adversarial attacks e l'avvelenamento dei dati.

9.2 Testing pairwise

AI-9.2.1 K2 Spiegare come viene utilizzato il testing pairwise per i sistemi basati su AI.

LO-9.2.1 H2 Applicare il testing pairwise per derivare ed eseguire casi di test per un sistema basato su AI.

9.3 Testing back-to-back

AI-9.3.1 K2 Spiegare come viene utilizzato il testing back-to-back per i sistemi basati su AI.

9.4 Testing A/B

AI-9.4.1 K2 Spiegare come viene applicato il testing A/B al testing dei sistemi basati su AI.

9.5 Testing metamorfico

AI-9.5.1 K3 Applicare il testing metamorfico per il testing dei sistemi basati su AI.

HO-9.5.1 H2 Applicare il testing metamorfico per derivare casi di test per un determinato scenario ed eseguirlo.

9.6 Testing basato sull'esperienza di sistemi basati su AI

AI-9.6.1 K2 Spiegare come il testing basato sull'esperienza può essere applicato al testing dei sistemi basati su AI.

HO-9.6.1 H2 Applicare il testing esplorativo a un sistema basato su AI.

9.7 Selezione delle tecniche di test per i sistemi basati su AI

AI-9.7.1 K4 Per un determinato scenario, selezionare le tecniche di test appropriate durante il testing di un sistema basato su AI.

© International Software Testing Qualifications Board

9.1 Adversarial attacks e avvelenamento dei dati

9.1.1 Adversarial attacks

Un **adversarial attack** è il caso in cui un attacker perturba sottilmente gli input validi che vengono passati al modello addestrato in modo che questo fornisca previsioni errate. Questi input perturbati, noti come **adversarial examples**, sono stati individuati per la prima volta con i filtri antispam, i quali modificando leggermente un'email di spam senza perdere la leggibilità, potevano essere ingannati. Recentemente gli adversarial examples, sono diventati più associati ai classificatori di immagini. Modificando semplicemente alcuni pixel che sono invisibili all'occhio umano, è possibile persuadere una rete neurale a cambiare la sua classificazione dell'immagine in un oggetto molto diverso e con un alto grado di fiducia.

Gli adversarial examples sono generalmente trasferibili [B17], il che significa che un adversarial example che causa il fallimento di un sistema di ML spesso causerà il fallimento di un altro sistema di ML addestrato a svolgere lo stesso compito. Anche nel caso in cui il secondo sistema di ML sia stato addestrato con dati diversi e sia basato su architetture diverse, spesso è ancora soggetto a guasti con gli stessi adversarial examples.

Gli **adversarial attacks white-box** sono quelli in cui l'attacker conosce quale algoritmo è stato utilizzato per addestrare il modello e anche quali impostazioni e parametri del modello sono stati utilizzati (c'è un livello ragionevole di trasparenza). L'attacker utilizza questa conoscenza per generare adversarial examples, ad esempio, apportando piccole perturbazioni negli input e monitorando quali causano grandi modifiche agli output del modello.

Gli **adversarial attacks black-box** coinvolgono l'attacker che esplora il modello per determinarne la sua funzionalità e quindi crea un modello duplicato che fornisce funzionalità simili. L'attacker usa poi un approccio white-box per identificare gli adversarial examples per questo modello duplicato. Poiché gli adversarial examples sono generalmente trasferibili, gli stessi normalmente funzioneranno anche sul modello originale.

Se non è possibile creare un modello duplicato, potrebbe essere possibile utilizzare il testing automatizzato ad alto volume per scoprire diversi adversarial examples e osservare i risultati.

L'**adversarial testing** implica semplicemente l'esecuzione di adversarial attacks allo scopo di identificare le vulnerabilità in modo da poter adottare misure preventive per proteggersi da guasti futuri. Gli adversarial examples identificati vengono aggiunti ai dati di addestramento in modo che il modello venga addestrato a riconoscerli correttamente.

9.1.2 Avvelenamento dei dati

Gli attacchi di avvelenamento dei dati sono i casi in cui un attacker manipola i dati di addestramento per ottenere uno dei seguenti obiettivi: l'attacker può inserire backdoor o trojan di reti neurali per facilitare future intrusioni, oppure (più spesso) utilizzerà dati di addestramento danneggiati (ad esempio, dati etichettati erroneamente) per indurre il modello addestrato a fornire previsioni errate.

Gli attacchi di avvelenamento possono essere **mirati**, con l'obiettivo di causare una classificazione errata del sistema di ML in situazioni specifiche. Possono anche essere **indiscriminati**, come nel caso di un attacco denial-of-service. Un noto esempio di attacco di avvelenamento è stata la corruzione del chatbot Microsoft Tay, in cui un numero relativamente piccolo di conversazioni Twitter dannose ha addestrato il sistema attraverso il feedback a fornire conversazioni contaminate in futuro. Una forma comunemente usata di attacco di avvelenamento dei dati utilizza la falsa segnalazione di milioni di email di spam come non spam nel tentativo di piegare il software del filtro antispam. Un'area

© International Software Testing Qualifications Board

di preoccupazione per l'avvelenamento dei dati è la possibilità che i set di dati AI pubblici e ampiamente utilizzati vengano avvelenati.

È possibile eseguire il **testing** per rilevare l'avvelenamento dei dati utilizzando l'EDA, poiché i dati avvelenati possono apparire come valori anomali. Inoltre, le politiche di acquisizione dei dati possono essere revisionate per garantire la provenienza dei dati di addestramento. Laddove un sistema di ML operativo possa essere attaccato fornendogli dati avvelenati, è possibile utilizzare il testing A/B (vedi Sezione 9.4) per verificare che la versione aggiornata del sistema sia ancora strettamente allineata con la versione precedente. In alternativa, anche il testing di regressione di un sistema aggiornato utilizzando una suite di test affidabile può determinare se un sistema è stato avvelenato.

9.2 Testing pairwise

Il numero di parametri di interesse per un sistema basato su AI può essere estremamente elevato, soprattutto quando il sistema utilizza big data o interagisce con il mondo esterno, come un'auto a guida autonoma. Un testing esaustivo richiederebbe di testare tutte le possibili combinazioni di questi parametri impostati su tutti i valori possibili. Tuttavia, poiché questo comporterebbe un numero praticamente infinito di test, vengono utilizzate le tecniche di test per selezionare un sottoinsieme che può essere eseguito nel tempo limitato a disposizione.

Laddove è possibile combinare numerosi parametri, ognuno dei quali può avere molti valori discreti, il testing combinatorio può essere applicato per ridurre significativamente il numero di casi di test richiesti, idealmente senza compromettere la capacità di rilevamento dei difetti della suite di test. Esistono diverse tecniche di **testing combinatorio** (vedi [I02] e [S08]). Tuttavia, nella pratica, il testing pairwise è la tecnica più utilizzata perché è di facile comprensione e ha un ampio supporto di strumenti. Inoltre, la ricerca ha dimostrato che la maggior parte dei difetti sono causati da interazioni che coinvolgono pochi parametri [B33].

Nella pratica, anche l'uso del testing pairwise può portare a suite di test estese per alcuni sistemi per cui l'uso di automazione e ambienti di test virtuali (vedi Sezione 10.2) spesso diventa necessario per consentire l'esecuzione del numero necessario di test. Ad esempio, quando si considerano le auto a guida autonoma, gli scenari di test di alto livello per il testing del sistema devono esercitare sia i diversi ambienti in cui ci si aspetta che le auto funzionino sia le varie funzioni del veicolo. Pertanto, i parametri dovrebbero includere la gamma di vincoli ambientali (ad esempio, tipi e superfici stradali, condizioni meteorologiche e di traffico e visibilità) e le varie funzioni di guida autonoma (ad esempio, cruise control adattativo, assistenza al mantenimento della corsia e assistenza al cambio di corsia). Oltre a questi parametri, gli input dei sensori potrebbero essere considerati a vari livelli di efficacia (ad esempio, gli input di una videocamera si degraderanno con l'avanzare del viaggio e quando si sporca).

La ricerca non è attualmente chiara sul livello di rigore necessario che sarebbe richiesto per l'uso del testing combinatorio con sistemi basati su AI safety-critical, come le auto a guida autonoma. Anche se il testing pairwise potrebbe non essere sufficiente, è noto che l'approccio è efficace nel trovare i difetti.

9.2.1 Esercizio pratico: Testing pairwise

Per un sistema basato su AI implementato con un minimo di cinque parametri e almeno cinquecento possibili combinazioni, utilizzate uno strumento di testing pairwise per identificare un insieme ridotto di combinazioni pairwise ed eseguite i test per queste combinazioni. Confrontate il numero di combinazioni pairwise testate con il numero richiesto se tutte le combinazioni teoricamente possibili dovessero essere testate.

© International Software Testing Qualifications Board

9.3 Testing back-to-back

Una delle potenziali soluzioni al problema dell'oracolo di test (vedi Sezione 8.7), quando si esegue il testing di sistemi basati su AI, consiste nell'utilizzare il testing back-to-back. Questo è anche noto come **testing differenziale**. Con il testing back-to-back, una versione alternativa del sistema viene utilizzata come pseudo-oracolo e i suoi output vengono confrontati con i risultati dei test prodotti dal SUT. Lo pseudo-oracolo potrebbe essere un sistema esistente, oppure potrebbe essere sviluppato da un team diverso, possibilmente su una piattaforma diversa, con un'architettura diversa e con un linguaggio di programmazione diverso. Quando si esegue il testing dell'adeguatezza funzionale (al contrario dei requisiti non funzionali), il sistema utilizzato come pseudo-oracolo non è vincolato a raggiungere gli stessi criteri di accettazione non funzionali del SUT. Ad esempio, potrebbe non dover essere eseguito così velocemente, nel qual caso può essere molto meno costoso da costruire.

Nel contesto del ML, è possibile utilizzare diversi framework, algoritmi e impostazioni del modello per creare uno pseudo-oracolo del ML. In alcune situazioni, potrebbe anche essere possibile creare uno pseudo-oracolo utilizzando un software convenzionale, non-AI.

Affinché gli pseudo-oracoli siano efficaci nel rilevare i difetti, non dovrebbe esserci un software comune sia nello pseudo-oracolo che nel SUT. In caso contrario, è possibile che lo stesso difetto in entrambi faccia coincidere i risultati dei due test quando entrambi sono difettosi. La presenza di un grande quantità di software AI immaturo, riutilizzabile e open-source adoperato per sviluppare sistemi basati su AI, oltre che il riutilizzo del codice tra lo pseudo-oracolo e il SUT, può compromettere lo pseudo-oracolo. Questa problematica è inoltre difficilmente individuabile a causa della scarsa documentazione delle soluzioni AI riutilizzabili.

9.4 Testing A/B

Il testing A/B è un metodo in cui la risposta di due varianti del programma (A e B) agli stessi input viene confrontata allo scopo di determinare quale delle due varianti è migliore. È un approccio di **testing statistico** che in genere richiede il confronto dei risultati dei test di diverse esecuzioni di test per determinare la differenza tra i programmi.

Un semplice esempio di questo metodo è quando due offerte promozionali vengono inviate tramite e-mail a un elenco di marketing diviso in due gruppi. Metà della lista riceve l'offerta A, metà riceve l'offerta B e il successo di ogni offerta aiuta a decidere quale utilizzare in futuro. Molte aziende di e-commerce e web-based usano il testing A/B in produzione, deviando diversi consumatori verso diverse funzionalità, per aiutare a identificare le preferenze dei consumatori.

Il testing A/B è un approccio per risolvere il problema dell'oracolo di test in cui il sistema esistente viene utilizzato come un oracolo parziale. Il testing A/B non genera casi di test e non fornisce indicazioni su come dovrebbero essere progettati i test, sebbene gli input operativi siano spesso utilizzati nei test.

Il testing A/B può essere utilizzato per testare gli aggiornamenti di un sistema basato su AI in cui esistono criteri di accettazione concordati, quali le metriche delle prestazioni funzionali del ML (come descritto nel Capitolo 5). Ogni volta che il sistema viene aggiornato, il testing A/B viene utilizzato per verificare che la variante aggiornata abbia prestazioni pari o migliori rispetto alla variante precedente. Tale approccio può essere usato per un semplice classificatore, ma può essere utilizzato anche per testare sistemi molto più complessi. Ad esempio, un aggiornamento per migliorare l'efficacia di un sistema di instradamento dei trasporti di una città intelligente può anche essere testato utilizzando il testing A/B (ad esempio, confrontando i tempi medi di percorrenza per due varianti del sistema in settimane consecutive).

© International Software Testing Qualifications Board

Il testing A/B può essere utilizzato anche per testare i sistemi di autoapprendimento. Quando il sistema apporta una modifica, vengono eseguiti dei test automatizzati e le caratteristiche del sistema risultanti vengono confrontate con quelle precedenti alla modifica. Se il sistema viene migliorato, la modifica viene accettata, altrimenti il sistema torna allo stato precedente.

Una delle principali differenze tra il testing A/B e il testing back-to-back riguarda l'uso del testing A/B per confrontare due varianti dello stesso sistema e l'uso del testing back-to-back per rilevare i difetti.

9.5 Testing metamorfico (MT)

Il testing metamorfico [B18] è una tecnica volta a generare **casi di test basati su un caso di test sorgente che è stato superato**. Uno o più casi di **test di follow-up** vengono generati modificando (metamorfizzando) il caso di test sorgente in base a una **relazione metamorfica (MR)**. La MR si basa su una proprietà di una funzione richiesta dell'oggetto di test, in modo tale da descrivere come una modifica negli input di test di un caso di test si rifletta nei risultati attesi dello stesso caso di test.

Considerate, ad esempio, un programma che determina la media di un insieme di numeri. Viene generato un caso di test sorgente che comprende un insieme di numeri e una media attesa e il caso di test viene eseguito per confermare il superamento. È ora possibile generare casi di test di follow-up in base a ciò che è noto sulla funzione media del programma. Inizialmente, l'ordine dei numeri di cui determinare la media può essere modificato semplicemente. Data la funzione media, si può prevedere che il risultato atteso rimanga lo stesso. Pertanto, è possibile generare un caso di test di follow-up con i numeri in un ordine diverso senza dover calcolare il risultato atteso. Con un grande insieme di numeri, questo potrebbe portare alla generazione di un gran numero di diversi insiemi di numeri in cui gli stessi numeri vengono utilizzati in sequenze diverse e ognuno di essi potrebbe essere utilizzato per creare un caso di test di follow-up separato. Tutti questi casi di test sarebbero basati sullo stesso caso di test sorgente e avrebbero lo stesso risultato atteso.

È comune avere MR e casi di test di follow-up in cui il risultato atteso è diverso dal risultato atteso originale del caso di test sorgente. Ad esempio, utilizzando la stessa funzione media, è possibile derivare una MR in cui ogni elemento dell'insieme di input viene moltiplicato per due. Il risultato atteso per un tale insieme è semplicemente il risultato atteso originale moltiplicato per due. Allo stesso modo, qualsiasi altro valore potrebbe essere utilizzato come moltiplicatore per generare potenzialmente un numero infinito di casi di test di follow-up basati su questa MR.

Il MT può essere utilizzato per la maggior parte degli oggetti di test e può essere applicato sia al testing funzionale che non funzionale (ad esempio, il testing di installabilità copre diverse configurazioni di destinazione in cui i parametri di installazione possono essere selezionati in sequenze diverse). È particolarmente utile quando la generazione dei risultati attesi è problematica, a causa della mancanza di un oracolo di test poco costoso. Questo è il caso di alcuni sistemi basati su AI che si basano sull'analisi di big data, o di quelli in cui i tester non sono chiari su come l'algoritmo di ML deriva le sue previsioni. Nel campo dell'AI, il MT è stato utilizzato per testare, tra gli altri, il riconoscimento delle immagini, i motori di ricerca, l'ottimizzazione dei percorsi e il riconoscimento vocale.

Come spiegato in precedenza, il MT può essere basato su un caso di test sorgente superato, ma può anche essere utilizzato quandonon sia possibile verificare che un -qualsiasi- caso di test sorgente sia corretto. Questo può essere il caso, ad esempio, di un programma che implementa una funzione troppo complessa affinchè un tester umano possa replicarla e utilizzarla come oracolo di test, come nel caso di alcuni sistemi basati su AI. In questa situazione, il MT può essere utilizzato per generare uno o più casi di test che, una volta eseguiti, creeranno un insieme di output in cui è possibile verificare la validità delle relazioni tra gli output. Con questa forma di MT, i singoli test non sono noti per essere corretti, ma le relazioni tra di loro devono essere vere, fornendo così una maggiore fiducia nel programma. Un esempio potrebbe essere un programma attuariale basato su AI che prevede

© International Software Testing Qualifications Board

un'età alla morte sulla base di un ampio insieme di dati, dove è noto, ad esempio, che se il numero di sigarette fumate aumentasse, l'età prevista alla morte dovrebbe diminuire (o, almeno, rimanere la stessa).

Il MT è una tecnica di test relativamente nuova, proposta per la prima volta nel 1998. Si differenzia dalle tecniche di test tradizionali in quanto i risultati attesi del caso di test di follow-up non sono descritti in termini di valori assoluti, bensì sono relativi ai risultati attesi nel caso di test sorgente. Si basa su un concetto facilmente comprensibile, può essere applicato da tester con poca esperienza nell'applicazione della tecnica ma che ne comprendono il dominio applicativo e ha costi simili rispetto alle tecniche tradizionali. È anche efficace nel rivelare i difetti, con la ricerca che dimostra che solo da tre a sei diverse MR possono rivelare oltre il 90% dei difetti che potrebbero essere rilevati utilizzando tecniche basate su un oracolo di test tradizionale [B19]. È possibile generare automaticamente casi di test di follow-up da MR ben specificate e da un caso di test sorgente. Tuttavia, gli strumenti commerciali non sono attualmente disponibili, sebbene Google stia già applicando il MT automatizzato per testare i driver grafici Android utilizzando lo strumento GraphicsFuzz, che è stato reso open source (vedi [R23]).

9.5.1 Esercizio pratico: Testing metamorfico

In questo esercizio, gli studenti acquisiranno esperienza pratica di quanto segue:

- Derivare diverse relazioni metamorfiche (MR) per una data applicazione o programma basato su AI. Queste MR dovrebbero includerne alcune in cui i risultati attesi dei casi di test sorgente e di follow-up sono gli stessi e alcune in cui sono diversi.
- Generazione di casi di test sorgente per l'applicazione o il programma basato su AI. Questi non devono essere garantiti per il superamento, ma agli studenti dovrebbero essere ricordati i limiti del MT dove non è disponibile un tale "gold standard".
- Utilizzo delle MR derivate e dei casi di test sorgente generati per derivare i casi di test di follow-up.
- Esecuzione dei casi di test di follow-up.

9.6 Testing basato sull'esperienza di sistemi basati su AI

Il testing basato sull'esperienza include l'error guessing, il testing esplorativo e il testing checklist-based [I01], che possono essere tutti applicati al testing di sistemi basati su AI.

L'**error guessing** si basa in genere sulla conoscenza dei tester, sugli errori tipici degli sviluppatori e sugli errori in sistemi simili (o versioni precedenti). Un esempio di error guessing applicato ai sistemi basati su AI potrebbe essere l'uso della conoscenza su come i sistemi di ML non hanno funzionato in passato a causa dell'uso di dati di addestramento sistematicamente distorti (bias).

Nel **testing esplorativo**, i test vengono progettati, generati ed eseguiti in modo iterativo, con l'opportunità di derivare test successivi, in base ai risultati dei test precedenti. Il testing esplorativo è particolarmente utile quando ci sono specifiche scadenti o problemi dell'oracolo di test, come spesso accade per i sistemi basati su AI. Di conseguenza, il testing esplorativo è spesso usato in questo contesto e dovrebbe essere usato per integrare il testing più sistematico basato su tecniche, come il testing metamorfico (vedi Sezione 9.5).

Un tour è una metafora utilizzata per un insieme di strategie e obiettivi a cui i tester possono fare riferimento quando eseguono il testing esplorativo organizzato intorno a un focus speciale [B20]. I tour tipici per il testing esplorativo dei sistemi basati su AI potrebbero concentrarsi sui concetti di bias, underfitting e overfitting nei sistemi di ML. Ad esempio, un tour dei dati potrebbe essere applicato per

© International Software Testing Qualifications Board

testare il modello. In questo tour il tester potrebbe identificare diversi tipi di dati utilizzati per l'addestramento, la loro distribuzione, le loro variazioni, il loro formato e gli intervalli, ecc. e poi utilizzare i tipi di dati per testare il modello.

I sistemi di ML dipendono fortemente dalla qualità dei dati di addestramento e il campo esistente dell'EDA è strettamente correlato all'approccio del testing esplorativo. L'EDA è dove i dati vengono esaminati per modelli, relazioni, tendenze e valori anomali. Implica l'esplorazione interattiva e guidata da ipotesi dei dati ed è descritta in [B21] come "Esploriamo i dati con delle aspettative. Rivediamo le nostre aspettative in base a ciò che vediamo nei dati. E ripetiamo questo processo". L'EDA in genere richiede il supporto di strumenti in due aree; per l'interazione con i dati, per consentire agli analisti di comprendere meglio i dati complessi, e per la visualizzazione dei dati, per consentire loro di visualizzare facilmente i risultati delle analisi. L'uso di tecniche esplorative, guidate principalmente dalla visualizzazione dei dati, può aiutare a validare l'algoritmo di ML utilizzato, identificare i cambiamenti che si traducono in modelli efficienti e sfruttare l'esperienza del dominio [B22].

Google ha una serie di ventotto test del ML scritti come asserzioni, nelle aree di dati, sviluppo del modello, infrastruttura e monitoraggio, che viene utilizzata come **checklist** di test all'interno di Google per i sistemi di ML [B23]. La "checklist di test del ML" di Google è presentata qui come pubblicata da Google:

Dati ML:

1. Le aspettative delle caratteristiche vengono acquisite in uno schema.
2. Tutte le caratteristiche sono vantaggiose.
3. Nessuna caratteristica è troppo costosa.
4. Le caratteristiche aderiscono ai requisiti del metalivello.
5. La pipeline di dati dispone di controlli sulla privacy appropriati.
6. Nuove funzionalità possono essere aggiunte rapidamente.
7. Tutto il codice delle funzioni di input è testato.

Sviluppo del modello:

1. Le specifiche del modello vengono revisionate ed emesse.
2. Le metriche offline e online sono correlate.
3. Tutti gli iperparametri sono stati regolati.
4. L'impatto dell'obsolescenza del modello è noto.
5. Un modello più semplice non è migliore.
6. La qualità del modello è sufficiente su porzioni di dati importanti.
7. Il modello è testato per considerazioni di inclusione.

Infrastruttura ML:

1. L'addestramento è riproducibile.
2. Le specifiche del modello sono testate per unità.
3. La pipeline del ML è testata per l'integrazione.
4. La qualità del modello viene validata prima di essere messo in produzione.
5. È possibile eseguire il debug del modello.
6. I modelli vengono testati tramite un processo canary prima di andare in produzione.

© International Software Testing Qualifications Board

7. È possibile eseguire il rollback dei modelli in produzione.

Test di monitoraggio:

1. Le modifiche alle dipendenze generano una notifica.
2. Le invarianti dei dati sono valide per gli input.
3. L'addestramento e la messa in produzione non sono distorti.
4. I modelli non sono troppo obsoleti.
5. I modelli sono numericamente stabili.
6. Le prestazioni di calcolo non sono regredite.
7. La qualità delle previsioni non è regredita.

9.6.1 Esercizio pratico: Testing esplorativo e Exploratory Data Analysis (EDA)

Per un modello e un set di dati selezionati, gli studenti eseguiranno un tour dei dati, considerando vari tipi di dati e la loro distribuzione per vari parametri.

Gli studenti eseguiranno l'EDA sui dati per identificare i dati mancanti e/o i potenziali bias nei dati.

9.7 Selezione delle tecniche di test per i sistemi basati su AI

Un sistema basato su AI includerà in genere sia componenti AI che non-AI. La selezione delle tecniche di test per testare i componenti non-AI è generalmente la stessa di qualsiasi testing convenzionale. Per i componenti basati su AI, la scelta può essere più limitata. Ad esempio, quando viene percepito un problema dell'oracolo di test (cioè, è difficile generare i risultati attesi), allora, in base ai rischi percepiti, è possibile mitigare questo problema utilizzando quanto segue:

- **Testing back-to-back**: Richiede che i casi di test siano disponibili o generati e che un sistema equivalente funga da pseudo-oracolo, che per il testing di regressione può essere una versione precedente del sistema. Per un rilevamento efficace dei difetti, potrebbe essere necessario un sistema sviluppato in modo indipendente.
- **Testing A/B**: Spesso utilizza input operativi come casi di test e viene normalmente utilizzato per confrontare due varianti dello stesso sistema utilizzando l'analisi statistica. Il testing A/B può essere utilizzato per verificare l'avvelenamento dei dati di una nuova variante, o per il testing di regressione automatizzato di un sistema di autoapprendimento.
- **Testing metamorfico**: Può essere utilizzato da tester inesperti per trovare i difetti in modo economico, anche se devono comprendere il dominio dell'applicazione. Il MT non è idoneo a fornire risultati definitivi in quanto i risultati attesi non sono assoluti, ma, invece, relativi ai casi di test sorgente. Il supporto di strumenti commerciali non è attualmente disponibile, ma molti test possono essere generati manualmente.

L'**adversarial testing** è in genere appropriato per i modelli di ML in cui la cattiva gestione degli adversarial examples potrebbe avere un impatto significativo o dove il sistema potrebbe essere attaccato. Allo stesso modo, il testing per l'**avvelenamento dei dati** può essere appropriato per i sistemi di ML dove il sistema potrebbe essere attaccato.

Quando i sistemi basati su AI sono complessi e hanno più parametri, il **testing pairwise** è spesso appropriato.

© International Software Testing Qualifications Board

Il **testing basato sull'esperienza** è spesso adatto per il testing dei sistemi basati su AI, in particolare per la considerazione dei dati utilizzati per l'addestramento e dei dati operativi. L'**EDA** può essere utilizzata per validare l'algoritmo di ML utilizzato, identificare miglioramenti dell'efficienza e sfruttare l'esperienza del dominio. Google ha scoperto che la loro **checklist** di test del ML è un approccio efficace per i sistemi di ML.

Nell'area specifica delle reti neurali, la copertura della rete è spesso adatta ai sistemi mission-critical, con alcuni criteri di copertura che richiedono una copertura più rigorosa di altri.

© International Software Testing Qualifications Board

10 Ambienti di test per sistemi basati su AI - 30 minuti

Parole chiave

Ambiente di test virtuale

Parole chiave specifiche per l'AI

Processore specifico per l'AI, sistema autonomo, big data, spiegabilità, sistema multi-agente, sistema di autoapprendimento

Obiettivi di apprendimento per il Capitolo 10:

10.1 Ambienti di test per i sistemi basati su AI

AI-10.1.1 K2 Descrivere i principali fattori che differenziano gli ambienti di test per i sistemi basati su AI da quelli richiesti per i sistemi convenzionali.

10.2 Ambienti di test virtuali per il testing dei sistemi basati su AI

AI-10.2.1 K2 Descrivere i benefici forniti dagli ambienti di test virtuali nel testing dei sistemi basati su AI.

© International Software Testing Qualifications Board

10.1 Ambienti di test per i sistemi basati su AI

I sistemi basati su AI possono essere utilizzati in un'ampia varietà di ambienti operativi, il che significa che gli ambienti di test sono altrettanto diversi. Le caratteristiche dei sistemi basati su AI che possono far sì che gli ambienti di test differiscano da quelli dei sistemi convenzionali includono:

- **Autoapprendimento**: Ci si aspetta che i sistemi di autoapprendimento e alcuni sistemi autonomi si adattino ad ambienti operativi mutevoli che potrebbero non essere stati completamente definiti quando il sistema è stato inizialmente distribuito (vedi Sezione 2.1). Di conseguenza, definire ambienti di test in grado di imitare questi cambiamenti ambientali indefiniti è intrinsecamente difficile e può richiedere sia l'immaginazione da parte dei tester che un livello di casualità integrato nell'ambiente di test.
- **Autonomia**: Ci si aspetta che i sistemi autonomi rispondano ai cambiamenti nel loro ambiente senza l'intervento umano e che riconoscano anche le situazioni in cui l'autonomia dovrebbe essere ceduta di nuovo agli operatori umani (vedi Sezione 2.2). Per alcuni sistemi, l'identificazione, e quindi l'imitazione delle circostanze per la cessione dell'autonomia, può richiedere che gli ambienti di test spingano i sistemi all'estremo. Alcuni sistemi autonomi hanno come scopo quello di lavorare in ambienti pericolosi e la creazione di ambienti di test pericolosi e rappresentativi della realtà può essere difficile.
- **Multi-agente**: Quando ci si aspetta che i sistemi basati su AI multi-agente funzionino in sinergia con altri sistemi basati su AI, l'ambiente di test potrebbe dover incorporare un livello di non-determinismo, facendo sì da poter imitare il non-determinismo dei sistemi basati su AI con cui interagisce il SUT.
- **Spiegabilità**: La natura di alcuni sistemi basati su AI può rendere difficile determinare come il sistema ha preso le sue decisioni (vedi Sezione 2.7). Se fosse importante comprendere questo aspetto prima della distribuzione, l'ambiente di test dovrebbe incorporare strumenti il cui fine è spiegare come le decisioni vengono prese.
- **Hardware**: Parte dell'hardware utilizzato per ospitare sistemi basati su AI è specificamente progettato per questo scopo, come i processori specifici per l'AI (vedi Sezione 1.6). La necessità di includere tale hardware nell'ambiente di test dovrebbe essere considerata come parte della relativa pianificazione dei test.
- **Big data**: Quando ci si aspetta che un sistema basato su AI consumi big data (ad esempio, dati ad alto volume, ad alta velocità e/o ad alta varietà), la configurazione come parte di un ambiente di test richiede un'attenta pianificazione e implementazione (vedi Sezione 7.3).

10.2 Ambienti di test virtuali per il testing dei sistemi basati su AI

L'uso di un ambiente di test virtuale durante il testing di un sistema basato su AI offre i seguenti vantaggi:

- **Scenari pericolosi**: Possono essere testati senza mettere in pericolo il SUT, altri sistemi interagenti, inclusi gli esseri umani, o l'ambiente operativo (ad esempio, alberi, edifici).
- **Scenari insoliti**: Possono essere testati quando altrimenti sarebbe molto lungo o costoso impostare questi scenari per operazioni reali (ad esempio, in attesa di un evento raro, come un'eclissi solare completa o quattro autobus che entrano contemporaneamente nello stesso incrocio stradale). Allo stesso modo, i casi limite, difficili da creare nel mondo reale, possono essere creati più facilmente, ripetutamente e in modo riproducibile in un ambiente di test virtuale.

© International Software Testing Qualifications Board

- **Scenari estremi**: Possono essere testati quando sarebbe costoso o impossibile configurarli nella realtà (ad esempio, per un disastro nucleare o per l'esplorazione dello spazio profondo).
- **Scenari che richiedono molto tempo**: Possono essere testati in tempi ridotti (ad esempio, più volte al secondo) in un ambiente virtuale. Al contrario, l'installazione e l'esecuzione in tempo reale potrebbero richiedere ore o giorni. Un ulteriore vantaggio è che più ambienti di test virtuali possono essere eseguiti in parallelo. Questo avviene in genere nel cloud e consente l'esecuzione simultanea di molti scenari, cosa che potrebbe non essere possibile utilizzando l'hardware del sistema effettivo.
- **Osservabilità e controllabilità**: Gli ambienti di test virtuali forniscono una maggiore controllabilità dell'ambiente di test. Ad esempio, possono garantire che un insieme insolito di condizioni di trading finanziario venga replicato. Inoltre, offrono un'osservabilità di gran lunga migliore, poiché tutte le parti dell'ambiente fornite digitalmente possono essere continuamente monitorate e registrate.
- **Disponibilità**: La simulazione dell'hardware tramite ambienti di test virtuali consente di testare i sistemi con componenti hardware (simulati) che altrimenti potrebbero non essere disponibili, forse perché non sono stati ancora sviluppati o sono troppo costosi.

Gli ambienti di test virtuali possono essere creati specificamente per un determinato sistema, possono essere generici o possono essere sviluppati per supportare domini di applicazione specifici. Sono disponibili ambienti di test virtuali sia commerciali che open-source per supportare il testing dei sistemi basati su AI. Alcuni esempi sono:

- **Morse**: Il Modular Open Robots Simulation Engine, è un simulatore per la simulazione generica di robot mobili singoli o multipli, basato sul motore di gioco Blender [R24].
- **AI Habitat**: È una piattaforma di simulazione creata da Facebook AI, progettata per addestrare agenti embodied (come i robot virtuali) in ambienti 3D fotorealistici [R25].
- **DRIVE Constellation**: È una piattaforma aperta e scalabile per auto a guida autonoma di NVIDIA. Si basa su una piattaforma basata su cloud ed è in grado di generare miliardi di miglia di testing per veicoli autonomi [R26].
- **MATLAB** e **Simulink**: Forniscono la capacità di preparare dati di addestramento, produrre modelli di ML e simulare l'esecuzione di sistemi basati su AI, inclusi i modelli che utilizzano dati sintetici [R27].

© International Software Testing Qualifications Board

11 Usare l'AI per il testing - 195 minuti

Parole chiave

Testing visivo

Parole chiave specifiche per l'AI

Tecniche bayesiane, classificazione, algoritmo di clustering, previsione dei difetti, Graphical User Interface (GUI)

Obiettivi di apprendimento per il Capitolo 11:

11.1 Tecnologie di AI per il testing

AI-11.1.1 K2 Classificare le tecnologie di AI utilizzate nel testing del software.

HO-11.1.1 H2 Discutere, usando esempi, quelle attività nel testing in cui è meno probabile che l'AI venga utilizzata.

11.2 Utilizzare l'AI per analizzare i difetti segnalati

AI-11.2.1 K2 Spiegare come l'AI può aiutare a supportare l'analisi di nuovi difetti.

11.3 Utilizzare l'AI per la generazione di casi di test

AI-11.3.1 K2 Spiegare come l'IA può aiutare nella generazione di casi di test.

11.4 Utilizzare l'AI per l'ottimizzazione delle suite di test di regressione

AI-11.4.1 K2 Spiegare come l'AI può aiutare nell'ottimizzazione delle suite di test di regressione.

11.5 Utilizzare l'AI per la previsione dei difetti

AI-11.5.1 K2 Spiegare come l'AI può aiutare nella previsione dei difetti.

HO-11.5.1 H2 Implementare un semplice sistema di previsione dei difetti basato su AI.

11.6 Utilizzare l'AI per il testing delle interfacce utente

AI-11.6.1 K2 Spiegare l'uso dell'AI nel testing delle interfacce utente.

© International Software Testing Qualifications Board

11.1 Tecnologie di AI per il testing

Diverse tecnologie di AI sono elencate nella Sezione 1.4, tutte utilizzabili per supportare alcuni aspetti specifici del testing del software. Secondo Harman [B24], la comunità dell'ingegneria del software utilizza tre grandi aree di tecnologie di AI:

- **Logica fuzzy** e **metodi probabilistici**: Implicano l'uso di tecniche di AI per gestire problemi del mondo reale che sono essi stessi probabilistici. Ad esempio, l'AI può essere utilizzata per analizzare e prevedere possibili guasti del sistema utilizzando tecniche Bayesiane. Queste possono stimare la probabilità che componenti o funzioni si guastino, o riflettere la natura potenzialmente casuale delle interazioni umane con il sistema.
- **Classificazione**, **apprendimento** e **previsione**: Possono essere utilizzate per vari use case come la previsione dei costi nell'ambito della pianificazione del progetto o della previsione dei difetti. Incorporata dal ML, quest'area viene utilizzata per molte attività di testing del software, inclusa la gestione dei difetti (vedi Sezione 11.2), la previsione dei difetti (vedi Sezione 11.5) e il testing dell'interfaccia utente (vedi Sezione 11.6).
- **Tecniche di ricerca** e **ottimizzazione computazionale**: Possono essere utilizzate per risolvere problemi di ottimizzazione utilizzando una ricerca computazionale di spazi di ricerca potenzialmente grandi e complessi (ad esempio, utilizzando algoritmi di ricerca). Esempi includono la generazione di casi di test (vedi Sezione 11.3), l'identificazione del minor numero di casi di test che raggiunge un determinato criterio di copertura e l'ottimizzazione dei casi di test di regressione (vedi Sezione 11.4).

La categorizzazione di cui sopra è necessariamente ampia, in quanto vi è una notevole sovrapposizione tra le attività di testing che possono essere implementate dall'AI e le diverse tecnologie di AI. È comunque solo una categorizzazione, e se ne potrebbero creare altre ugualmente valide.

11.1.1 Esercizio pratico: L'uso dell'AI nel testing

Come parte di una discussione, gli studenti identificheranno le attività e i compiti di testing che sono attualmente poco pratici per l'implementazione come AI. Questi potrebbero includere:

- specificare gli oracoli di test.
- comunicare con gli stakeholder per chiarire le ambiguità e recuperare le informazioni mancanti.
- suggerire miglioramenti alla user experience.
- sfidare le ipotesi degli stakeholder e porre domande scomode.
- comprendere le esigenze degli utenti.

Occorre fare una distinzione tra AI debole, che potrebbe essere utilizzata per alcuni compiti limitati, e AI generale, che attualmente non è disponibile (vedi Sezione 1.2).

11.2 Utilizzare l'AI per analizzare i difetti segnalati

I difetti segnalati sono generalmente categorizzati, prioritizzati e tutti i duplicati vengono identificati. Questa attività viene spesso definita triage o analisi dei difetti e ha lo scopo di ottimizzare il tempo

© International Software Testing Qualifications Board

impiegato nella risoluzione dei difetti. L'AI può essere utilizzata per supportare questa attività in vari modi, come ad esempio:

- **Categorizzazione**: NLP [B25] può essere utilizzata per analizzare il testo all'interno dei report dei difetti ed estrarre argomenti, come l'area della funzionalità interessata, che possono poi essere forniti insieme ad altri metadati ad algoritmi di clustering, come k-nearest neighbors o macchine a vettore di supporto. Questi algoritmi possono identificare categorie di difetti adatte ed evidenziare difetti simili o duplicati. La categorizzazione basata su AI è particolarmente utile per i sistemi automatizzati di defect reporting (ad esempio, per Microsoft Windows e Firefox) e per progetti di grandi dimensioni con molti sviluppatori.
- **Criticità**: I modelli di ML addestrati sulle caratteristiche dei difetti più critici possono essere utilizzati per identificare quei difetti che con maggiore probabilità causano quei guasti del sistema che rappresentano un'ampia percentuale dei difetti segnalati [B26].
- **Assegnazione**: I modelli di ML possono suggerire quali sviluppatori sono più adatti per correggere particolari difetti, in base al contenuto del difetto e alle precedenti assegnazioni degli sviluppatori.

11.3 Utilizzare l'AI per la generazione di casi di test

L'uso dell'AI per generare test può essere una tecnica molto efficace per creare rapidamente risorse di testing e massimizzare la copertura (ad esempio, copertura del codice o dei requisiti). La base per la generazione di questi test include il codice sorgente, l'interfaccia utente e un modello di test leggibile dalla macchina. Alcuni strumenti basano i test anche sull'osservazione del comportamento di basso livello del sistema tramite strumentazione o file di log [B27].

Tuttavia, a meno che non venga utilizzato un modello di test che definisce i comportamenti richiesti come base dei test, questa forma di generazione di test generalmente soffre di un problema dell'oracolo di test perché lo strumento basato su AI non sa quali dovrebbero essere i risultati attesi per un determinato insieme di dati di test. Una soluzione è utilizzare il testing back-to-back se è disponibile un sistema adatto da utilizzare come pseudo-oracolo (vedi Sezione 9.3). In alternativa, i test potrebbero essere eseguiti con il risultato atteso che non si sia verificata né una "applicazione che non risponde" né un arresto anomalo del sistema, così come altri semplici indicatori di errore simili.

Gli studi finalizzati a confrontare gli strumenti di generazione di test basati su AI con strumenti di Fuzz testing non-AI simili mostrano che gli strumenti basati su AI possono raggiungere livelli di copertura equivalenti e trovare più difetti riducendo al contempo la sequenza media di passaggi necessari per causare un errore da una media di circa 15.000 passaggi a circa 100 passaggi. Questo rende il debugging molto più semplice [B27].

11.4 Utilizzare l'AI per l'ottimizzazione delle suite di test di regressione

Man mano che vengono apportate modifiche a un sistema, nuovi test vengono creati, eseguiti e diventano candidati per una suite di test di regressione. Per evitare che le suite di test di regressione diventino troppo grandi, dovrebbero essere spesso ottimizzate selezionando, prioritizzando e persino aumentando i casi di test al fine di creare una suite di test di regressione più efficace ed efficiente.

Uno strumento basato su AI può eseguire l'ottimizzazione della suite di test di regressione analizzando, ad esempio, le informazioni dai risultati dei test precedenti, i difetti associati e le ultime modifiche apportate, come le feature che si interrompono più frequentemente e quali test verificano il codice interessato dalle modifiche recenti.

© International Software Testing Qualifications Board

La ricerca mostra che è possibile ottenere riduzioni del 50% nelle dimensioni di una suite di test di regressione continuando a rilevare la maggior parte dei difetti [B28] e riduzioni del 40% nella durata di esecuzione dei test senza una riduzione significativa nel rilevamento dei guasti per il testing di Continuous Integration [B29].

11.5 Utilizzare l'AI per la previsione dei difetti

La previsione dei difetti può essere utilizzata per prevedere se è presente un difetto, quanti difetti sono presenti, o se possono essere trovati difetti. Questa capacità dipende dalla sofisticatezza dello strumento utilizzato. I risultati vengono normalmente utilizzati per prioritizzare il testing (ad esempio, più test per quei componenti in cui sono previsti più difetti).

La previsione dei difetti si basa in genere su metriche del codice sorgente, metriche di processo e/o metriche relative alle persone e all'organizzazione. Poiché ci sono così tanti potenziali fattori da considerare, determinare la relazione tra questi fattori e la probabilità di difetti va oltre le capacità umane. Di conseguenza, l'uso di un approccio basato su AI, che in genere utilizza il ML, è una necessità. La previsione dei difetti è più efficace se basata su esperienze precedenti in una situazione simile (ad esempio, con la stessa base di codice e/o gli stessi sviluppatori).

La previsione dei difetti mediante il ML è stata utilizzata con successo in diverse situazioni (ad esempio, [B30] e [B31]). È stato riscontrato che i migliori previsori sono le persone e le misure organizzative piuttosto che le metriche del codice sorgente più ampiamente utilizzate, come le linee di codice e la complessità ciclomatica [B32].

11.5.1 Esercizio pratico: Costruire un sistema di previsione dei difetti

Gli studenti utilizzeranno un set di dati adatto (ad esempio, includendo le misure del codice sorgente e i dati sui difetti corrispondenti) per costruire un semplice modello di previsione dei difetti e utilizzarlo per prevedere la probabilità di difetti utilizzando misure del codice sorgente da codice simile.

Il modello dovrebbe utilizzare almeno quattro feature del set di dati, e la classe dovrebbe esplorare i risultati utilizzando diverse caratteristiche per evidenziare come i risultati cambiano in base alle caratteristiche selezionate.

11.6 Utilizzare l'AI per il testing delle interfacce utente

11.6.1 Utilizzare l'AI per testare tramite la Graphical User Interface (GUI)

Il testing tramite la GUI è l'approccio tipico per il testing manuale (a parte il testing di componente) ed è spesso il punto di partenza per iniziative di automazione dei test. I test risultanti emulano l'interazione umana con l'oggetto di test. Questa automazione dei test tramite script può essere implementata applicando un approccio di cattura/riesecuzione, utilizzando le coordinate effettive degli elementi dell'interfaccia utente o gli oggetti/widget definiti dal software dell'interfaccia. Tuttavia, questo approccio presenta diversi inconvenienti con l'identificazione degli oggetti, inclusa la sensibilità alle modifiche dell'interfaccia, alle modifiche al codice e alle modifiche della piattaforma.

L'AI può essere utilizzata per ridurre la fragilità di questo approccio, utilizzando strumenti basati su AI per identificare gli oggetti corretti utilizzando vari criteri (ad esempio, XPath, etichetta, id, classe, coordinate X/Y) e per scegliere i criteri di identificazione storicamente più stabili. Ad esempio, l'ID di un pulsante in una particolare area dell'applicazione può cambiare ad ogni rilascio, quindi lo strumento basato su AI può assegnare un'importanza minore a questo ID nel tempo e fare maggiore

© International Software Testing Qualifications Board

affidamento su altri criteri. Questo approccio classifica gli oggetti nell'interfaccia utente come corrispondenti al test o non corrispondenti al test.

In alternativa, il testing visivo utilizza il riconoscimento delle immagini per interagire con gli oggetti della GUI attraverso la stessa interfaccia di un utente reale, e quindi non ha bisogno di accedere al codice sottostante e alle definizioni dell'interfaccia. Questo lo rende completamente non invadente e indipendente dalla tecnologia sottostante. Gli script devono funzionare solo tramite l'interfaccia utente visibile. Questo approccio consente al tester di creare script che interagiscono direttamente con le immagini, i pulsanti e i campi di testo sullo schermo allo stesso modo di un utente umano, senza essere influenzato dal layout generale dello schermo. L'uso del riconoscimento delle immagini nell'automazione dei test può essere limitato dalle risorse di calcolo necessarie. Tuttavia, la disponibilità di AI a prezzi accessibili che supporta un sofisticato riconoscimento delle immagini ora rende possibile questo approccio per l'uso tradizionale.

11.6.2 Utilizzare l'AI per testare la GUI

I modelli di ML possono essere utilizzati per determinare l'accettabilità delle schermate dell'interfaccia utente (ad esempio, utilizzando l'euristica e l'apprendimento supervisionato). Gli strumenti basati su questi modelli possono identificare elementi renderizzati in modo errato, determinare se alcuni oggetti sono inaccessibili o difficili da rilevare, e individuare vari altri problemi con l’aspetto visivo della GUI.

Sebbene il riconoscimento delle immagini sia una forma di algoritmo di visione artificiale, altre forme di visione artificiale basate su AI possono essere utilizzate per confrontare le immagini (ad esempio, gli screenshot) per identificare modifiche indesiderate al layout, alle dimensioni, alla posizione, al colore, al carattere o ad altri attributi visibili degli oggetti. I risultati di questi confronti possono essere utilizzati per supportare il testing di regressione, ovvero verificare che le modifiche all'oggetto di test non abbiano influito negativamente sull'interfaccia utente.

La tecnologia per verificare l'accettabilità delle schermate può essere combinata con strumenti di confronto per creare strumenti di testing di regressione basati su AI più sofisticati in grado di consigliare se le modifiche all'interfaccia utente rilevate sono probabilmente accettabili per gli utenti o se queste modifiche dovrebbero essere marcate al fine di un controllo da parte di un umano. Tali strumenti basati su AI possono essere utilizzati anche per supportare il testing di compatibilità su diversi browser, dispositivi o piattaforme finalizzato a verificare che l'interfaccia utente per la stessa applicazione funzioni correttamente su vari browser/dispositivi/piattaforme.

© International Software Testing Qualifications Board

12 Riferimenti

12.1 Standard

[S01] ISO/IEC TR 29119-11:2020, Software and systems engineering - Software testing - Part 11 Guidelines on the testing of AI-based systems

[S02] DIN SPEC 92001-1, Artificial Intelligence – Life Cycle Processes and Quality Requirements - Part 1: Quality Meta Model, https://www.din.de/en/wdc-beuth:din21:303650673 (accesso a maggio 2021).

[S03] DIN SPEC 92001-2, Artificial Intelligence – Life Cycle Processes and Quality Requirements - Part 2: Technical and Organizational Requirements, https://www.din.de/en/innovation-and-research/din-spec-en/projects/wdc-proj:din21:298702628 (accesso a maggio 2021).

[S04] ISO 26262 - https://www.iso.org/standard/68383.html (accesso a maggio 2021)

[S05] ISO/PAS 21448:2019, Road vehicles – Safety of the intended functionality (SOTIF) - https://www.iso.org/standard/70939.html (accesso a maggio 2021)

[S06] ISO/IEC 25010:2011, Systems and software engineeering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Systems and software quality models, 2011.

[S07] ISO 26262-6:2018 – Road vehicles – Functional safety - Part 6: Product development at the software level.

[S08] ISO/IEC/IEEE 29119-4:2015, Software and systems engineering - Software testing - Part 4: Test techniques.

12.2 Documenti ISTQB®

[I01] ISTQB® Certified Tester Foundation Level Syllabus, Version 2018 V3.1 https://www.istqb.org/downloads/category/2-foundation-level-documents.html (accesso a maggio 2021).

[I02] ISTQB® Certified Tester Advanced Level Test Analyst Syllabus, Version 3.1, section 3.2.6 https://www.istqb.org/downloads/category/75-advanced-level-test-analyst-v3-1.html (accesso ad agosto 2021).

[I03] ISTQB® Certified Tester AI Testing, Overview of Syllabus, Version 1.0

© International Software Testing Qualifications Board

12.3 Libri e articoli

[B01] Cadwalladr, Carole (2014). “Are the robots about to rise? Google’s new director of engineering thinks so...” The Guardian. Guardian News and Media Limited., https://www.theguardian.com/technology/2014/feb/22/robots-google-ray-kurzweil-terminator-singularity-artificial-intelligence (accesso a maggio 2021).

[B02] Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson, 2020.

[B03] M. Davies et al., “Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook”, Proceedings of the IEEE, vol. 109, no. 5, pp. 911-934, May 2021, doi: 10.1109/JPROC.2021.3067593.

[B04] Chris Wiltz, Can Apple Use Its Latest AI Chip for More Than Photos?, Electronics & Test, Artificial Intelligence, https://www.designnews.com/electronics-test/can-apple-use-its-latest-ai-chip-more-photos/153617253461497 (accesso a maggio 2021).

[B05] HUAWEI Reveals the Future of Mobile AI at IFA 2017, Huawei Press Release, https://consumer.huawei.com/en/press/news/2017/ifa2017-kirin970/ (accesso a maggio 2021).

[B06] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016, https://eur-lex.europa.eu/eli/reg/2016/679/oj (accesso a maggio 2021)

[B07] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806, SAE, https://www.sae.org/standards/content/j3016_201806/ (accesso a maggio 2021).

[B08] G20 Ministerial Statement on Trade and Digital Economy: Annex. Available from: https://www.mofa.go.jp/files/000486596.pdf (accesso a maggio 2021).

[B09] Concrete Problems in AI Safety, Dario Amodei (Google Brain), Chris Olah (Google Brain), Jacob Steinhardt (Stanford University), Paul Christiano (UC Berkeley), John Schulman (OpenAI), Dan Man´e (Google Brain), March 2016. https://arxiv.org/pdf/1606.06565 (accesso a maggio 2021).

[B10] Explainable AI: the basics, Policy briefing, Issued: November 2019 DES6051, ISBN: 978-1-78252-433-5, The Royal Society.

[B11] The Ultimate Guide to Data Labeling for Machine Learning, www.cloudfactory.com/data-labeling-guide (accesso a maggio 2021).

© International Software Testing Qualifications Board

[B12] Pei et al, DeepXplore: Automated Whitebox Testing of Deep Learning Systems, Proceedings of ACM Symposium on Operating Systems Principles (SOSP '17), Jan 2017.

[B13] Sun et al, Testing Deep Neural Networks, https://www.researchgate.net/publication/323747173_Testing_Deep_Neural_Networks, accessed Nov 2019 (accesso a maggio 2021).

[B14] A. Odena and I. Goodfellow, TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing, ArXiv e-prints, Jul. 2018, https://arxiv.org/pdf/1807.10875 (accesso a maggio 2021)

[B15] Riccio, V. et al, Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering, https://link.springer.com/article/10.1007/s10664-020-09881-0 (accesso a maggio 2021)

[B16] Baudel, Thomas et al, Addressing Cognitive Biases in Augmented Business Decision Systems., https://arxiv.org/abs/2009.08127 (accesso a maggio 2021)

[B17] Papernot, N. et al, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277, 2016. https://arxiv.org/pdf/1605.07277 (accesso a maggio 2021).

[B18] Chen et al, Metamorphic Testing: A Review of Challenges and Opportunities, ACM Comput. Surv. 51, 1, Article 4, January 2018. https://www.researchgate.net/publication/322261865_Metamorphic_Testing_A_Review_of_Challenges_and_Opportunities (accesso a maggio 2021).

[B19] Huai Liu, Fei-Ching Kuo, Dave Towey and Tsong Yueh Chen, How effectively does metamorphic testing alleviate the oracle problem?, IEEE Transactions on Software Engineering 40, 1, 4-22, 2014

[B20] James Whittaker, Exploratory Software Testing: Tips, Tricks, Tours and Techniques to Guide Test Design, 1. Edition, Addison-Wesley Professional, 2009.

[B21] L. Wilkinson, A. Anand and R. Grossman. High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. Visualization and Computer Graphics, IEEE Transactions on, 12(6):1363-1372, 2006, https://www.cs.uic.edu/~wilkinson/Publications/sorting.pdf (accesso a maggio 2021).

[B22] Ryan Hafen and Terence Critchlow, EDA and ML - A Perfect Pair for Large-Scale Data Analysis, IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6651091 (accesso a maggio 2021).

[B23] Breck, Eric, Shanqing Cai, Eric Nielsen, Michael Salib e D. Sculley, The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction, IEEE International Conference on Big Data (Big Data), 2017, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8258038 (accesso a maggio 2021).

© International Software Testing Qualifications Board

[B24] Harman, The Role of Artificial Intelligence in Software Engineering, In First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), pp. 1-6. IEEE, June 2012, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6227961 (accesso a maggio 2021).

[B25] Nilambri et al, A Survey on Automated Duplicate Detection in a Bug Repository, International Journal of Engineering Research & Technology (IJERT), 2014, https://www.ijert.org/research/a-survey-on-automated-duplicate-detection-in-a-bug-repository-IJERTV3IS041769.pdf (accesso a maggio 2021)

[B26] Kim, D.; Wang, X.; Kim, S.; Zeller, A.; Cheung, S.C.; Park, S. (2011). "Which Crashes Should I Fix First? Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts", in IEEE Transactions on Software Engineering, volume 37, https://ieeexplore.ieee.org/document/5711013 (accesso a maggio 2021).

[B27] Mao et al, Sapienz: multi-objective automated testing for Android applications, Proceedings of the 25th International Symposium on Software Testing and Analysis, July 2016, http://www0.cs.ucl.ac.uk/staff/K.Mao/archive/p_issta16_sapienz.pdf (accesso a maggio 2021).

[B28] Rai et al, Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base, World Applied Sciences Journal 31 (4): 654-662, 2014, https://www.researchgate.net/publication/336133351_Regression_Test_Case_Optimization_Using_Honey_Bee_Mating_Optimization_Algorithm_with_Fuzzy_Rule_Base (accesso a maggio 2021).

[B29] Dusica Marijan, Arnaud Gotlieb, Marius Liaaen. A learning algorithm for optimizing continuous integration development and testing practice, Journal of Software : Practice and Experience, Nov 2018.

[B30] Tosun et al, AI-Based Software Defect Predictors: Applications and Benefits in a Case Study, Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference (IAAI-10), 2010.

[B31] Kim et al, Predicting Faults from Cached History, 29th International Conference on Software Engineering (ICSE'07), 2007.

[B32] Nagappan 2008 Nagappan et al, The Influence of Organizational Structure on Software Quality: An Empirical Case Study, Proceedings of the 30th international conference on Software engineering (ICSE'08), May 2008.

[B33] Kuhn et al, Software Fault Interactions and Implications for Software Testing, IEEE Transactions on Software Engineering vol. 30, no. 6, (June 2004) pp. 418-421.

© International Software Testing Qualifications Board

12.4 Altri riferimenti

I seguenti riferimenti rimandano a informazioni disponibili su Internet. Anche se questi riferimenti sono stati verificati al momento della pubblicazione, l'ISTQB® non può essere ritenuto responsabile se i questi non sono più disponibili.

[R01] Wikipedia contributors, "AI effect", Wikipedia, https://en.wikipedia.org/wiki/AI_effect (accesso a maggio 2021).

[R02] https://mxnet.apache.org/ (accesso a maggio 2021).

[R03] https://docs.microsoft.com/en-us/cognitive-toolkit/ (accesso a maggio 2021).

[R04] IBM Watson, https://www.ibm.com/watson/ai-services

[R05] https://www.tensorflow.org/ (accesso a maggio 2021).

[R06] https://keras.io/ (accesso a maggio 2021).

[R07] https://pytorch.org/ (accesso a maggio 2021).

[R08] https://scikit-learn.org/stable/whats_new/v0.23.html (accesso a maggio 2021).

[R09] NVIDIA VOLTA, https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/ (accesso a maggio 2021).

[R10] Cloud TPU, https://cloud.google.com/tpu/ (accesso a maggio 2021).

[R11] Edge TPU, https://cloud.google.com/edge-tpu/ (accesso a maggio 2021).

[R12] Intel® Nervana™ Neural Network processors deliver the scale and efficiency demanded by deep learning model evolution, https://www.intel.ai/nervana-nnp/ (accesso a maggio 2021).

[R13] The Evolution of EyeQ, https://www.mobileye.com/our-technology/evolution-eyeq-chip/ (accesso a maggio 2021).

[R14] ImageNet - http://www.image-net.org/ (accesso a maggio 2021).

[R15] Google's BERT - https://github.com/google-research/bert (accesso a maggio 2021).

[R16] https://www.kaggle.com/datasets (accesso a maggio 2021).

[R17] https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-data-science-survey (accesso a maggio 2021).

[R18] MLCommons - https://mlcommons.org/ (accesso a maggio 2021).

[R19] DAWNBench - https://dawn.cs.stanford.edu/benchmark (accesso a maggio 2021).

[R20] MLMark - https://www. eembc.org/mlmark (accesso a maggio 2021).

© International Software Testing Qualifications Board

[R21] https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment| Shaping Europe's digital future (europa.eu) (accesso ad agosto 2021)

[R22] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (accesso ad agosto 2021)

[R23] Google GraphicsFuzz, https://github.com/google/graphicsfuzz (accesso a maggio 2021).

[R24] http://www.openrobots.org/morse/doc/0.2.1/morse.html (accesso a maggio 2021).

[R25] https://ai.facebook.com/blog/open-sourcing-ai-habitat-a-simulation-platform-for-embodied-ai-research/ (accesso a maggio 2021).

[R26] https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/ (accesso a maggio 2021).

[R27] https://uk.mathworks.com/discovery/artificial-intelligence.html#ai-with-matlab (accesso a maggio 2021).

© International Software Testing Qualifications Board

13 Appendice A - Abbreviazioni

Abbreviazione	Descrizione
AI	Artificial Intelligence (intelligenza artificiale)
AIaaS	AI as a service
API	Application programming interface
AUC	Area under curve (area sotto il grafico)
DL	Deep learning
DNN	Deep neural network (rete neurale profonda)
EDA	Exploratory data analysis
EU	European Union (Unione Europea)
FN	False negative (falso negativo)
FP	False positive (falso positivo)
GDPR	General data protection regulation
GPU	Graphical processing unit (unità di elaborazione grafica)
GUI	Graphical user interface
LIME	Local interpretable model-agonistic explanations
MC/DC	Modified condition / decision coverage (Copertura delle decisioni / condizioni modificate)
ML	Machine learning
MR	Metamorphic relation (relazione metamorfica)
MSE	Mean square error (errore quadratico medio)
MT	Metamorphic testing (testing metamorfico)
NLP	Natural language processing (elaborazione del linguaggio naturale)
ROC	Receiver operating characteristic
SUT	System under test
SVM	Support vector machine (macchina a vettore di supporto)
TN	True negative (vero negativo)
TP	True positive (vero positivo)
XAI	Explainable AI

© International Software Testing Qualifications Board

14 Appendice B - Termini specifici di AI e altri termini

Nome del termine	Definizione
Accuratezza	La metrica delle prestazioni funzionali del ML utilizzata per valutare un classificatore, che misura la proporzione di previsioni corrette (Dopo ISO/IEC TR 29119-11)
Acquisizione dei dati	L'attività di acquisizione di dati rilevanti per il problema di business da risolvere con un modello di ML
Addestramento del modello di ML	Il processo di applicazione dell'algoritmo di ML al set di dati di addestramento per creare un modello di ML
Adversarial attack	L'uso deliberato di adversarial examples per causare il fallimento di un modello di ML
Agente intelligente	Un programma autonomo che orienta la propria attività verso il raggiungimento degli obiettivi attraverso osservazioni e azioni
AI as a Service (AIaaS)	Un modello di licenza e distribuzione del software in cui l'AI e i servizi di sviluppo dell'AI sono ospitati centralmente
AI generale	AI che mostra un comportamento intelligente paragonabile a quello di un umano nell'intera gamma delle capacità cognitive (ISO/IEC TR 29119-11) Sinonimo: AI forte
AI ristretta	AI concentrata su un unico compito ben definito per affrontare un problema specifico (ISO/IEC TR 29119-11) Sinonimo: AI debole
Albero delle decisioni	Un modello di ML ad albero i cui nodi rappresentano le decisioni e i cui rami rappresentano i possibili risultati
Algoritmo di apprendimento	Un programma che produce un modello di ML basato sulle proprietà del set di dati di addestramento
Algoritmo di clustering	Un tipo di algoritmo di ML utilizzato per raggruppare oggetti simili in cluster
Algoritmo di ML	Un algoritmo utilizzato per creare un modello di ML da un set di dati di addestramento

© International Software Testing Qualifications Board

Nome del termine	Definizione
Algoritmo di ricerca	Un algoritmo che visita sistematicamente un sottoinsieme di tutti i possibili stati o strutture fino al raggiungimento dello stato o della struttura obiettivo (Dopo ISO/IEC TR 29119-11)
Annotazione	L'attività di identificazione degli oggetti nelle immagini con riquadri di delimitazione per fornire dati etichettati per la classificazione
Apprendimento non supervisionato	Addestrare un modello di ML dai dati di input utilizzando un set di dati non etichettati
Apprendimento per rinforzo	L'attività di costruzione di un modello di ML utilizzando un processo di trial e reward per raggiungere un obiettivo (Dopo ISO/IEC TR 29119-11)
Apprendimento supervisionato	Addestrare un modello di ML dai dati di input e dalle relative etichette
Architettura di von Neumann	Un'architettura di computer composta da cinque componenti principali: memoria, un'unità di elaborazione centrale, un'unità di controllo, input e output
Area sotto il grafico (AUC) (Area Under Curve)	Una misura di quanto bene un classificatore può distinguere tra due classi
Artificial Intelligence (AI) (intelligenza artificiale)	La capacità di un sistema ingegnerizzato di acquisire, elaborare, creare e applicare conoscenze e abilità (ISO/IEC TR 29119-11)
Associazione	Una tecnica di apprendimento non supervisionato che identifica le relazioni e le dipendenze tra i campioni
Aumento	L'attività di creare nuovi punti di dati basati su un set di dati esistente
Automation bias (condizionamento dell'automazione)	Un tipo di bias causato da una persona che preferisce le raccomandazioni di un sistema decisionale automatizzato rispetto ad altre fonti Sinonimo: complacency bias
Autonomia	La capacità di un sistema di funzionare per periodi prolungati senza intervento umano (ISO/IEC TR 29119-11)
Avvelenamento dei dati	La manipolazione deliberata e dannosa dei dati di addestramento o di input in un modello di ML
Bias	La differenza sistematica nel trattamento di determinati oggetti, persone o gruppi rispetto ad altri (ISO/IEC DIS 22989)

© International Software Testing Qualifications Board

Nome del termine	Definizione
Bias algoritmico	Un tipo di bias causato dall'algoritmo di ML
Bias di campionamento	Un tipo di bias in cui il set di dati non è completamente rappresentativo dello spazio di dati a cui viene applicato il ML
Bias inappropriato	Un tipo di bias che fa sì che un sistema produca risultati che portano a effetti negativi per un particolare gruppo
Big data	Set di dati estesi le cui caratteristiche in termini di volume, varietà, velocità e/o variabilità richiedono tecnologie e tecniche specializzate per l'elaborazione
Caratteristica	Un singolo attributo misurabile dei dati di input utilizzati per l'addestramento da un algoritmo di ML e per la previsione da un modello di ML
Chatbot	Un'applicazione utilizzata per condurre una conversazione tramite testo o sintesi vocale
Classificatore	Un modello di ML utilizzato per la classificazione Sinonimo: modello di classificazione
Classificatore deduttivo	Un classificatore basato sull'applicazione di inferenza e logica ai dati di input
Classificazione	Un tipo di funzione di ML che prevede la classe di output per un dato input (Dopo ISO/IEC TR 29119-11)
Clustering	Un tipo di funzione di ML che raggruppa punti di dati simili
Coefficiente di silhouette	Una misura di clustering tra -1 e +1 basata sulla media delle differenze tra inter-cluster e intra-cluster Sinonimo: punteggio di silhouette
Componente AI	Un componente che fornisce funzionalità AI
Curva Receiver Operating Characteristic (ROC)	Un grafico che illustra la capacità di un classificatore binario al variare della sua soglia di discriminazione
Deep learning (DL)	ML che utilizza reti neurali con strati multipli

© International Software Testing Qualifications Board

Nome del termine	Definizione
Deriva concettuale	Un cambiamento nell'accuratezza percepita delle previsioni di un modello di ML nel tempo, causato da cambiamenti nelle aspettative degli utenti, nel comportamento e nell'ambiente operativo
Edge computing	La parte di un'architettura distribuita in cui l'elaborazione delle informazioni viene eseguita vicino a dove vengono utilizzate tali informazioni
Effetto AI	La situazione in cui un sistema di AI precedentemente etichettato non è più considerato AI con l'avanzare della tecnologia (ISO/IEC TR 29119-11)
Elaborazione del linguaggio naturale (Natural Language Processing) (NLP)	Un campo dell'informatica che fornisce la capacità di leggere, comprendere e ricavare significati dai linguaggi naturali
Epoch	Un'iterazione di addestramento del ML sull'intero set di dati di addestramento
Errore quadratico medio (Mean Square Error) (MSE)	La misura statistica della differenza quadratica media tra i valori stimati e il valore effettivo
Etichettatura dei dati	L'attività di aggiungere etichette significative agli oggetti nei dati grezzi per supportare la classificazione nel ML
Evoluzione	Il processo di cambiamento continuo da uno stato inferiore, più semplice o peggiore a uno stato superiore, più complesso o migliore
Explainable AI (XAI)	Il campo di studio relativo alla comprensione dei fattori che influenzano gli output dei sistemi di AI
Exploratory Data Analysis (EDA)	L'esplorazione interattiva, guidata da ipotesi e visiva dei dati utilizzata per supportare l'ingegneria delle caratteristiche
Falso negativo (FN)	Una previsione del modello di ML in cui il modello prevede erroneamente la classe negativa
Falso positivo (FP)	Una previsione del modello di ML in cui il modello prevede erroneamente la classe positiva
Flessibilità	La capacità di un sistema di funzionare in contesti al di fuori della sua specifica iniziale (Dopo ISO/IEC TR 29119-11)

© International Software Testing Qualifications Board

Nome del termine	Definizione
Flusso di lavoro di ML	Una sequenza di attività utilizzate per gestire lo sviluppo e la distribuzione di un modello di ML
Foresta casuale	La tecnologia di ML Ensemble per la classificazione, la regressione e altre attività che operano costruendo ed eseguendo molti alberi decisionali e poi producendo la moda della classe o la previsione media dei singoli alberi
Framework di ML	Uno strumento o una libreria che supporta la creazione di un modello di ML
Funzione di attivazione	La formula associata a un neurone in una rete neurale che determina l'output del neurone dagli input al neurone
Funzione di ML	Funzionalità implementate da un modello di ML, come classificazione, regressione o clustering
Funzione di ricompensa	Una funzione che definisce il successo dell'apprendimento per rinforzo
General Data Protection Regulation (GDPR)	Il regolamento dell'Unione europea (UE) sulla protezione dei dati e la privacy che si applica ai dati dei cittadini dell'UE e dell'Area Economica Europea
Ingegneria delle caratteristiche	L'attività in cui vengono identificati quegli attributi nei dati grezzi che rappresentano al meglio le relazioni sottostanti che dovrebbero apparire nel modello di ML per l'uso nei dati di addestramento (ISO/IEC TR 29119-11)
Interpretabilità	Il livello di comprensione del funzionamento della tecnologia di AI sottostante (ISO/IEC TR 29119-11)
Iperparametro	Un parametro utilizzato per controllare l'addestramento di un modello di ML o per impostare la configurazione di un modello di ML
K-nearest neighbor	Un approccio alla classificazione che stima la probabilità di appartenenza a un gruppo per un punto di dati in base all'appartenenza al gruppo dei punti di dati più vicini ad esso
Logica fuzzy	Un tipo di logica basata sul concetto di verità parziale rappresentato da fattori di certezza compresi tra 0 e 1
Macchina a vettore di supporto (SVM)	Una tecnica di ML in cui i punti di dati sono visti come vettori nello spazio multidimensionale separati da un iperpiano

© International Software Testing Qualifications Board

Nome del termine	Definizione
Machine Learning (ML)	Il processo che utilizza tecniche computazionali per consentire ai sistemi di apprendere dai dati o dall'esperienza (ISO/IEC TR 29119-11)
Matrice di confusione	Una tecnica per riassumere le prestazioni funzionali del ML di un algoritmo di classificazione
Metodo LIME	Il programma Local Interpretable Model-Agnostic Explanations per spiegare le previsioni da un modello di ML
Metrica inter-cluster	Una metrica che misura la somiglianza dei punti di dati in diversi cluster
Metrica intra-cluster	Una metrica che misura la somiglianza dei punti di dati all'interno di un cluster
Modello Bayesiano	Un modello statistico che utilizza la probabilità per rappresentare l'incertezza sia degli input che degli output del modello
Modello di regressione	Un modello di ML il cui output previsto per un dato input numerico è una variabile continua (Dopo ISO/IEC DIS 23053)
Modello pre-addestrato	Un modello di ML già addestrato quando è stato ottenuto
Motore di regole	Un insieme di regole che determinano quali azioni dovrebbero verificarsi quando vengono soddisfatte determinate condizioni
Neurone	Un nodo in una rete neurale che di solito riceve più valori di input e genera un valore di attivazione
Overfitting	La generazione di un modello di ML che corrisponde troppo strettamente al set di dati di addestramento, risultando in un modello che ha difficoltà a generalizzarsi a nuovi dati (dopo ISO/IEC TR 29119-11)
Percettrone	Una rete neurale con un solo strato e un solo neurone
Peso	Una variabile interna di una connessione tra neuroni in una rete neurale che influenza il modo in cui calcola i suoi output e che cambia quando la rete neurale viene addestrata
Pipeline di dati	L'implementazione di attività di preparazione dei dati per fornire dati di input per supportare l'addestramento mediante un algoritmo di ML o la previsione mediante un modello di ML

© International Software Testing Qualifications Board

Nome del termine	Definizione
Precisione	Una metrica delle prestazioni funzionali del ML utilizzata per valutare un classificatore, che misura la proporzione di positivi previsti che erano corretti (Dopo ISO/IEC TR 29119-11)
Pre-elaborazione dei dati	Le attività di pulizia dei dati, trasformazione dei dati, aumento dei dati e campionamento dei dati nel flusso di lavoro di ML
Preparazione dei dati	Le attività di acquisizione dei dati, pre-elaborazione dei dati e ingegneria delle caratteristiche nel flusso di lavoro di ML
Previsione dei difetti	Una tecnica per prevedere le aree all'interno dell'oggetto di test in cui si verificheranno i difetti o la quantità di difetti presenti
Problema dell'oracolo di test	La sfida di determinare se un test è stato superato o meno per un determinato insieme di input e stato di test
Processore neuromorfico	Un circuito integrato progettato per imitare i neuroni biologici del cervello umano
Processore specifico per l'AI	Un tipo di hardware specializzato progettato per accelerare le applicazioni di AI
Punteggio F1	Una metrica delle prestazioni funzionali del ML utilizzata per valutare un classificatore che fornisce un equilibrio tra richiamo e precisione
Punto di dati	Un insieme di una o più misurazioni che comprende una singola osservazione utilizzata come parte di un set di dati
R al quadrato	Una misura statistica della vicinanza dei punti di dati alla retta di regressione adattata. Sinonimo: coefficiente di determinazione
Ragionamento basato sul caso	La tecnica di risolvere un nuovo problema basandosi sulle soluzioni di problemi simili avvenuti in passato
Ragionamento procedurale	Tecnologia di AI utilizzata per costruire sistemi di ragionamento in tempo reale in grado di eseguire compiti complessi in ambienti dinamici
Regressione	Un tipo di funzione di ML che risulta in un valore di output numerico o continuo per un dato input (Dopo ISO/IEC TR 29119-11)

© International Software Testing Qualifications Board

Nome del termine	Definizione
Regressione lineare	Una tecnica statistica che modella la relazione tra le variabili adattando un'equazione lineare ai dati osservati quando la variabile target è numerica
Regressione logistica	Una tecnica statistica che modella la relazione tra le variabili quando la variabile target è categorica anziché numerica
Rete neurale	Una rete di elementi di elaborazione primitivi collegati da collegamenti ponderati con pesi regolabili, in cui ogni elemento produce un valore applicando una funzione non lineare ai suoi valori di input, e lo trasmette ad altri elementi o lo presenta come valore di output (ISO/IEC 2382) Sinonimo: rete neurale artificiale
Rete neurale profonda	Una rete neurale composta da diversi strati di neuroni Sinonimo: percettrone multistrato
Rete neurale trojan	Una vulnerabilità iniettata in una rete neurale tramite un attacco di avvelenamento dei dati con l'intento di sfruttarla in seguito
Reward hacking	L'attività svolta da un agente intelligente per massimizzare la sua funzione di ricompensa a scapito del raggiungimento dell'obiettivo originario (Dopo ISO/IEC TR 29119-11)
Richiamo	Una metrica delle prestazioni funzionali del ML utilizzata per valutare un classificatore, che misura la proporzione di positivi effettivi previsti correttamente (Dopo ISO/IEC TR 29119-11) Sinonimo: sensibilità
Rumore	Una distorsione o una corruzione dei dati
Safety	L'aspettativa che un sistema non porti, in condizioni definite, a uno stato in cui la vita umana, la salute, la proprietà o l'ambiente siano in pericolo (ISO/IEC/IEEE 12207)
Set di dati	Una raccolta di dati utilizzati per l'addestramento, la valutazione, il testing e la previsione nel ML
Set di dati di addestramento	Un set di dati utilizzato per addestrare un modello di ML

© International Software Testing Qualifications Board

Nome del termine	Definizione
Set di dati di validazione	Un set di dati utilizzato per valutare un modello di ML addestrato allo scopo del tuning del modello
Singolarità tecnologica	Un punto nel futuro in cui i progressi tecnologici non saranno più controllabili dalle persone (Dopo ISO/IEC TR 29119-11)
Sistema autonomo	Un sistema in grado di funzionare senza l'intervento umano per periodi prolungati
Sistema basato su AI	Un sistema che integra uno o più componenti AI
Sistema deterministico	Un sistema che produrrà lo stesso insieme di output e lo stesso stato finale da un dato insieme di input e stato iniziale
Sistema di autoapprendimento	Un sistema adattivo che cambia il suo comportamento basandosi sull'apprendimento attraverso trial and error (Dopo ISO/IEC TR 29119-11)
Sistema di ML	Un sistema che integra uno o più modelli di ML
Sistema esperto	Un sistema basato su AI per risolvere problemi in un particolare dominio o area di applicazione traendo inferenze da una base di conoscenza sviluppata dall'esperienza umana
Sistema multi-agente	Un sistema che comprende più agenti intelligenti
Sistema non deterministico	Un sistema che non produrrà sempre lo stesso insieme di output e lo stesso stato finale dato un particolare insieme di input e stato iniziale
Sistema probabilistico	Un sistema il cui comportamento è descritto in termini di probabilità; quindi i suoi output non possono essere previsti perfettamente
Soglia di decisione	Un valore che trasforma il risultato di una funzione di previsione in un risultato binario superiore o inferiore al valore Sinonimo: soglia di discriminazione
Spiegabilità	Il livello di comprensione di come il sistema basato su AI ha ottenuto un determinato risultato (ISO/IEC TR 29119-11)
Suite di benchmark per il ML	Un set di dati utilizzato per confrontare i modelli di ML e gli algoritmi di ML su una serie di metriche di valutazione
Super AI	Un sistema basato sull'intelligenza artificiale che supera di gran lunga le capacità umane

© International Software Testing Qualifications Board

Nome del termine	Definizione
Tecnica Bayesiana	Una tecnica che considera le distribuzioni di probabilità prima e dopo come parametri di un modello statistico
Tecnica di ragionamento	AI che genera conclusioni dalle informazioni disponibili utilizzando tecniche logiche (Dopo ISO/IEC TR 29119-11)
Transfer learning	Una tecnica per modificare un modello di ML pre-addestrato per eseguire un'attività correlata diversa
Trasparenza	Il livello di visibilità dell'algoritmo e dei dati utilizzati dal sistema basato su AI (Dopo ISO/IEC TR 29119-11)
Tuning del modello di ML	Il processo di testing degli iperparametri per ottenere prestazioni ottimali
Tuning dell'iperparametro	L'attività di determinazione degli iperparametri ottimali in base a particolari obiettivi
Underfitting	La generazione di un modello di ML che non riflette la tendenza alla base del set di dati di addestramento, risultando in un modello che ha difficoltà a fare previsioni accurate (ISO/IEC TR 29119-11)
Unità di elaborazione grafica (Graphical Processing Unit) (GPU)	Un circuito integrato specifico dell'applicazione progettato per manipolare e alterare la memoria per accelerare la creazione di immagini in un frame buffer destinato all'output su un dispositivo di visualizzazione
Valore anomalo	Un'osservazione che si trova al di fuori del modello generale della distribuzione dei dati
Valore di attivazione	L'output di una funzione di attivazione di un neurone in una rete neurale
Valutazione del modello di ML	Il processo di confronto delle metriche delle prestazioni funzionali del ML raggiunte con i criteri richiesti e quelle di altri modelli di ML
Verità di base	Le informazioni fornite dall'osservazione diretta e dalla misurazione che sono note per essere reali o vere
Vero negativo (TN)	Una previsione in cui il modello prevede correttamente la classe negativa
Vero positivo (TP)	Una previsione in cui il modello prevede correttamente la classe positiva

© International Software Testing Qualifications Board

Nome del termine	Definizione
Visualizzazione dei dati	Una tecnica per rappresentare graficamente relazioni, tendenze e modelli di dati

© International Software Testing Qualifications Board

15 Indice

A/B testing; 72; 73; 76
acceptance testing; 56
adaptability; 21; 22; 23; 54; 55; 58
adversarial attack; 19; 36; 70
adversarial example; 36; 70; 76
API testing; 56
attacker; 19; 40; 70
availability; 17; 57; 80
back-to-back testing; 67; 72; 73; 76; 83
bias; 23; 25; 39; 50; 63
black-box testing; 65
checklist-based testing; 74
classification; 28; 32; 44; 45
clustering; 28; 32; 45; 47
combinatorial testing; 71
compatibility; 67
complexity; 25; 65
component; 55; 56; 59
component integration testing; 55
component testing; 55
confirmation testing; 64
continuous testing; 62
coverage; 55; 77; 82; 83
data poisoning; 70; 76
data privacy; 40
debugging; 83
decision tree; 15; 30
denial-of-service; 70
dynamic testing; 55; 65
effectiveness; 19; 23; 71; 72
efficiency; 23; 77
encryption; 56
error guessing; 74
experience-based testing; 33; 74; 77
explainability; 21; 25; 26; 46; 56; 57; 65; 66; 67; 68; 79
exploratory testing; 65; 74; 75
flexibility; 22
follow-up test case; 73; 74
functional correctness; 58
functional suitability; 67
fuzz testing; 83
graphical user interface; 81; 84; 85
input data testing; 55
installability; 73
integration testing; 55; 56; 58
interpretability; 25; 65
machine learning; 46; 57
maintainability; 67
metamorphic relation; 73; 74
metamorphic testing; 73; 74
ML algorithms; 28
ML functional performance criteria; 55; 56; 58; 59; 60
ML functional performance metrics; 17; 29; 30; 43; 45; 46; 57; 68; 72
ML model; 16; 17; 18; 29; 33; 36; 38; 40; 46
ML model testing; 28; 55; 56
ML workflow; 32; 36; 38
neural network; 49; 50; 51; 70
neuron coverage; 51
non-functional requirement; 56; 58; 67; 72
non-functional test; 30; 73
operational environment; 23; 56; 58; 62; 63; 67; 79
pairwise testing; 71; 76

© International Software Testing Qualifications Board

performance efficiency; 57; 67
poisoning attack; 70
portability; 57; 67
pseudo-oracle; 72; 76; 83
quality characteristics; 21; 33; 46
regression; 45; 47
regression testing; 64; 83
reinforcement learning; 28; 29; 32
reliability; 67
reward hacking; 23; 24
scalable; 16; 37; 80
security; 17; 25; 39; 40; 57; 67
security vulnerabilities; 40
sign-change coverage; 51
sign-sign coverage; 51
simulator; 56; 80
source test case; 73; 74
supervised learning; 28; 32; 41; 46; 49
system testing; 56; 71
system under test; 56; 72; 79
test automation; 84; 85
test data; 56
test dataset; 30; 33; 38; 57
test design; 62; 74
test environment; 56; 63; 71; 78; 79
test level; 54; 56
test model; 83
test oracle; 21; 54; 66; 67; 72; 73; 74; 83
test planning; 79
test run; 72
test suite; 71; 83; 84
test technique; 67; 71; 74; 76; 86
testability; 65
threshold coverage; 51
tour; 74; 76
training dataset; 30; 33; 38
transparency; 25; 65
unsupervised learning; 28; 32
usability; 67
user experience; 82
user interface; 54; 55; 82; 83; 84; 85
validation; 38; 57; 75; 77
validation dataset; 30; 38
value-change coverage; 51
virtual test environment; 79; 80
visual testing; 85
white-box testing; 58; 65

© International Software Testing Qualifications Board