

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



**SalesCRM: integrazione di un workflow per
l'automazione delle vendite online**

Tesi di laurea

Relatore

Prof. Marco Zanella

Laureando

Valerio Occhinegro

Matricola 2011069

Valerio Occhinegro: *SalesCRM: integrazione di un workflow per l'automazione delle vendite online*, Tesi di laurea, © Dicembre 2024.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal laureando Valerio Occhinegro presso l'azienda Spazio Dev Srl di Tombolo (PD). Lo stage, svoltosi al termine del percorso di studi della Laurea Triennale in "Scienze Informatiche", ha avuto una durata complessiva di 306 ore.

Il lavoro in questione è stato suddiviso in 4 diverse fasi, ognuna delle quali caratterizzata da obiettivi specifici e scadenze precise, e si basa su un'analisi intelligente di siti web, improntata alla vendita dei servizi aziendali a potenziali clienti. Il ruolo del laureando è quello di gestire e archiviare i siti web target e successivamente classificarli per fornire a Spazio Dev input utili per acquisire l'abilità di ampliare la quantità dei propri clienti in maniera rapida e automatizzata. Nello specifico, il progetto di stage ha l'obiettivo di sviluppare competenze avanzate nell'intelligenza artificiale, con un focus particolare sulla classificazione e organizzazione dei dati; uno studio che si inserisce nel contesto di "SalesCRM: integrazione di un workflow per l'automazione delle vendite online". Lo scopo ultimo del progetto è la creazione di un sistema intelligente che sia in grado di analizzare *dataset* complessi, classificare la qualità dei siti web (siti che potrebbero essere migliorati e siti che non necessitano di modifiche) e proporre i propri servizi alle aziende che ne necessitano in maniera automatica. Il laureando è responsabile dello sviluppo di funzionalità chiave del sistema che consentiranno di fornire soluzioni strategiche agli utilizzatori della piattaforma.

“Non importa quanto piano vai, finché non ti fermi”

— Confucio

Ringraziamenti

Prima di tutto vorrei ringraziare profondamente il Prof. Marco Zanella, per avermi aiutato durante la stesura di questa tesi; senza i suoi consigli e la sua disponibilità fuori dal comune avrei sicuramente scritto un elaborato di qualità inferiore. Ha svolto un lavoro talmente impeccabile da meritarsi tutta la mia stima e per quel poco che conta un po’di pubblicità positiva con i miei colleghi che devono ancora laurearsi.

Ringrazio tutta la mia famiglia, i miei zii e i miei cugini per essermi stati vicini e avermi sostenuto in questi anni difficili. Un pensiero speciale va a mia sorella Claudia, per gli ottimi consigli dispensati.

Dedico degli ulteriori ringraziamenti ai miei amici Irene, Paño, Alessia e Mattia che mi hanno aiutato a tenere alto il morale durante questo percorso; a tutti i colleghi universitari il cui supporto è stato di fondamentale importanza per superare gli ultimi esami e all’intero Dipartimento di Chimica per avermi ospitato innumerevoli volte.

Un ringraziamento particolare è riservato a Matteo che posso considerare un vero e proprio fratello maggiore (senza nulla da togliere a Claudia), lo ringrazio per avermi aiutato durante uno degli anni più faticosi e per avermi fatto appassionare al mondo della "Geomatica".

Padova, Dicembre 2024

Valerio Occhinegro

Indice

1	Introduzione	1
1.1	Organizzazione del testo	1
1.2	L'azienda	2
1.3	Servizi offerti	2
1.4	Clientela	3
1.5	Prodotto di punta	3
2	Processi e metodologie	4
2.1	Ciclo di vita del software	4
2.1.1	Filosofia Agile	4
2.1.2	Tecnologie di supporto	5
2.2	Gestione della configurazione	6
2.2.1	Tecnologie di supporto	6
3	Il progetto	8
3.1	Analisi del progetto	8
3.1.1	SalesCRM	8
3.1.2	Integrazione	10
3.2	Analisi e gestione dei rischi	10
3.3	Obiettivi	11
4	Analisi dei requisiti	12
4.1	Casi d'uso	12
5	Strumenti e tecnologie	16
5.1	Strumenti	16
5.1.1	Visual Studio Code	16
5.1.2	PhpMyAdmin	17
5.1.3	remoteripple	17
5.2	Tecnologie	18
5.2.1	Python	18
5.2.2	CUDA	19
5.2.3	Docker	20
5.2.4	DDEV	20
5.2.5	WSL	21
5.2.6	Laravel	21
5.2.7	Filament	22
5.2.8	MySQL	23

6 Machine learning	24
6.1 K-means	26
6.2 PCA	27
6.3 Reti neurali	28
6.3.1 Apprendimento	29
6.3.2 Classificazione	30
6.4 Deep learning	31
6.4.1 Convolutional neural network	31
6.5 Autoencoder	35
6.5.1 Funzionamento	35
7 Progettazione	36
7.1 Architettura	36
7.2 Cattura immagini	37
7.3 Studio preliminare per l'estrazione delle feature	38
7.3.1 Tentativi con autoencoder	38
7.3.2 Modelli pre-addestrati	42
7.4 Clustering	42
7.4.1 Preparazione delle immagini	42
7.4.2 Estrazione delle feature	44
7.4.3 Applicazione del clustering	45
7.5 Valutazione	46
7.5.1 Preparazione del dataset	46
7.5.2 Addestramento del modello	46
7.5.3 Fine-tuning del modello	46
7.5.4 Valutazione delle immagini	47
7.6 Invio e-mail	47
7.7 Database	47
7.7.1 Domains	47
7.7.2 Pages	48
8 Conclusioni	49
8.1 Raggiungimento degli obiettivi	49
8.1.1 Clustering	51
8.1.2 Valutazione delle immagini	52
8.2 Valutazione personale	53
Glossario	54
Bibliografia	56

Elenco delle figure

1.1	Logo dell'azienda	2
1.2	Logo dell'applicazione RelAI	3
2.1	Schema del modello <i>scrum</i>	5
2.2	Logo di Plane	5
2.3	Logo di Telegram	5
2.4	Logo di Git	6
2.5	Logo di Gitea	6
2.6	Esempio di Git Flow	7
3.1	Homepage di SalesCRM	9
3.2	Calendario contenuto all'interno di SalesCRM	9
3.3	Mappa contenuta all'interno di SalesCRM	9
4.1	Use Case - UC0: Scenario principale	12
4.2	Use Case - UC1: Acquisizione screenshot	13
4.3	Use Case - UC2: Clusterizzazione degli screenshot	14
4.4	Use Case - UC3: IA classificativa	14
4.5	Use Case - UC4: invio e-mail	15
5.1	Logo di Visual Studio Code	16
5.2	Logo di phpMyAdmin	17
5.3	Logo di Remote Ripple	17
5.4	Python e librerie utilizzate	19
5.5	Logo di CUDA	20
5.6	Logo di Docker	20
5.7	Logo di DDEV	21
5.8	Logo di WSL	21
5.9	Logo di Laravel	22
5.10	Esempio di interfaccia grafica creata in Filament	22
5.11	Logo di Filament	22
5.12	Logo di MySQL	23
6.1	Tre tipologie di pianta	25
6.2	Schema delle feature	25
6.3	Esempio di clustering	26
6.4	Grafico rappresentante il metodo del gomito	27
6.5	Grafico contenente PCA	28
6.6	Esempio di rete neurale	28

6.7	Funzionamento del neurone	29
6.8	Funzioni di attivazione	30
6.9	Forward e backward propagation	30
6.10	Classificazione dei dati	31
6.11	Esempio di rete neurale con più strati nascosti	31
6.12	Esempio di rete neurale convoluzionale	32
6.13	Funzionamento del kernel	32
6.14	Funzionamento del pooling	33
6.15	Schema generale di un autoencoder	35
7.1	Schema architetturale del progetto	37
7.2	Schema della fase di cattura	37
7.3	Autoencoder con strati densi	38
7.4	Ricostruzione immagine ottenuta dal bottleneck dell'autoencoder a strati densi con 150 epochi di addestramento	39
7.5	Autoencoder con strati convoluzionali	39
7.6	Ricostruzione immagine ottenuta dal bottleneck dell'autoencoder convoluzionale con 500 epochi di addestramento	40
7.7	Feature map estratte dal primo strato convoluzionale	40
7.8	Feature map estratte dal secondo strato convoluzionale	41
7.9	Feature map estratte dal terzo strato convoluzionale	41
7.10	Clustering effettuato utilizzando le <code>feature</code> estratte dall'autoencoder convoluzionale; 20 cluster e 3000 immagini	42
7.11	Conversione e ridimensionamento	43
7.12	Struttura del tensore	44
7.13	Esempio di feature maps	44
7.14	Struttura di ResNet50	44
7.15	Clustering effettuato utilizzando ResNet50; 20 cluster e 3000 immagini	45
7.16	Struttura del dataset	46
7.17	Schema della tabella Domains	47
7.18	Schema della tabella Pages	48
8.1	Immagini appartenenti a contesti differenti	51
8.2	Immagini appartenenti allo stesso contesto	52

Elenco delle tabelle

3.1	Tabella degli obiettivi	11
-----	-----------------------------------	----

ELENCO DELLE TABELLE

viii

8.1 Grado di soddisfacimento degli obiettivi	50
--	----

Capitolo 1

Introduzione

L'esperienza dello stage è l'elemento che porta alla conclusione del percorso accademico dello studente; pertanto è da considerare una delle prove più formanti e impegnative di tutto l'arco di studi.

Durante questo periodo lo studente ha l'occasione di mettere alla prova le conoscenze acquisite negli anni, ma soprattutto ha la possibilità di valutare le proprie abilità di autoapprendimento, scontrandosi con tecnologie e problemi mai visti in precedenza. L'introduzione al mondo del lavoro è fondamentale per rendere il più lineare possibile la transizione da studio ad applicazione pratica delle conoscenze, fornisce inoltre un assaggio delle responsabilità che gravano su un programmatore durante la permanenza nella propria azienda. In questo capitolo viene presentata l'azienda Spazio Dev presso la quale lo studente ha svolto lo stage nel periodo che va dal giorno 23/09/2024 al 14/11/2024.

1.1 Organizzazione del testo

Il secondo capitolo descrive i processi e le metodologie applicate dallo studente durante lo sviluppo del progetto.

Il terzo capitolo contiene tutte le informazioni relative al progetto da realizzare.

Il quarto capitolo sviluppa la fase di analisi dei requisiti.

Il quinto capitolo tratta gli strumenti e le tecnologie adottate dallo studente.

Il sesto capitolo fornisce una base teorica.

Il settimo capitolo approfondisce l'architettura del progetto.

L'ottavo capitolo mostra lo stato di completamento e le future migliorie implementabili nel progetto.

Per quanto riguarda la stesura del documento sono state adottate le seguenti convenzioni tipografiche:

- I termini che necessitano di una definizione in quanto facenti parte del gergo tecnico sono definiti all'interno del glossario.
- La prima occorrenza dei termini contenuti nel glossario viene evidenziata nella maniera seguente: *esempio^[g]*.
- I termini in lingua inglese vengono scritti in *corsivo*.

1.2 L'azienda

Spazio Dev¹ è una *software house* italiana nata a Tombolo (PD) nel 2023; nonostante la giovane età è riuscita ad affermarsi nel suo campo dando vita a molteplici progetti ambiziosi. L'obiettivo primario dell'azienda è quello di supportare diverse imprese tramite soluzioni digitali e tecnologiche sviluppate in base alle necessità. L'azienda conta 16 membri che ricoprono i diversi ruoli di sviluppatori, grafici, responsabili marketing e sistemisti. L'organico ha un'età media inferiore ai trenta anni, questo fattore contribuisce a rendere l'ambiente lavorativo dinamico e conviviale, perfetto per coltivare nuovi talenti. Spazio Dev affianca e aggiorna il cliente tramite una comunicazione costante, necessaria per adattare il prodotto alle esigenze richieste. Il focus principale dell'azienda è infatti quello di fornire prodotti specifici per ciascun cliente e non la vendita di prodotti standard.

Il logo dell'azienda è rappresentato in Fig. 1.1



Figura 1.1: Logo dell'azienda

1.3 Servizi offerti

L'offerta di Spazio Dev è molto ampia e consente allo staff di aiutare i clienti nei seguenti ambiti:

- **Sviluppo di siti web:** La produzione di siti web è uno dei servizi più richiesti all'azienda e ne rappresenta uno dei motori trainanti. Gli sviluppatori hanno un occhio di riguardo per le nuove tecnologie, che sfruttano per offrire al cliente un prodotto accattivante e ottimizzato in maniera tale da garantire numerose visite.
- **Sviluppo software e integrazione IA:** I software prodotti sono solitamente indirizzati a una clientela variegata e possono spaziare in vari campi, in base alle esigenze di ciascun acquirente. Gli applicativi più venduti hanno come scopo principale la gestione dell'azienda e delle relazioni con i clienti. Per differenziarsi dalla concorrenza Spazio Dev ha introdotto nuove funzionalità impiegando modelli di IA capaci di predire *trend* futuri di acquisto e di anticipare i bisogni dei clienti.
- **Marketing:** Gli esperti di marketing creano contenuti di alta qualità, pertinenti e ottimizzati, atti a migliorare la presenza *online* del brand. Per aumentare l'*engagement* vengono studiate apposite campagne social e di posta elettronica.

¹Spazio Dev. URL: <https://spaziodev.eu/>.

1.4 Clientela

La clientela di Spazio Dev è molto variegata ed è suddivisibile nelle seguenti categorie:

- Imprese di varie dimensioni specializzate in commercio e manifattura che necessitano di migliorare la propria presenza nel web, la gestione della contabilità, la logistica e le risorse umane.
- *Startup* in cerca di visibilità e di strategie di marketing.
- Liberi professionisti come avvocati, medici, eccetera.

1.5 Prodotto di punta

RelAI² (Fig. 1.2) rappresenta in tutto e per tutto l’azienda poiché nasce dall’unione di tutti i suoi *core business*: sviluppo web, integrazione IA e marketing. Il software si occupa del *CRM*^[gl] (*Customer Relationship Management*) tramite una piattaforma *cloud* che consente di personalizzare le relazioni con i clienti e di incrementare le vendite grazie a campagne di comunicazione su misura. La sua architettura modulare e le numerose configurazioni si adattano perfettamente in molti settori. L’IA implementata è utile per predire futuri *trend* di acquisto fondamentali per la creazione di strategie di marketing efficaci.



Figura 1.2: Logo dell’applicazione RelAI

²RelAI. URL: <https://tessere.spaziodev.eu/>.

Capitolo 2

Processi e metodologie

In questo capitolo sono riassunti i processi e le metodologie che lo studente ha applicato durante lo sviluppo del progetto di stage.

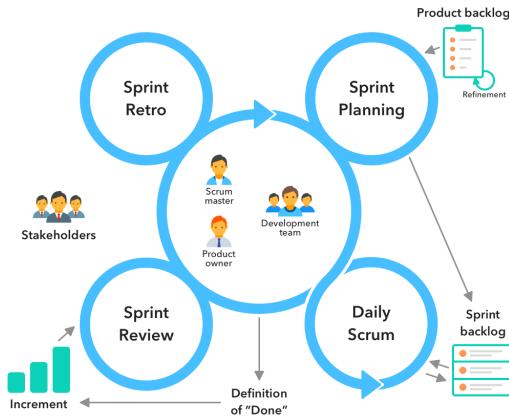
2.1 Ciclo di vita del software

Il ciclo di vita del software è l'insieme delle attività e dei processi che tutti i membri coinvolti devono seguire per la buona riuscita del prodotto. L'adozione di un modello di ciclo di vita è fondamentale per rispettare le scadenze, rimanere all'interno del *budget* ed evitare conflitti all'interno e all'esterno del team.

2.1.1 Filosofia Agile

Per favorire la collaborazione l'azienda si affida alla filosofia *agile*¹ e in particolare al modello *scrum* (Fig. 2.1) che ha come unità fondamentale lo *sprint* ossia un periodo di tempo breve (1 settimana) che viene utilizzato per sviluppare e testare nuove funzionalità; grazie a questi avanzamenti rapidi è possibile fornire codice flessibile e rapidamente adattabile alle nuove esigenze.

¹Agile. URL: <https://www.atlassian.com/it/agile>.

**Figura 2.1:** Schema del modello scrum

2.1.2 Tecnologie di supporto

- Plane (Fig. 2.2): è un *ITS (Issue Tracking System)* utilizzato per monitorare l'avanzamento del progetto e per assegnare a ciascun membro i compiti da effettuare; consente a tutte le parti coinvolte di avere una visione di insieme su quello che bisogna realizzare per portare a compimento lo sviluppo. In particolare è stato scelto perché altamente personalizzabile e per la sua natura *open source*^[gl].

**Figura 2.2:** Logo di Plane

- Telegram (Fig. 2.3): è una piattaforma di messaggistica completa con molte funzionalità, viene sfruttata dall'azienda per coordinare i vari team di lavoro tramite l'ausilio di chat di gruppo. Fornisce inoltre molte funzionalità avanzate come *bot*^[gl] utili per i più svariati casi d'uso.

**Figura 2.3:** Logo di Telegram

2.2 Gestione della configurazione

La gestione della configurazione è il processo tramite il quale è possibile identificare, controllare e coordinare i vari componenti del software e le risorse a esso associate durante l'intero ciclo di vita del prodotto. Grazie a questo processo gli sviluppatori possono tenere traccia delle modifiche e gestire le varie versioni garantendo il funzionamento del prodotto nel tempo.

2.2.1 Tecnologie di supporto

Le tecnologie utilizzate per il versionamento del prodotto sono:

- Git (Fig. 2.4): è un *DVCS (Distributed Version Control System)* ampiamente diffuso che consente di gestire e monitorare i cambiamenti del codice e della documentazione. Questo strumento è fondamentale per avere una storia completa dello sviluppo da poter sfruttare in caso di necessità. Il codice viene inserito all'interno di un *repository*, ossia una cartella che contiene tutti i file relativi al progetto, dove è possibile salvare ogni cambiamento tramite *commit*. Il *commit* segnala ogni modifica e la data in cui viene attuata, è dunque sufficiente spostarsi tra i vari *commit* per tornare a una versione più o meno aggiornata.



Figura 2.4: Logo di Git

- Gitea (Fig. 2.5): è un servizio che supporta le *repository* di Git, è simile a GitHub, Bitbucket e GitLab, ma ha il vantaggio di essere *self hosted*^[gl]. Essendo tutti i *repository* aziendali mantenuti sul server interno, le comunicazioni atte al versionamento hanno risposta molto rapida.



Figura 2.5: Logo di Gitea

Regole di branch e commit

Per standardizzare l'utilizzo delle tecnologie di versionamento l'azienda adotta due convenzioni:

- **Git Flow²** (Fig. 2.6): è un modello di *branching* ideato per avere un miglior controllo delle *release*.
- **Conventional Commits³**: è una specifica basata su regole, che consentono una facile lettura dei *commit* sia da parte di utenti che da parte di strumenti automatici.

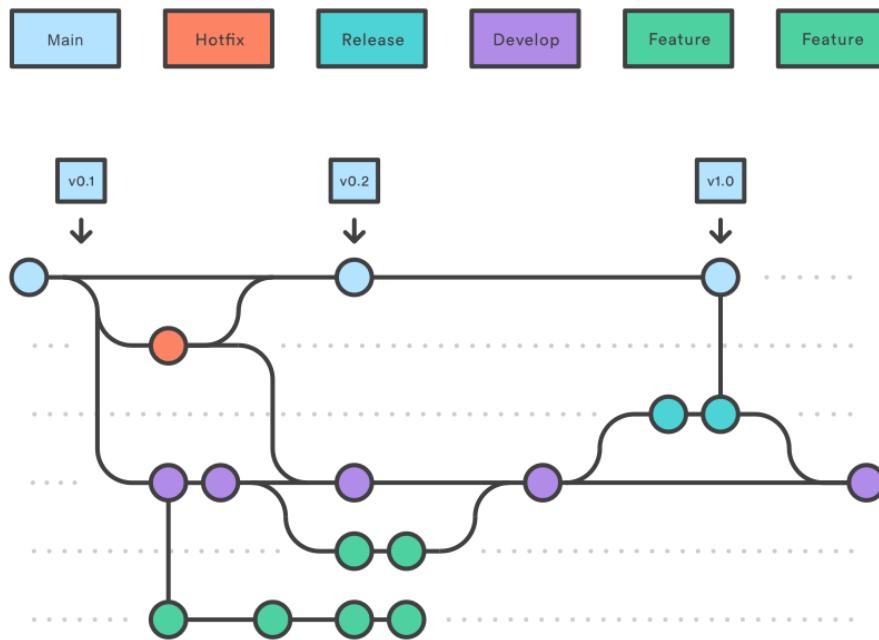


Figura 2.6: Esempio di Git Flow

²Git Flow. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>.

³Conventional Commits. URL: <https://www.conventionalcommits.org/en/v1.0.0/>.

Capitolo 3

Il progetto

Questo capitolo contiene le informazioni preliminari che sono state fornite allo studente e le misure preventive adottate per iniziare la produzione.

3.1 Analisi del progetto

Il progetto si basa sull'integrazione di un nuovo *workflow*^[g] all'interno di un applicativo preesistente; lo studente deve occuparsi dello sviluppo di una soluzione in grado di migliorare la portata dell'azienda per quanto riguarda il raggiungimento di nuovi clienti.

3.1.1 SalesCRM

SalesCRM è il **CRM** che viene utilizzato dai commerciali interni all'azienda; contiene tutte le informazioni relative ai contatti che sono stati raggiunti dal reparto vendite. Gli utenti sono in grado di immagazzinare i dati raccolti su clienti e potenziali clienti tramite un interfaccia intuitiva sviluppata utilizzando **Laravel** e **Filament**.

Componenti principali

- *Homepage* (Fig. 3.1) contenente grafici informativi.
- Il modulo contenente il calendario (Fig. 3.2) è utile per pianificare appuntamenti e visite.
- La sezione della mappa (Fig. 3.3) viene sfruttato per conoscere il posizionamento dei clienti e organizzare incontri locali.

3.1. ANALISI DEL PROGETTO

9

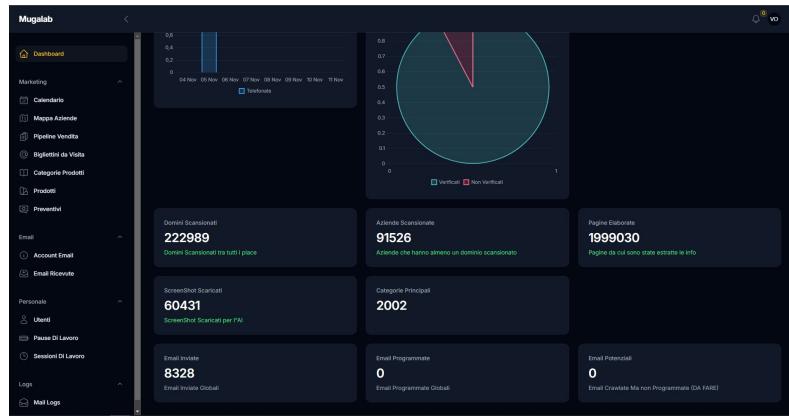


Figura 3.1: Homepage di SalesCRM

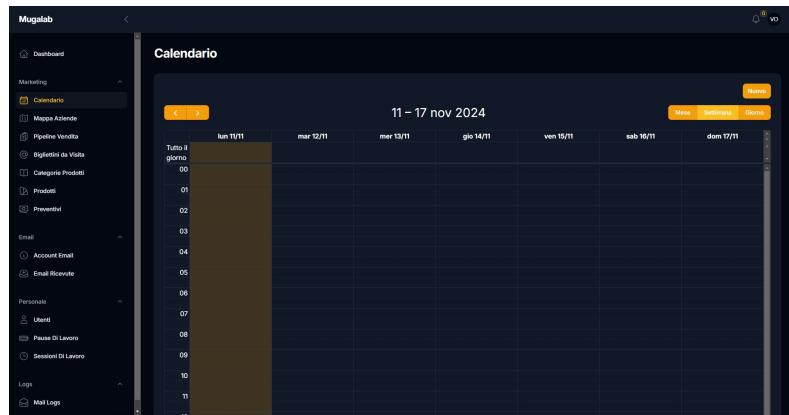


Figura 3.2: Calendario contenuto all'interno di SalesCRM

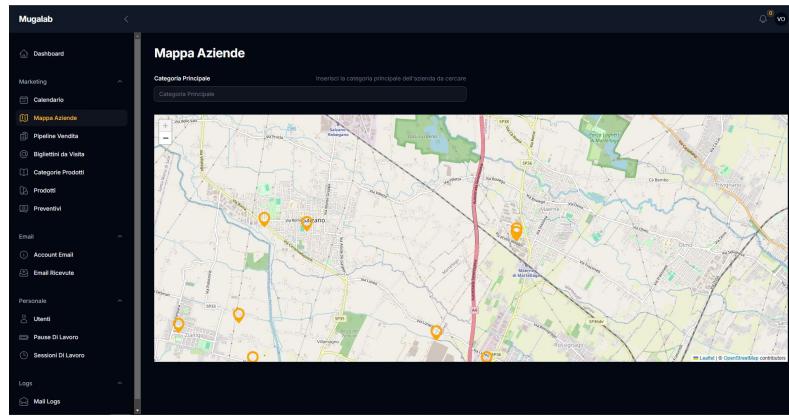


Figura 3.3: Mappa contenuta all'interno di SalesCRM

3.1.2 Integrazione

- Raccolta di dati: l'applicativo ha una funzionalità di *web scraping*^[gl] che raccoglie i link dei siti web di tutti i potenziali clienti, il compito del tirocinante è quello di creare un *workflow* automatico che si integri con il *web scraper* per acquisire screenshot delle varie pagine del sito web.
- Clusterizzazione: la fase successiva alla raccolta dei dati consiste nello sviluppo di una IA addestrata in maniera non supervisionata che sia in grado di suddividere le varie immagini in *cluster*, differenziati in base alle caratteristiche riconosciute in ogni sito.
- IA classificativa: in seguito alla raccolta di un *dataset*^[gl] di dimensioni congrue si procede alla suddivisione manuale degli screenshot precedentemente clusterizzati su una base qualitativa (siti migliorabili e siti ottimi); questo processo viene svolto con l'ottica dell'addestramento di una IA classificativa. L'IA addestrata in maniera supervisionata ha l'obiettivo di affidare un punteggio in valori centesimali a ogni sito.
- Invio di e-mail automatizzato: l'automazione della posta elettronica procede con l'invio di e-mail personalizzate ai proprietari dei siti web che hanno ricevuto una valutazione scarsa, per offrire loro un servizio di miglioramento.

3.2 Analisi e gestione dei rischi

Durante l'analisi del progetto lo stagista ha individuato alcuni rischi in cui potrà incorrere. Nella lista seguente sono elencati i rischi e le soluzioni ideate.

1. Rimodulazione dell'attività

Descrizione: dopo un mese dall'inizio dello stage lo studente è stato riposizionato sul progetto attuale scartando il progetto precedente e trovandosi dunque con meno settimane a disposizione per la produzione.

Soluzione: ridimensionamento delle attività e richieste di supporto più frequenti.

2. Approccio sperimentale

Descrizione: il progetto prevede dei contributi originali e sperimentali per cui non sono disponibili soluzioni già pronte.

Soluzione: auto apprendimento tramite tutorial online e richiesta di coinvolgimento di colleghi più esperti nell'ambito.

3. Costo dell'Addestramento

Descrizione: l'addestramento dell'intelligenza artificiale richiede l'utilizzo di potenti GPU di cui spesso l'hardware a disposizione è sprovvisto.

Soluzione: utilizzo del server aziendale per l'addestramento su CPU, ottenendo un compromesso tra costi e tempo di addestramento.

4. Quantità di dati di training insufficiente

Descrizione: l'IA necessita una grande quantità di dati in input per effettuare un training efficace.

Soluzione: aumento manuale del *dataset* e ricerca di *dataset* già pronti online.

5. Risultati della clusterizzazione non soddisfacenti

Descrizione: il *clustering* potrebbe risultare non conforme alle aspettative.

Soluzione: valutare la quantità di cluster da creare e sperimentare con altri metodi di *clustering*.

6. Overfitting del modello

Descrizione: il modello IA fornisce valutazioni accurate solo per le immagini utilizzate durante il training.

Soluzione: sperimentare con metodi per la risoluzione dell'*overfit* (*dropout, cross-validation, ecc...*).

7. Dipendenze delle librerie

Descrizione: il progetto utilizza molte tecnologie e librerie differenti, è probabile che si presentino delle incompatibilità.

Soluzione: utilizzo del minor numero possibile di librerie e verifica delle compatibilità prima dell'inizio della scrittura del codice.

3.3 Obiettivi

Gli obiettivi hanno lo scopo di delineare il percorso che lo studente deve affrontare per portare a termine il progetto nella maniera desiderata dall'azienda. Sono suddivisibili in:

- **O:** obbligatori
- **D:** desiderabili
- **F:** facoltativi

Codice	Descrizione
O01	Implementare un sistema robusto per la cattura degli screenshot, assicurando l'integrazione con il database per l'archiviazione e l'analisi.
O02	Garantire la creazione di una documentazione tecnica completa che supporti sia l'uso che la manutenzione del sistema sviluppato.
O03	Creazione di un IA in grado di suddividere in cluster i siti web.
O04	Creazione di un IA classificativa in grado di assegnare un punteggio ai siti web analizzati.
D01	Aggiungere la valutazione creata dall'IA nel database utilizzato dal CRM.
D02	Automatizzare l'invio di e-mail alle aziende che hanno ottenuto una valutazione scarsa.
F01	Migliorare la raccolta dei dati delle aziende dal web
F02	Aggiornare automaticamente la valutazione dei siti web

Tabella 3.1: Tabella degli obiettivi

Capitolo 4

Analisi dei requisiti

L'analisi dei requisiti è alla base della comprensione del prodotto da sviluppare, è utile per chiarire eventuali dubbi e per procedere in maniera spedita evitando perdite di tempo future

4.1 Casi d'uso

Per rendere più chiari i casi d'uso sono stati utilizzati dei diagrammi di tipo UML^[gl]. Questo tipo di diagrammi descrivono funzioni e servizi forniti dal sistema agli attori che lo utilizzano. Essendo il progetto l'implementazione di un *workflow* automatico, le interazioni dell'utente devono essere minime. Per questo i casi d'uso sono pochi e molto sintetici.

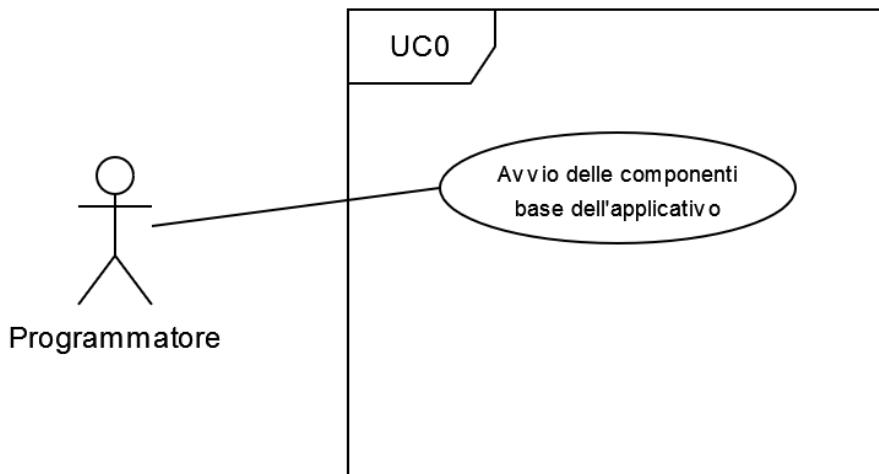


Figura 4.1: Use Case - UC0: Scenario principale

UC0: Scenario principale

Attori Principali: Programmatore.

Precondizioni: Il programmatore ha avviato l'ambiente di programmazione integrato.

Descrizione: Avvio delle componenti base dell'applicativo.

Postcondizioni: Il sistema è pronto per eseguire il *workflow*.

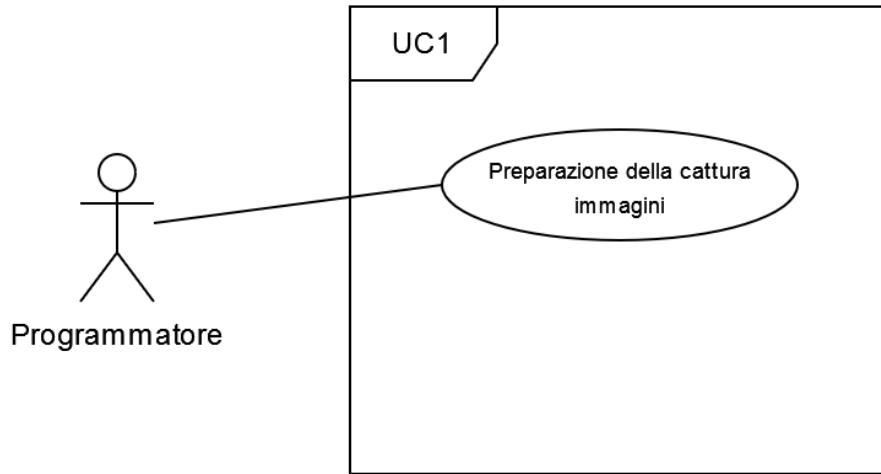


Figura 4.2: Use Case - UC1: Acquisizione screenshot

UC1: Acquisizione screenshot

Attori Principali: Programmatore.

Precondizioni: Il *web scraper* ha già raccolto i link alle pagine dei siti web delle aziende.

Descrizione: Il programmatore prepara il *workflow* per acquisire in maniera automatica gli screenshot delle pagine raccolte in precedenza.

Postcondizioni: Gli screenshot vengono salvati nel database, pronti per la fase di analisi e classificazione.

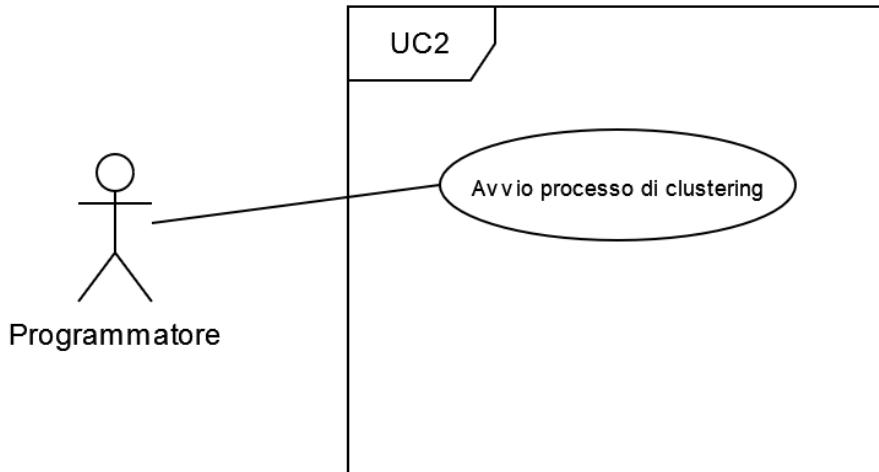


Figura 4.3: Use Case - UC2: Clusterizzazione degli screenshot

UC2: Clusterizzazione degli screenshot

Attori Principali: Programmatore.

Precondizioni: Gli screenshot delle pagine sono stati acquisiti e salvati nel database.

Descrizione: Il programmatore avvia il processo di *clustering*, che organizza gli screenshot in *cluster* in base alle *feature* *[gl]estratte*.

Postcondizioni: Gli screenshot vengono clusterizzati e inseriti in apposite cartelle.

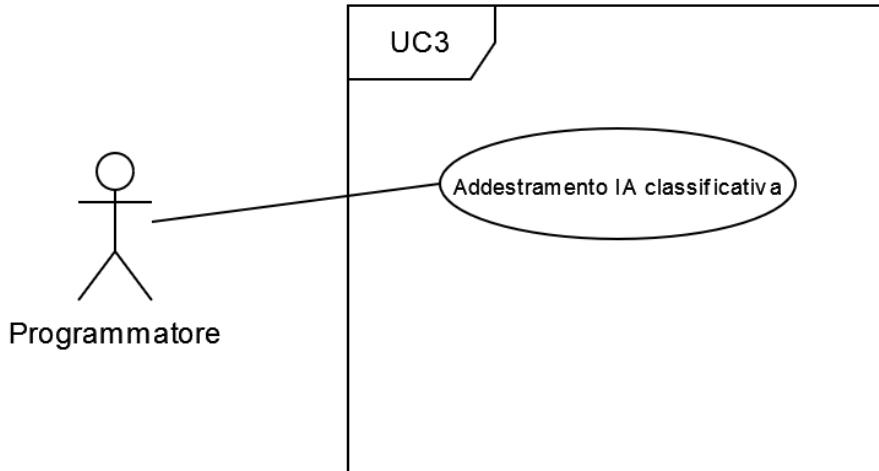


Figura 4.4: Use Case - UC3: IA classificativa

UC3: Valutazione qualitativa e classificazione

Attori Principali: Programmatore.

Precondizioni: Gli screenshot sono stati inseriti in cartelle.

Descrizione: Lo sviluppatore addestra l'IA classificativa sfruttando i cluster precedentemente creati, e creando un *dataset*.

Postcondizioni: Ogni sito ottiene un punteggio qualitativo che viene salvato nel database per operazioni future.

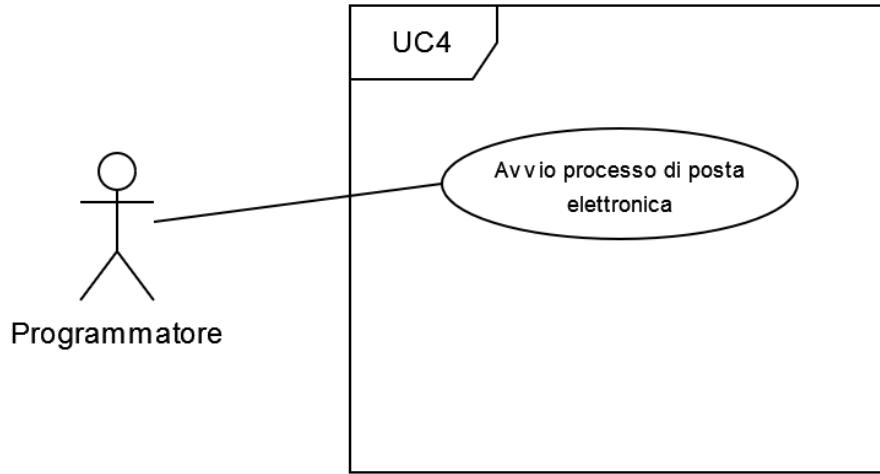


Figura 4.5: Use Case - UC4: invio e-mail

UC4: Automazione dell'invio di e-mail

Attori Principali: Programmatore.

Precondizioni: I siti web contenuti nel database hanno ricevuto una valutazione.

Descrizione: Il sistema invia automaticamente e-mail personalizzate ai proprietari dei siti con punteggi di qualità bassi, offrendo servizi di miglioramento.

Postcondizioni: Le e-mail vengono inviate con successo ai destinatari.

Capitolo 5

Strumenti e tecnologie

L'obiettivo principale di questo capitolo è l'illustrazione delle tecnologie e degli strumenti ausiliari utilizzati per raggiungere lo scopo finale del progetto

5.1 Strumenti

Gli strumenti di supporto adoperati per il progetto sono elencati nella lista seguente.

5.1.1 Visual Studio Code

Visual Studio Code (Fig. 5.1): è un ambiente di sviluppo integrato, disponibile per Linux, macOS e Windows. È un'applicazione che supporta la maggior parte dei linguaggi di programmazione ed è quindi molto vantaggiosa per lavorare a un progetto multi linguaggio senza dover cambiare ambiente. Un'altra delle funzionalità principali è la fornitura di numerose estensioni che semplificano il processo di scrittura e verifica del codice.



Figura 5.1: Logo di Visual Studio Code

5.1.2 PhpMyAdmin

PhpMyAdmin (Fig. 5.2): è una *web app*[gl]scritta utilizzando il linguaggio di programmazione PHP, che offre la capacità di gestione di un database MySQL attraverso un *browser* qualsiasi. Consente la creazione di tabelle, l'inserimento, la modifica e l'interrogazione dei dati. Fornisce un'interfaccia grafica per la visione d'insieme e per le operazioni amministrative.



Figura 5.2: Logo di phpMyAdmin

5.1.3 remoteripple

Remote Ripple (Fig. 5.3): è un software per l'accesso remoto, che viene utilizzato dallo studente per avviare i programmi presenti nel server aziendale.



Figura 5.3: Logo di Remote Ripple

5.2 Tecnologie

Nelle sezioni seguenti viene data una spiegazione di tutte le tecnologie utilizzate.

5.2.1 Python

Versione: 3.9.0

Python (Fig. 5.4) è un linguaggio di programmazione ampiamente utilizzato in settori come il *data mining* e l'intelligenza artificiale. Offre solide basi per l'integrazione con disparati linguaggi, ma il vantaggio più di risalto è la presenza di numerose librerie che aiutano lo sviluppatore a velocizzare il processo di codifica; in particolare sono state utilizzate le seguenti librerie:

Tensorflow-gpu 2.10.1

TensorFlow è un *framework*^[gl] utilizzato per il *machine learning*, su di esso si basa tutta la struttura del progetto.

L'unità di base del *framework* è il tensore, ossia un vettore di n dimensioni, tutte le operazioni possibili vengono eseguite su elementi appartenenti a questo tipo. Inoltre contiene modelli preaddestrati molto utili per eseguire determinate operazioni in maniera più rapida. Viene utilizzata la versione 2.10 perché è l'ultima ad avere il supporto per GPU su sistema operativo Windows.

Keras 2.10.0

Keras è un *API*^[gl] (*Application Programming Interface*) sviluppata per rendere la codifica di IA più semplice per la comprensione umana. Minimizza le azioni richieste all'utente per i casi d'uso più comuni, rendendo l'utilizzo di *framework* come JAX, TensorFlow e PyTorch più *user friendly*.

Scikit-Learn 1.5.2

Scikit-learn è una libreria che offre semplici *tool* utili per la classificazione, la regressione, il clustering, eccetera.

Matplotlib 3.9.2

Matplotlib è una libreria che viene utilizzata per la stampa dei grafici ottenuti durante le varie compilazioni.

Numpy 1.26.4

Numpy è il pacchetto che si occupa della gestione di *array* multi dimensionali fornendo anche un'ampia gamma di funzioni matematiche da applicare agli stessi.

OpenCV-python 4.10.0

OpenCV è una libreria utilizzata per sviluppare applicazioni di tipo *computer vision*^[gl]. Nell'ambito del progetto il suo scopo primario è l'elaborazione delle immagini per adattarle all'utilizzo da parte delle IA.

Puppeteer 2.0.0

Puppeteer è una libreria di automazione per browser basati su chromium.

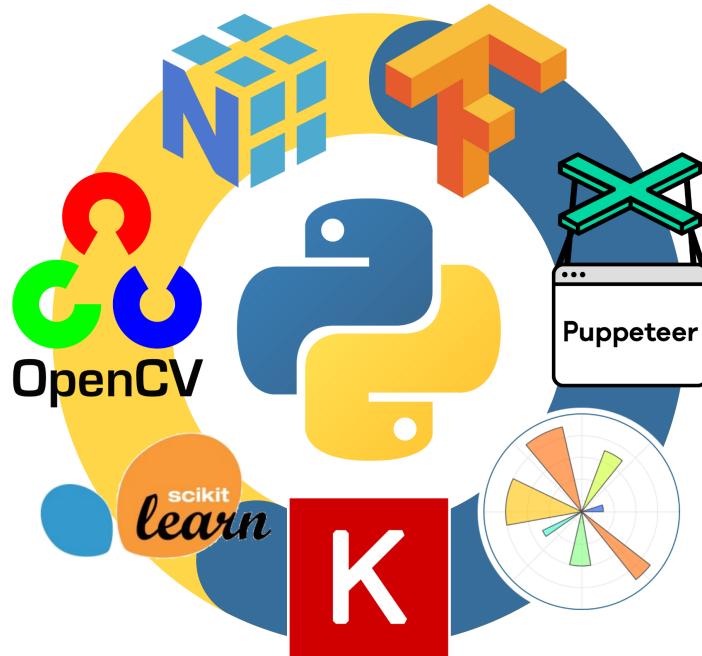


Figura 5.4: Python e librerie utilizzate

5.2.2 CUDA

Versione: 11.2

CUDA (Fig. 5.5) è un *toolkit* che fornisce un ambiente di sviluppo in grado di creare applicazioni che sfruttino a pieno le GPU di NVIDIA. Nel caso di studio CUDA è un elemento portante in quanto fornisce a Python e alle sue librerie la capacità di eseguire calcoli in maniera più rapida utilizzando la GPU. La GPU hanno un'architettura diversa rispetto alle CPU che consente loro di eseguire parallelamente più operazioni, per questo sono molto portate per l'addestramento delle IA. La versione 11.2 viene utilizzata perché è l'unica compatibile con [Tensorflow-gpu 2.10.1](#).



Figura 5.5: Logo di CUDA

5.2.3 Docker

Versione: 27.2.0

Docker (Fig. 5.6) è un sistema che consente di semplificare lo sviluppo delle applicazioni fornendo un ambiente isolato e facilmente riproducibile. Tale ambiente viene definito come container, al suo interno vengono eseguite le immagini di tutti i componenti necessari per il funzionamento dell'applicativo; senza che sia necessario installarli sulla propria macchina.

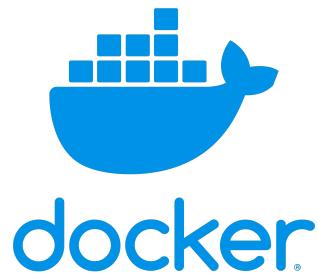


Figura 5.6: Logo di Docker

5.2.4 DDEV

Versione: 1.23.2

DDEV (Fig. 5.7) è un *tool* per velocizzare il lancio di ambienti di sviluppo web in locale; consente di utilizzare il *workflow* di Docker in maniera più rapida.



Figura 5.7: Logo di DDEV

5.2.5 WSL

Versione: 2.3.24.0

WSL (Fig. 5.8) è un sottosistema che viene utilizzato per eseguire applicazioni create per il sistema operativo Linux su Windows.



Figura 5.8: Logo di WSL

5.2.6 Laravel

Versione: 11.29.0

Laravel (Fig. 5.9) è un *framework* ideato per la creazione di web-application scritto in PHP. Tra i suoi vantaggi abbiamo la semplicità della sintassi, il supporto continuo della community e la presenza di numerosi tutorial che è fondamentale per fornire a sviluppatori alle prime armi l'aiuto necessario. Laravel si basa sul pattern architettonicale *MVC* (*Model View Controller*):

- **Model:** accede ai dati contenuti nell'applicativo.
- **View:** mostra i dati contenuti nel model e gestisce le interazioni con gli utenti.
- **Controller:** riceve le operazioni degli utenti e modifica di conseguenza i dati contenuti nel model e la visualizzazione mostrata dalla view.

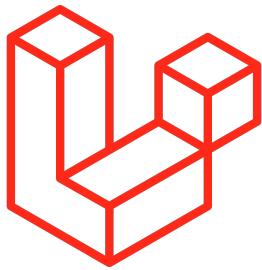


Figura 5.9: Logo di Laravel

5.2.7 Filament

Versione: 3.2.121

Filament (Fig. 5.11) è un *framework* basato su **Laravel** che fornisce la possibilità di generare componenti per l'interfaccia grafica (Fig. 5.10) in maniera rapida.

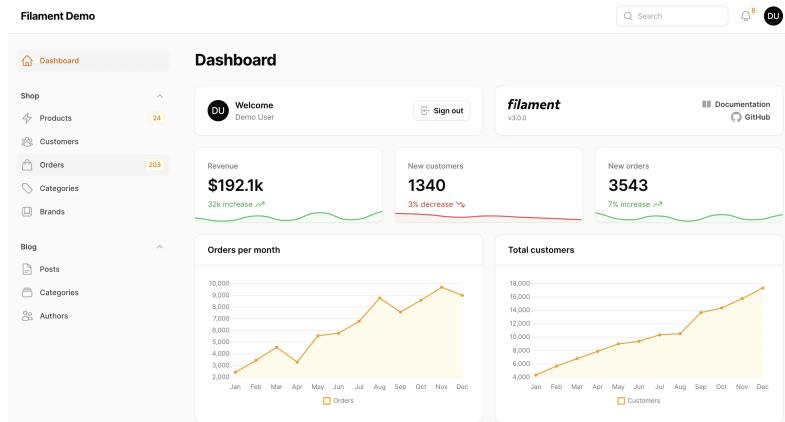


Figura 5.10: Esempio di interfaccia grafica creata in Filament



Figura 5.11: Logo di Filament

5.2.8 MySQL

Versione: 8.0.36

MySQL (Fig. 5.12) è un database relazionale *open source*.



Figura 5.12: Logo di MySQL

Capitolo 6

Machine learning

Per comprendere al meglio gli argomenti trattati nel capitolo successivo è necessaria un'introduzione teorica

Il *machine learning* è un'applicazione della statistica che si concentra sullo sviluppo di algoritmi in grado di imparare dai dati che vengono loro forniti in *input*. Da un punto di vista più filosofico il *machine learning* è interessante perché sviluppando la nostra conoscenza su di esso stiamo di conseguenza migliorando la nostra comprensione dei principi su cui si sorregge l'intelligenza¹.

Gli algoritmi di *machine learning* si suddividono in due categorie principali:

- **Apprendimento supervisionato:** gli algoritmi che appartengono a questa categoria apprendono quali risultati generare seguendo la guida di un set di dati etichettati e con un *output* predefinito².
- **Apprendimento non supervisionato:** gli algoritmi non supervisionati, anche conosciuti come apprendimento automatico, analizzano e raggruppano *dataset* non etichettati. Questi algoritmi riconoscono raggruppamenti di dati (*cluster*) senza la necessità dell'intervento umano³.

Tali algoritmi vengono utilizzati nel progetto per il processo di classificazione.

La classificazione è un metodo che consente di prevedere a quale classe un determinato *input* appartenga. Quindi generalizzando un'entità viene rappresentata come vettore in uno spazio delle *feature*, tipicamente R^n , successivamente vengono eseguite delle operazioni sul vettore che producono un valore preso da un insieme di etichette L noto a priori, il classificatore è quindi una funzione $R^n \rightarrow L$.

Un classico esempio di classificazione è fornito dal *dataset* Iris⁴, è uno dei primi *dataset* utilizzati in letteratura per sperimentare i metodi di classificazione; contiene 3 classi composte da 50 istanze ciascuna, ogni classe si riferisce a un tipologia differente di iris (Fig. 6.1).

In fase di addestramento vengono utilizzate le *feature* e le etichette (Fig. 6.2) di ciascuna pianta e in fase di test l'algoritmo utilizza le caratteristiche apprese per eseguire una classificazione su dati che non ha ricevuto durante il *training*.

¹Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016, p. 97.

²Machine learning, deep learning e reti neurali. URL: <https://www.ibm.com/it-it/topics/machine-learning>.

³Ibid.

⁴Iris dataset. URL: <https://archive.ics.uci.edu/dataset/53/iris>.

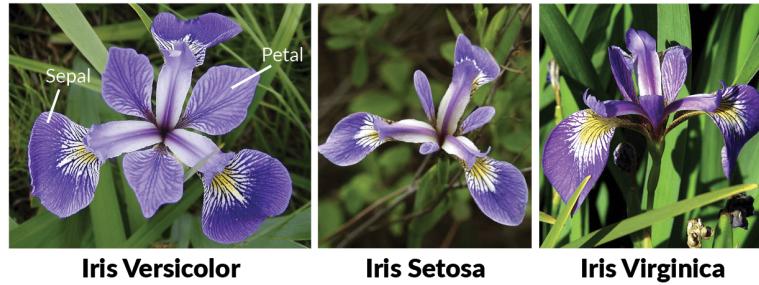


Figura 6.1: Tre tipologie di pianta

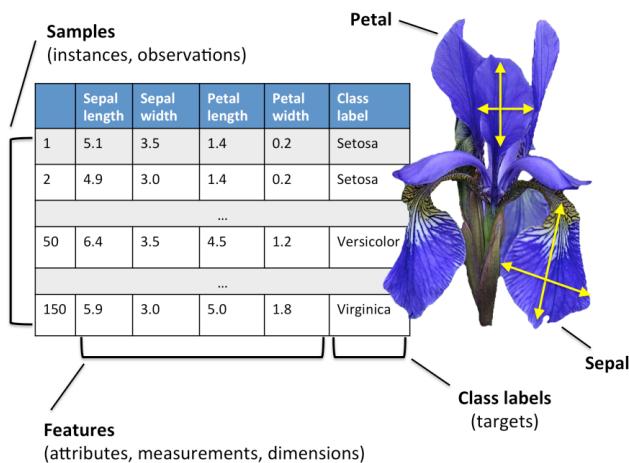


Figura 6.2: Schema delle feature

L'introduzione al *machine learning* è utile per presentare le applicazioni di tale disciplina utilizzate ai fini del progetto:

- Clustering
- PCA
- Reti neurali
- Deep learning
- Autoencoder

Gli elementi citati in precedenza vengono introdotti e trattati nelle sezioni seguenti.

6.1 K-means

Il K-means è un algoritmo non supervisionato che viene utilizzato per il *clustering*, ossia per la suddivisione del *dataset* in gruppi che contengono caratteristiche simili. Suddivide un set di dati in gruppi simili sulla base della distanza tra i loro centroidi. Il centroide, è la media di tutti i punti presenti all'interno del *cluster*.

Il numero K sta a indicare quanti *cluster* dovranno essere assegnati dall'algoritmo, più il numero di *cluster* è elevato più essi saranno piccoli e dettagliati al contrario se i *cluster* sono pochi il risultato saranno dei *cluster* più grandi ma meno dettagliati (Fig. 6.3).

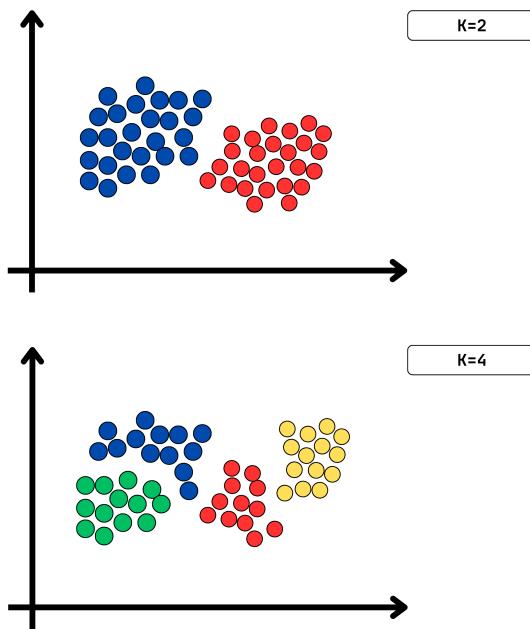


Figura 6.3: Esempio di clustering

Ad esempio se volessimo clusterizzare un insieme di animali composto da mammiferi e volatili utilizzando K=2 l'algoritmo creerebbe due *cluster* uno contenente i mammiferi e uno i volatili (*cluster* grandi e poco dettagliati). Se aumentassimo il numero di *cluster* l'algoritmo sarebbe in grado di essere più specifico andando a creare una quantità molto più elevata di *cluster* ben dettagliati contenenti meno elementi. Quindi è importante scegliere con cura il numero iniziale K da assegnare, per semplificare questa operazione esistono dei metodi di supporto:

- Il metodo del gomito sfrutta l'inerzia, ossia la somma delle distanze al quadrato dei punti dal centroide (*SSE*, *Sum of Squared Errors*). Con l'aumento dei *cluster*, l'inerzia diminuisce, ma dopo un certo punto, la riduzione diventa superflua, venendo a creare una curva a gomito. Il punto in cui la curva si piega indica il numero di *cluster* da utilizzare.

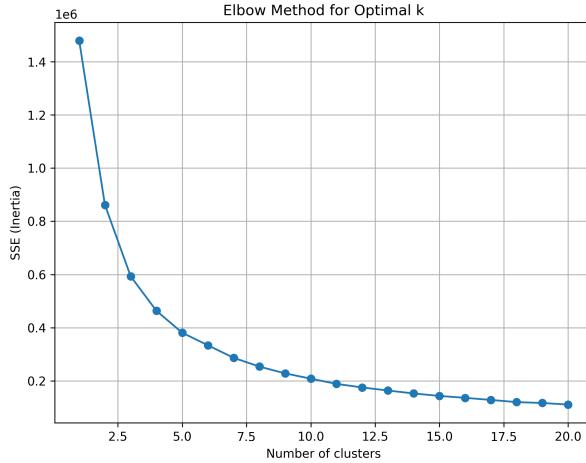


Figura 6.4: Grafico rappresentante il metodo del gomito

- Il Silhouette score indica quanto i punti appartenenti a un *cluster* sono simili tra loro e quanto differiscono dai punti contenuti negli altri *cluster*. Varia tra -1 e 1, se il valore è vicino a 1 il punto è all'interno del suo *cluster* e ben separato dai *cluster* limitrofi, se è vicino a 0 il punto è confinante tra due *cluster* infine un valore negativo indica l'assegnazione a un *cluster* sbagliato.

6.2 PCA

L'analisi dei componenti principali è un algoritmo non supervisionato in grado di estrarre le informazioni più rilevanti da *dataset* di grandi dimensioni, riducendo così la complessità del modello ed evitando che si presenti il fenomeno della "maledizione della dimensionalità"⁵.

La maledizione della dimensionalità si verifica all'aumentare della dimensionalità dei dati: con l'aumentare delle dimensioni, il volume dello spazio aumenta in maniera esponenziale, rendendo difficile l'elaborazione da parte degli algoritmi di *machine learning*, che dovranno lavorare su dati molto sparsi con conseguenze negative per le prestazioni.

Tra gli ulteriori vantaggi che hanno portato all'adozione della PCA sono presenti la riduzione dei tempi di calcolo dovuta alla minor quantità di *feature* e la possibilità di visualizzare i risultati del *clustering* su un grafico cartesiano bidimensionale (Fig. 6.5) aumentando il livello di comprensione dell'utente.

⁵ Cos'è l'analisi delle componenti principali (PCA)?. URL: <https://www.ibm.com/it-it/topics/principal-component-analysis>.

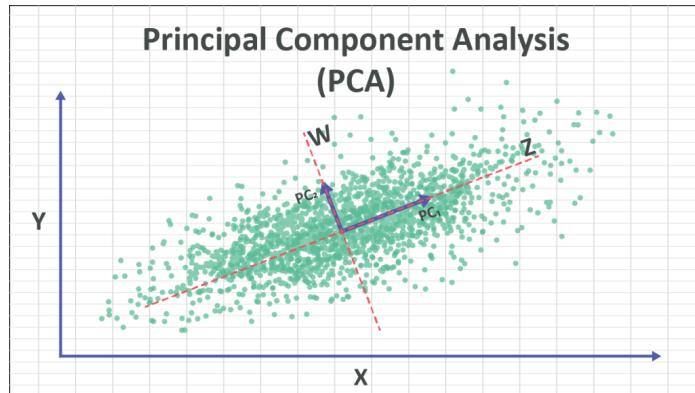


Figura 6.5: Grafico contenente PCA

6.3 Reti neurali

Le reti neurali (Fig. 6.6) sono un processo di *machine learning* che sfrutta nodi interconnessi tra di loro (anche chiamati neuroni) in una struttura a strati ispirata al cervello umano (da qui *neural network*). In genere consistono di un sistema adattivo utilizzato dai computer per imparare dagli errori commessi e automigliorarsi⁶.

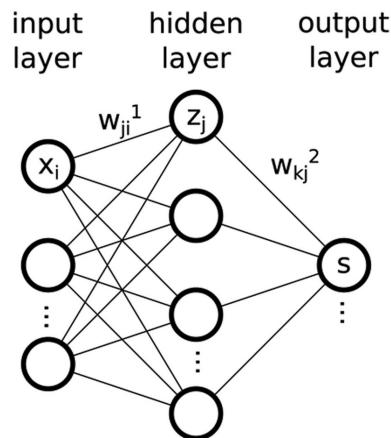


Figura 6.6: Esempio di rete neurale

Struttura

All'interno di una rete neurale sono presenti:

- **Input layer:** riceve l'*input* che la rete deve elaborare.
- **Hidden layer:** possono essere uno o più e si occupano della elaborazione del dato in *input*.
- **Output layer:** mostra l'*output* dell'elaborazione.

⁶ Cos'è una rete neurale? URL: <https://aws.amazon.com/it/what-is/neural-network/>.

6.3.1 Apprendimento

Gli strati densi (*dense layer*) sono alla base della rete neurale e sono composti dai neuroni. Ogni *layer* è interconnesso con lo strato che lo precede e con quello successivo, in particolare ogni neurone dello strato è connesso a ogni neurone dello strato successivo.

Funzionamento del neurone

- Ogni connessione ha un peso che indica la forza della connessione e il suo valore può essere positivo o negativo.
- L'*input* che un neurone riceve da ciascuno dei neuroni a cui è connesso si calcola moltiplicando il segnale proveniente da quel neurone per il peso sulla connessione.
- L'*input* totale del neurone è la sommatoria delle attivazioni che il neurone riceve dagli altri neuroni.
- Lo stato di attivazione finale viene calcolato attraverso una funzione di attivazione (ad esempio RELU o sigmoide, Fig. 6.8).
- L'*output* dei neuroni viene inviato ai neuroni successivi⁷.

Lo schema successivo mostra il procedimento appena descritto (Fig. 6.7)

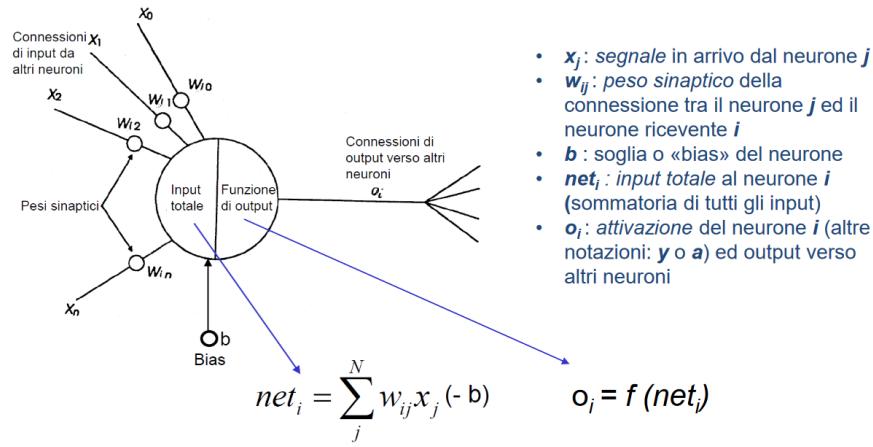
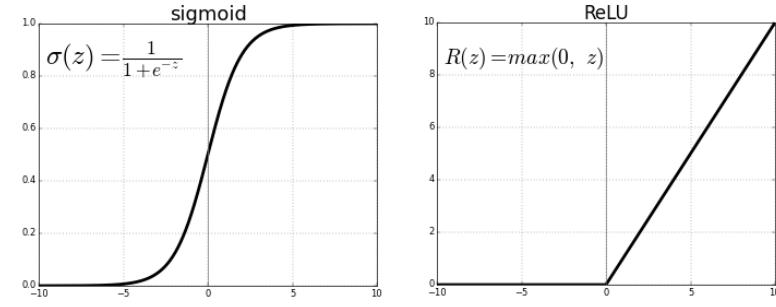


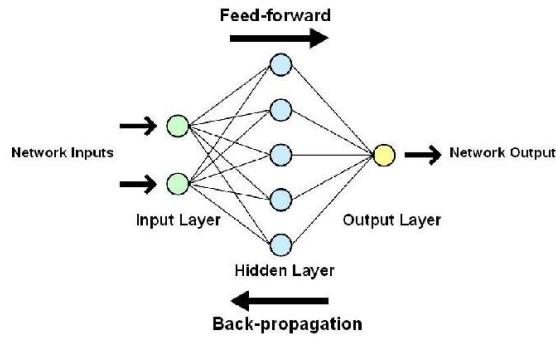
Figura 6.7: Funzionamento del neurone

⁷Marco Zorzi. *La computazione neurale*. University Lecture. 2023.

**Figura 6.8:** Funzioni di attivazione

Fasi di apprendimento

Inizialmente la rete riceve in *input* il vettore, i pesi e i *bias* vengono decisi in maniera casuale. I dati si spostano di strato in strato fino a raggiungere quello di *output* (*forward propagation*, Fig. 6.9), successivamente la funzione di perdita misura il grado di errore e tramite la *backward propagation* avviene l'Aggiornamento dei pesi. Queste fasi vengono ripetute per un numero adeguato di volte (epoch) per ridurre man mano l'errore.

**Figura 6.9:** Forward e backward propagation

6.3.2 Classificazione

I dati da classificare vengono inseriti all'interno della rete neurale che fornisce in *output* la classe di appartenenza di ciascun dato (Fig. 6.10).

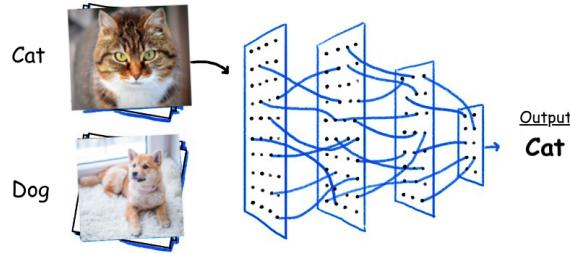


Figura 6.10: Classificazione dei dati

6.4 Deep learning

Con il termine *deep learning* si identificano reti neurali con un numero di strati nascosti maggiore o uguale a due (Fig. 6.11).

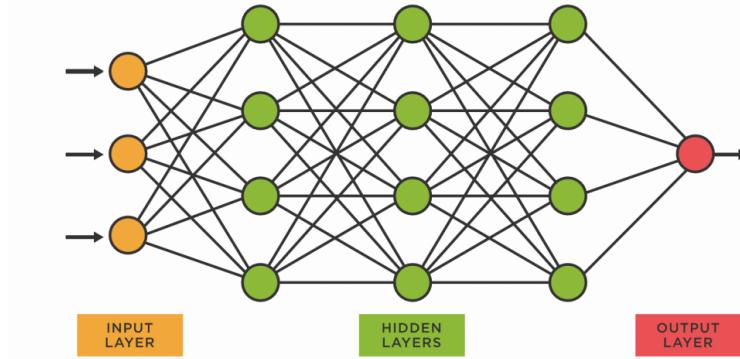


Figura 6.11: Esempio di rete neurale con più strati nascosti

6.4.1 Convolutional neural network

Le reti neurali convoluzionali (CNN, Fig. 6.12) sono un esempio di *deep learning*, esse sfruttano dei *layer* convoluzionali per eseguire la classificazione di dati tridimensionali, come ad esempio le classiche immagini raster.

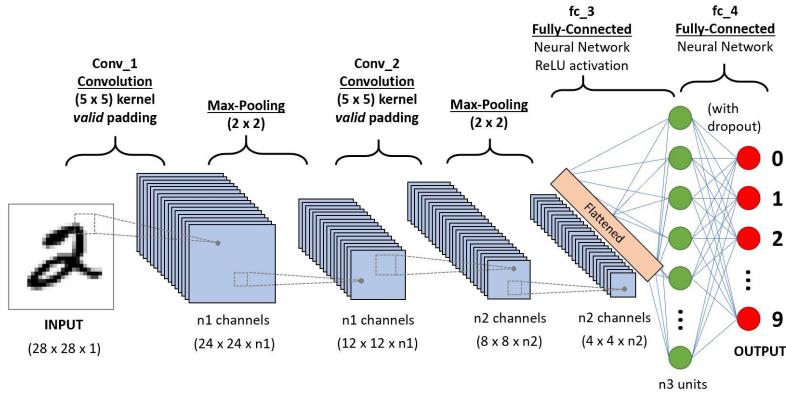


Figura 6.12: Esempio di rete neurale convoluzionale

Layer convoluzionali

I *layer* convoluzionali hanno come scopo l'estrazione di *feature* dalle immagini (texture, bordi, ecc...). Per spiegare il funzionamento di un *layer* convoluzionale è necessario introdurre la nozione di *kernel*.

Il *kernel* (filtro, Fig. 6.13) è una matrice contenente dei pesi, che viene spostata lungo l'immagine effettuando una convoluzione in ogni posizione.

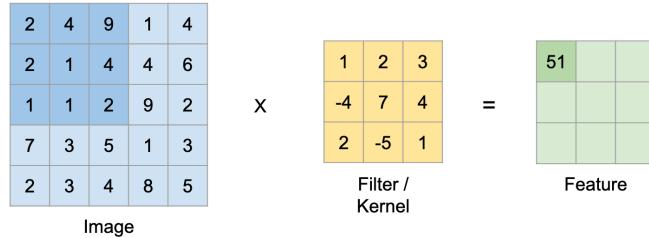


Figura 6.13: Funzionamento del kernel

Esempio con *kernel* di dimensioni 3x3:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

1. Posizionamento del filtro: il *kernel* viene inserito sulla prima zona 3x3 dell'immagine.
2. Moltiplicazione di ogni elemento: ogni peso presente nel *kernel* viene moltiplicato per il valore corrispondente contenuto nella porzione di immagine.

```

PORZIONE DI IMMAGINE
[ 10, 20, 30 ]
[ 40, 50, 60 ]
[ 70, 80, 90 ]

PRODOTTO CON I PESI DEL KERNEL
(-1 * 10) + (0 * 20) + (1 * 30) +
(-1 * 40) + (0 * 50) + (1 * 60) +
(-1 * 70) + (0 * 80) + (1 * 90)

```

3. *Output* della convoluzione: tutti i risultati delle moltiplicazioni vengono sommati tra di loro e il valore ottenuto viene posizionato nella *feature map* nel punto che corrisponde alla regione dell'immagine elaborata.

$$\begin{aligned}
& (-10) + (0) + (30) + \\
& (-40) + (0) + (60) + \\
& (-70) + (0) + (90) = 60
\end{aligned}$$

4. Riposizionamento del filtro: il *kernel* viene riposizionato spostandolo di uno o più pixel (*stride*) e il flusso si ripete fino alla copertura completa dell'immagine.

L'output del *layer* convoluzionale è una serie di *feature map* ognuna corrispondente a ciascun filtro.

```
conv_layer= Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
```

- 32 rappresenta il numero di filtri.
- (3,3) rappresenta la dimensione dei *kernel*.
- activation='relu' indica che come funzione di attivazione verrà utilizzata relu.
- padding='same' consente di mantenere la dimensione dell'immagine in *input* anche dopo l'applicazione dei filtri

Layer di pooling

I *layer* di *pooling* (Fig. 6.14) servono per ridurre le dimensioni delle *feature map* mantenendo le caratteristiche importanti e riducendo di conseguenza la complessità di calcolo.

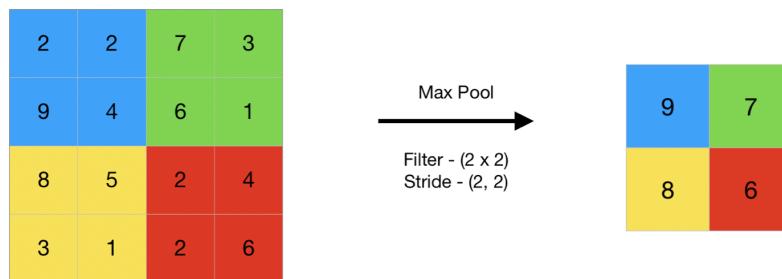


Figura 6.14: Funzionamento del pooling

Funzionamento di un layer di pooling

Esempio di *Max pooling* con matrici 2x2 e *stride* di 2 pixel

```
FEATURE MAP 4x4
[ 1, 3, 2, 4 ]
[ 5, 6, 8, 7 ]
[ 3, 2, 1, 0 ]
[ 1, 2, 4, 3 ]
```

1. Divisione della *feature map* in matrici 2x2:

```
[ 1, 3 ]
[ 5, 6 ]

[ 2, 4 ]
[ 8, 7 ]

[ 3, 2 ]
[ 1, 2 ]

[ 1, 0 ]
[ 4, 3 ]
```

2. Calcolo del valore massimo per ogni matrice:

```
max(1, 3, 5, 6) = 6
max(2, 4, 8, 7) = 8
max(3, 2, 1, 0) = 3
max(1, 2, 4, 3) = 4
```

3. Creazione di una nuova *feature map* contenente i valori massimi:

```
[ 6, 8 ]
[ 3, 4 ]
```

L'*output* del *layer* di *pooling* in questo caso è una *feature map* ridotta da 4x4 a 2x2.

```
MaxPooling2D((2, 2), padding='same')(x)
```

6.5 Autoencoder

L'autoencoder (Fig. 6.15) è un *neural network* in grado di comprimere il suo *input* e di ricostruirlo in maniera simile come *output*.⁸

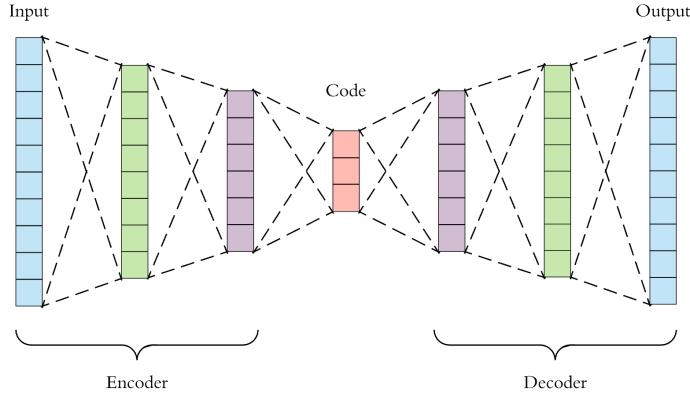


Figura 6.15: Schema generale di un autoencoder

L'architettura si compone di due parti principali:

1. **Encoder:** comprime l'*input* riducendolo di dimensionalità (codifica).
2. **Decoder:** ricostruisce l'*input* iniziale partendo dalla codifica fatta dall'*encoder*

Il *bottleneck* o *latent space* contiene la codifica e sta nel mezzo delle due componenti; è il punto di arrivo del *encoder* e la partenza del *decoder*.

6.5.1 Funzionamento

1. L'*encoder* riceve un *input* x e lo codifica creando una funzione $z = f(x)$
2. Il *bottleneck* contiene la codifica z
3. Il *decoder* cerca di decodificare z tramite la funzione g ottenendo come *output* $\hat{x} = g(z)$.

Il processo di apprendimento ha come obiettivo minimizzare la loss function che misura la differenza tra *input* iniziale e *output* ricostruito dal *decoder*.

$$L(x, \hat{x}) = \|x - \hat{x}\|^2$$

Nell'esempio viene utilizzata la *MSE* (*Mean Squared Error*). Ai fini del progetto la parte interessante dell'autoencoder è proprio l'*encoder* poiché comprime l'immagine in *input* lasciando un set di *feature* nel *bottleneck*; che possono essere utilizzate successivamente per il *clustering*.

⁸Goodfellow, Bengio e Courville, *Deep Learning*, p. 499.

Capitolo 7

Progettazione

Durante la fase di progettazione viene definita l'architettura del software, questa operazione consiste nella suddivisione del sistema in componenti distinti, ognuno con compiti differenti. L'obiettivo è pianificare in maniera chiara tutte le azioni che l'applicativo dovrà svolgere prima di passare effettivamente alla codifica.

7.1 Architettura

L'architettura (Fig. 7.1) pensata prevede l'utilizzo di 4 componenti principali che si occupano di:

1. Acquisire gli screenshot.
2. Effettuare il *clustering* degli screenshot acquisiti.
3. Valutare i siti web.
4. Inviare e-mail promozionali.

La suddivisione in componenti consente di avere un codice più strutturato e specializzato nell'assolvimento di determinati compiti; rende anche la comprensione da parte di altri programmatori più rapida e chiara. I vari moduli comunicano con il database di SalesCRM attraverso l'utilizzo di API che eseguono interrogazioni e inserimenti. Nel caso della comunicazione tra il modulo di *clustering* e quello di valutazione la condivisione di dati avviene in maniera diretta utilizzando la memoria dello stesso PC; questa modalità non si rivela problematica poiché garantisce uno scambio rapido di informazioni che deve essere svolto solo ed esclusivamente in fase di addestramento.

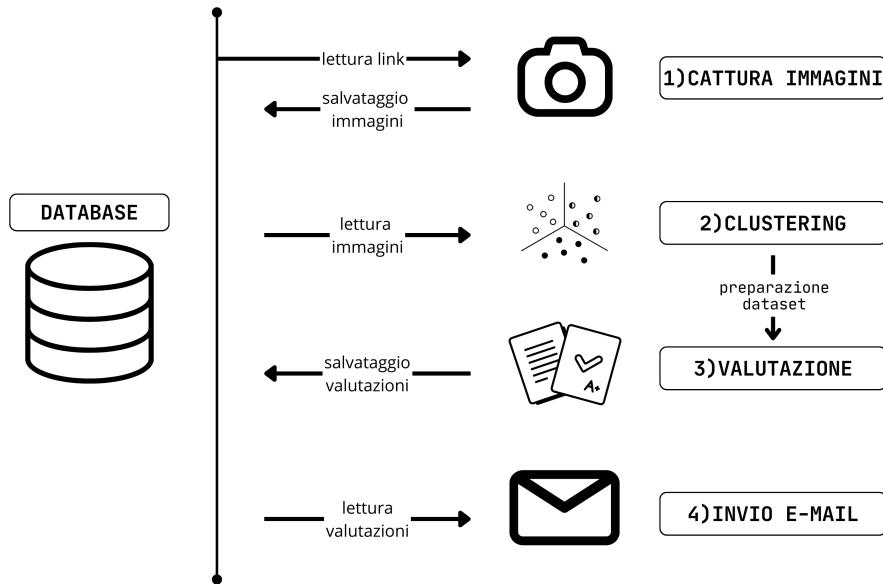


Figura 7.1: Schema architetturale del progetto

7.2 Cattura immagini

La prima fase del *workflow* (Fig. 7.2) è composta da uno script Python che usufruisce della libreria Pypeteer per acquisire gli screenshot dei siti contenuti nel database. Più precisamente un *web scraper* già implementato raccoglie i link delle pagine web dei clienti potenziali, successivamente li carica nel database di SalesCRM, dove verranno infine letti dallo script. Per funzionare Pypeteer necessita di chromium, dopo aver effettuato il controllo per verificare se esso sia presente o meno procede con la lettura dei link. L’automazione apre ogni indirizzo, aspetta qualche secondo e scatta uno screenshot. Tutte le immagini vengono poi convertite in formato Base64 per adattarsi alle tabelle già esistenti e salvate nel database.

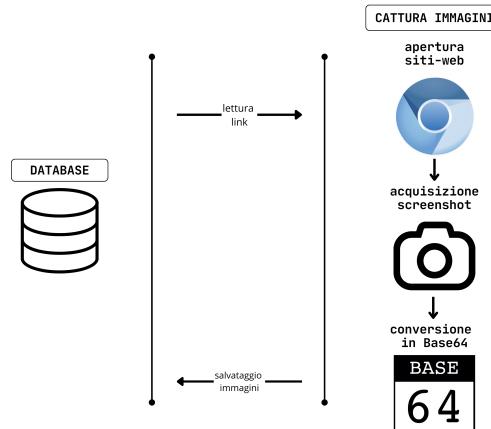


Figura 7.2: Schema della fase di cattura

7.3 Studio preliminare per l'estrazione delle feature

L'estrazione delle *feature* è una fase fondamentale e necessita pertanto di uno studio più approfondito.

7.3.1 Tentativi con autoencoder

Gli *autoencoder* sono stati utilizzati inizialmente poiché si pensava che la soluzione migliore fosse la creazione di un modello *CNN* che si adattasse completamente alle *feature* presenti nei siti web, ma si è rivelato particolarmente difficile per i motivi seguenti:

- Numero elevato di epoche di addestramento.
- Studio della quantità e tipologia degli strati.
- Bilanciamento arduo tra dimensioni accettabili e memoria a disposizione.

Autoencoder con soli strati densi

Il primo tentativo è stato effettuato utilizzando un *autoencoder* che sfrutta esclusivamente *layer* densi (Fig. 7.3), questo metodo nonostante abbia portato a dei risultati si è rivelato problematico visto che per funzionare in maniera efficiente, senza richiedere una GPU troppo prestante, necessita di immagini in bianco e nero di dimensioni ridotte a 128*128 (Fig. 7.4).

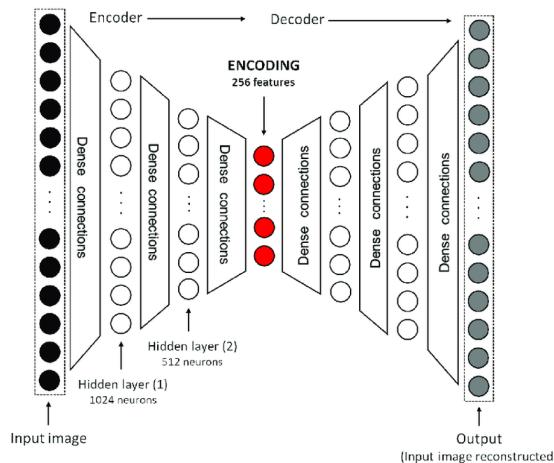


Figura 7.3: Autoencoder con strati densi



Figura 7.4: Ricostruzione immagine ottenuta dal bottleneck dell'autoencoder a strati densi con 150 epoche di addestramento

Autoencoder con strati convoluzionali

L'*autoencoder* con strati convoluzionali (Fig. 7.5) ha consentito un netto miglioramento delle prestazioni e un consumo inferiore della memoria tali da poter aumentare le dimensioni delle immagini e implementare nuovamente l'utilizzo del colore. Le modifiche non sono state comunque sufficienti per avere un'esperienza di utilizzo solida causando:

- *Overflow* della *V-RAM*^[gl] se non utilizzato sotto specifiche condizioni.
- Inconsistenza nei *cluster*.

Viene sotto riportata un'immagine in bianco e nero di dimensioni 256*256 ottenuta dal *bottleneck* dell'*autoencoder* per comparare con il modello precedente (Fig. 7.6).

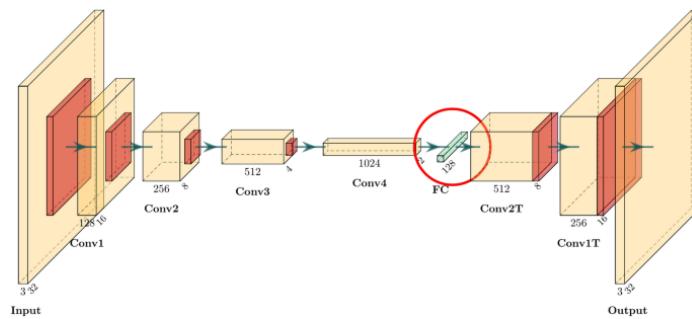


Figura 7.5: Autoencoder con strati convoluzionali

Un altro vantaggio dell'utilizzo di strati convoluzionali è la possibilità di visualizzare le *feature map* estratte da ciascun *layer* (Fig. 7.7, Fig. 7.8, Fig. 7.9). Grazie a esse lo sperimentatore ha un'idea più chiara di quello che avviene all'interno dell'*autoencoder*.



Figura 7.6: Ricostruzione immagine ottenuta dal bottleneck dell'autoencoder convoluzionale con 500 epoche di addestramento

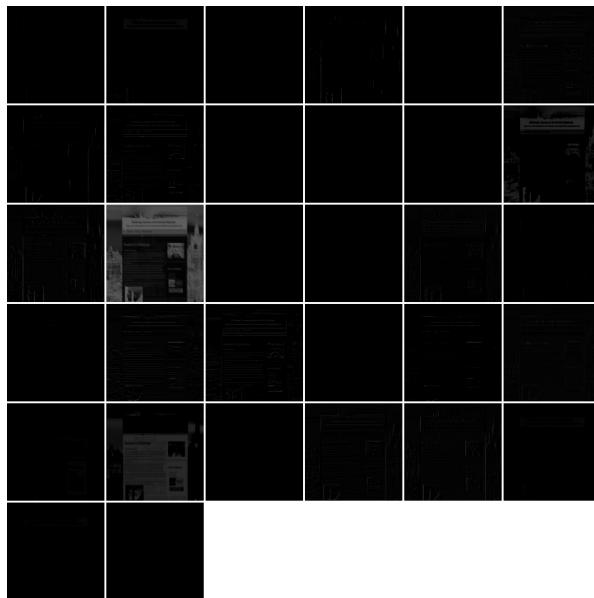


Figura 7.7: Feature map estratte dal primo strato convoluzionale

Tramite l'utilizzo delle *feature* estratte dal *bottleneck* è possibile effettuare un *clustering* (Fig. 7.10)

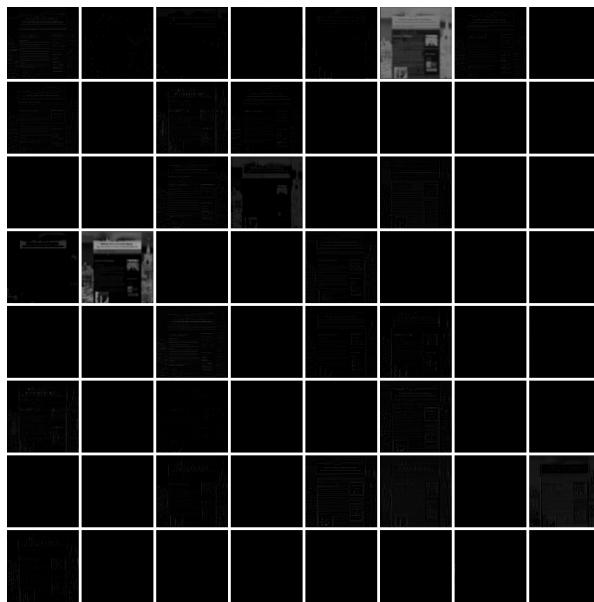


Figura 7.8: Feature map estratte dal secondo strato convoluzionale

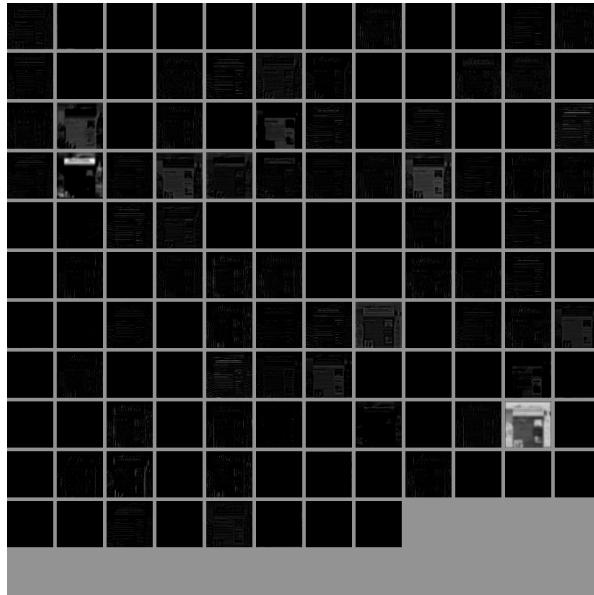


Figura 7.9: Feature map estratte dal terzo strato convoluzionale

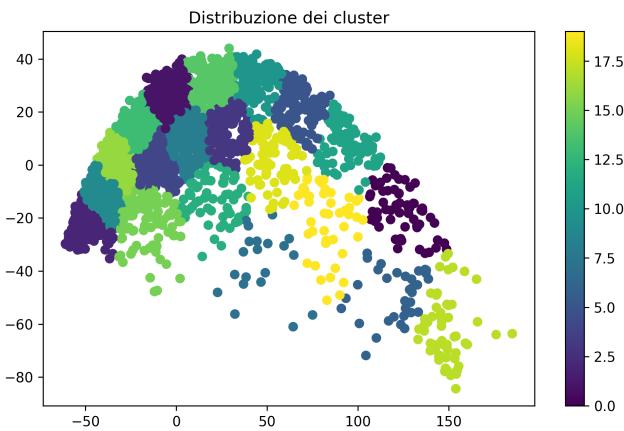


Figura 7.10: Clustering effettuato utilizzando le `feature` estratte dall’autoencoder convoluzionale; 20 cluster e 3000 immagini

7.3.2 Modelli pre-addestrati

L’utilizzo di un modello preaddestrato consente di evitare la fase di addestramento e la complessa ingegnerizzazione di una rete neurale, per questo motivo l’utilizzo degli *autoencoder* è stato scartato. Il modello consigliato dal tutor è ResNet50, la scelta viene avvalorata dalle motivazioni riportate nell’articolo¹:

- Facilità di addestramento.
- Bilanciamento tra performance ed efficienza rispetto ad altri modelli della stessa famiglia.
- Risultati simili o migliori utilizzando meno risorse rispetto alla famiglia VGGNet.

7.4 Clustering

7.4.1 Preparazione delle immagini

In seguito alla cattura le immagini vengono lette dal database e convertite in un formato utilizzabile dagli algoritmi di machine learning (Fig. 7.11). Per velocizzare le operazioni in fase di addestramento tutte le immagini vengono scaricate dal database e convertite in formato PNG, poiché le librerie utilizzate necessitano di dati *rester* per funzionare, successivamente le immagini salvate nella memoria del PC vengono processate in maniera tale da essere adatte all’utilizzo da parte di ResNet50. In questo caso le dimensioni dei tensori richieste specificatamente dal modello (Fig. 7.12) sono le seguenti (224*224*3):

- Altezza dell’immagine
- Larghezza dell’immagine

¹ Understanding ResNet: A Milestone in Deep Learning and Image Recognition. URL: <https://www.ikomia.ai/blog/mastering-resnet-deep-learning-image-recognition>.

- Numero di canali (Red Green Blue)

Per rientrare nel formato richiesto le immagini subiscono un processo di *scaling* che ne riduce la dimensionialità lasciando il contenuto invariato. Tale operazione è stata preferita al *cropping* perché utilizzandolo andrebbero a perdere caratteristiche specifiche; ad esempio i margini bianchi ai lati del sito sono spesso segnale di un design un po'antiquato e migliorabile, la rimozione dei margini potrebbe compromettere il *clustering* impedendo la creazione di un *cluster* contenente pagine di quel tipo. Inoltre ogni cella contiene valori compresi tra 0 e 255, per facilitare le operazioni tali valori vengono divisi per 255 e normalizzati in un intervallo compreso tra 0 e 1.

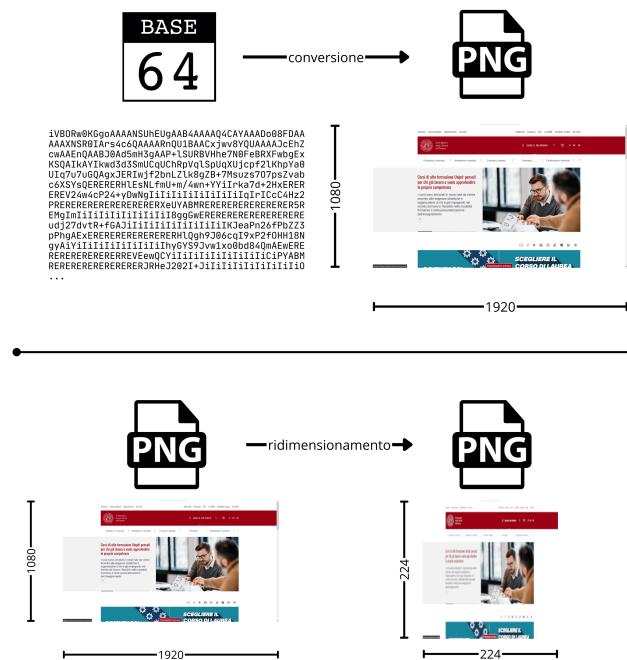
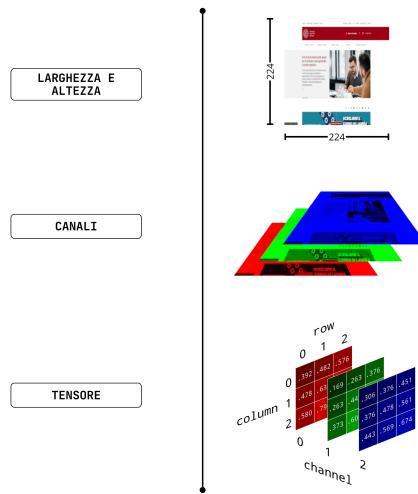
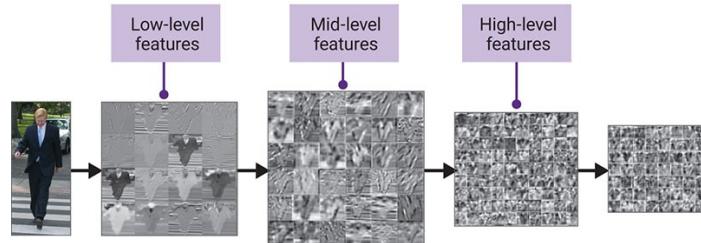
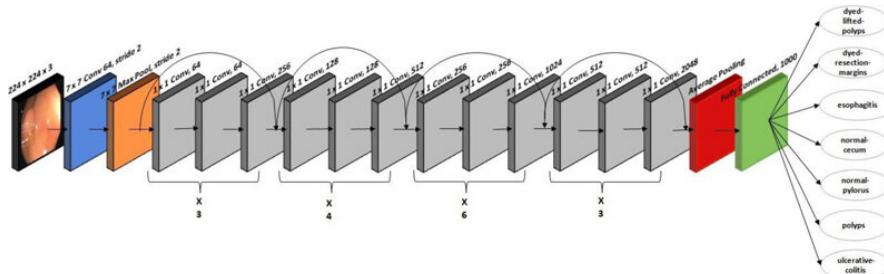


Figura 7.11: Conversione e ridimensionamento

**Figura 7.12:** Struttura del tensore

7.4.2 Estrazione delle feature

Per ottenere un dato utilizzabile in modo efficace dall'algoritmo di *clustering* è necessario modificare ulteriormente le immagini riducendole a un set di *feature* (Fig. 7.13). Per adempire a questo compito viene utilizzato un CNN Convolutional Neural Network pre-addestrato chiamato ResNet50 (Fig. 7.14). Il modello viene caricato senza i top *layer*, ossia gli strati densi addetti alla classificazione, per sfruttare esclusivamente la sua abilità di riduzione in *feature*.

**Figura 7.13:** Esempio di feature maps**Figura 7.14:** Struttura di ResNet50

7.4.3 Applicazione del clustering

Le *feature* estratte dai *layer* convoluzionali vengono ridotte a uno stato bidimensionale sfruttando l'analisi dei componenti principali, in questa maniera è possibile visualizzare i risultati del *clustering* su un grafico cartesiano, ridurre i tempi di calcolo e aumentare l'accuratezza delle predizioni; ma al costo di una ovvia perdita di informazioni. Successivamente viene applicato l'algoritmo dei K-means per l'effettiva suddivisione del *dataset* in *cluster* (Fig. 7.15). Le immagini vengono inserite, in base al *cluster* a loro assegnato, nelle rispettive cartelle di appartenenza. Questa operazione viene svolta per semplificare lo step successivo in cui lo sviluppatore deve preparare manualmente il *dataset* per l'addestramento supervisionato.

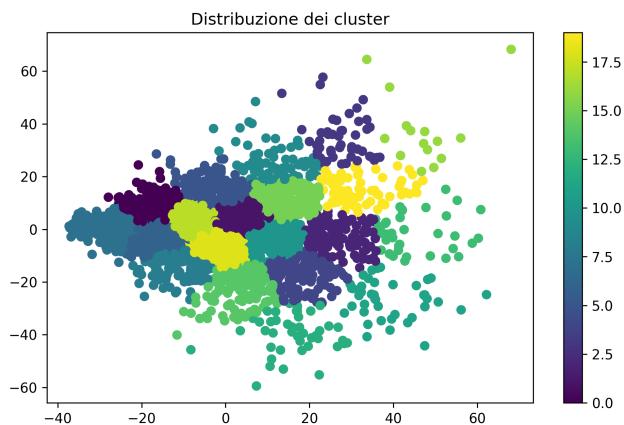


Figura 7.15: Clustering effettuato utilizzando ResNet50; 20 cluster e 3000 immagini

7.5 Valutazione

7.5.1 Preparazione del dataset

Il programmatore visiona i *cluster* ottenuti precedentemente e verifica quali possono appartenere alle categorie "siti migliorabili" e "siti ottimi"; prepara due cartelle corrispondenti alle categorie e inserisce le immagini che reputa appartenere a ciascun dominio. Lo script prepara il *dataset* (Fig. 7.16) secondo la logica seguente:

- Training, il 70% delle immagini viene utilizzato per l'addestramento effettivo.
- Validation, il 15% delle immagini viene utilizzato per ottimizzare i parametri del modello.
- Test, il 15% delle immagini serve per valutare l'*output* del modello

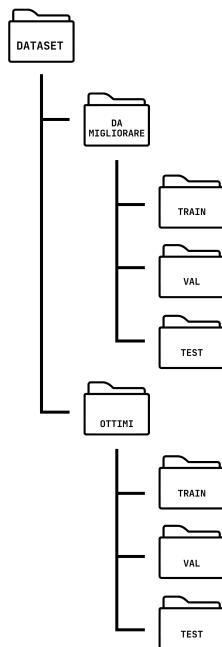


Figura 7.16: Struttura del dataset

7.5.2 Addestramento del modello

Viene caricato il modello pre-addestrato ResNet50 escludendo i *top layer*, congelando i pesi e aggiungendo degli strati personalizzati per la classificazione. Gli strati di classificazione sono successivamente addestrati utilizzando come *input* le *feature* ricavate dai *layer* convoluzionali congelati di ResNet50.

7.5.3 Fine-tuning del modello

I *layer* convoluzionali vengono sbloccati e il modello viene addestrato nuovamente nella sua interezza utilizzando un learning rate ridotto in maniera tale da non modificare

completamente i pesi del modello preaddestrato. Il modello viene salvato e riutilizzato ogniqualvolta sia necessario.

7.5.4 Valutazione delle immagini

Le immagini presenti nel database vengono classificate dal modello e le valutazioni in centesimi vengono restituite.

7.6 Invio e-mail

Le valutazioni vengono lette dal database e tramite Laravel si procede all'invio di mail personalizzate alle aziende che dispongono di siti web che potrebbero essere ancora migliorati.

7.7 Database

Il database di SalesCRM contiene molte tabelle ma in questa sezione vengono descritte solo quelle utilizzate dal *workflow*.

7.7.1 Domains

Questa tabella contiene tutte i siti web (Fig. 7.17) delle aziende collezionate dal *web scraper* e le informazioni a essi correlate.

#	Nome	Tipo	Codifica caratteri	Atributi	Null	Predefinito	Commenti	Extra	Azione
1	<code>id</code>	bigint(20)		UNSIGNED	No	Nessuno		AUTO_INCREMENT	Modifica Elimina Più
2	<code>place_id</code>	bigint(20)		UNSIGNED	Sì	NULL			Modifica Elimina Più
3	<code>url</code>	longtext	utf8mb4_unicode_ci		No	Nessuno			Modifica Elimina Più
4	<code>is_scanned</code>	tinyint(1)			No	0			Modifica Elimina Più
5	<code>is_duplicate_free</code>	tinyint(1)			Sì	0	Flag means duplicate free (pages) domain		Modifica Elimina Più
6	<code>error</code>	longtext	utf8mb4_unicode_ci		Sì	NULL			Modifica Elimina Più
7	<code>scan_time_ms</code>	int(11)			Sì	NULL	Time taken for a complete scan in milliseconds		Modifica Elimina Più
8	<code>rating_scan_manual</code>	int(11)			Sì	0	Rating manuale in centesimi		Modifica Elimina Più
9	<code>rating_scan</code>	int(11)			Sì	0	Rating su base AI del sito web in centesimi		Modifica Elimina Più
10	<code>num_pages_scanned</code>	bigint(20)		UNSIGNED	Sì	NULL			Modifica Elimina Più
11	<code>created_at</code>	timestamp			Sì	NULL			Modifica Elimina Più
12	<code>updated_at</code>	timestamp			Sì	NULL			Modifica Elimina Più
13	<code>deleted_at</code>	timestamp			Sì	NULL			Modifica Elimina Più

Azione	Chiave	Tipo	Unica	Compresso	Colonna	Cardinalità	Codifica caratteri	Null	Commenti
Modifica Elimina	PRIMARY	BTREE	Sì	No	id	0	A	No	
Modifica Elimina	domains_place_id_foreign	BTREE	No	No	place_id	0	A	Sì	

Figura 7.17: Schema della tabella Domains

7.7.2 Pages

Contiene i link a tutte le pagine (Fig. 7.18) di ciascun dominio.

The screenshot shows the phpMyAdmin interface for the 'pages' table. The top navigation bar includes tabs for Mostra, Struttura, SQL, Cerca, Inserisci, Esporta, Importa, Privilegi, Operazioni, and Trigger. The 'Struttura della tabella' tab is selected. Below the tabs is a toolbar with buttons for Stampa, Proprieta della struttura della tabella, Muovi campi, Normalizza, Aggiungi, and Esegui. The main area displays the table structure with 15 columns:

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
1	id	bigint(20)		UNSIGNED	No	Nessuno		AUTO_INCREMENT	Modifica Elimina
2	domain_id	bigint(20)		UNSIGNED	No	Nessuno			Modifica Elimina
3	url	longtext	utf8mb4_unicode_ci		No	Nessuno			Modifica Elimina
4	html_content	longtext	utf8mb4_unicode_ci		Si	NULL			Modifica Elimina
5	email	longtext	utf8mb4_unicode_ci		Si	NULL			Modifica Elimina
6	phone	longtext	utf8mb4_unicode_ci		Si	NULL			Modifica Elimina
7	address	longtext	utf8mb4_unicode_ci		Si	NULL			Modifica Elimina
8	is_checked	tinyint(1)			No	0			Modifica Elimina
9	is_downloaded	tinyint(1)			No	0	Update it when downloaded image		Modifica Elimina
10	checked_at	timestamp			Si	NULL			Modifica Elimina
11	check_info	longtext	utf8mb4_unicode_ci		Si	NULL			Modifica Elimina
12	created_at	timestamp			Si	NULL			Modifica Elimina
13	updated_at	timestamp			Si	NULL			Modifica Elimina
14	deleted_at	timestamp			Si	NULL			Modifica Elimina
15	screenshot	longtext	utf8mb4_unicode_ci		Si	NULL	Base64 image of PNG screenshot		Modifica Elimina

Below the table structure, there are buttons for Selezione tutto, Se selezionati: Mostra, Modifica, Elimina, Primaria, Unica, Indice, Spaziale, and Testo completo. The 'Indici' section shows two indexes:

Azione	Chiave	Tipo	Unica	Compresso	Colonna	Cardinalità	Codifica caratteri	Null	Commenti
Modifica Rimozione	PRIMARY	BTREE	Si	No	id	0	A	No	
Modifica Rimozione	pages_domain_id_foreign	BTREE	No	No	domain_id	0	A	No	

At the bottom, there is a button for Crea un indice su 1 campi Esegui.

Figura 7.18: Schema della tabella Pages

Capitolo 8

Conclusioni

8.1 Raggiungimento degli obiettivi

Per terminare il percorso di stage è necessario verificare il grado di soddisfacimento degli obiettivi preimposti, tale processo viene rappresentato nella tabella 8.1. L'implementazione del sistema di cattura rispecchia completamente le aspettative e funziona in maniera corretta, integrandosi inoltre con il database preesistente. I requisiti desiderabili e facoltativi, come l'inserimento e l'aggiornamento della valutazione dei siti web, sono risultati una naturale estensione del processo di valutazione tramite IA e quindi sono stati implementati come se facessero parte del medesimo obiettivo. L'automazione dell'invio delle e-mail è stata preparata e testata, ma per motivi di sicurezza necessita ancora dell'approvazione di un supervisore prima di lanciare la campagna di invio. Nelle sotto sezioni successive è presente un'analisi più dettagliata dei processi fondamentali del processo.

Codice	Descrizione	Soddisfacimento
O01	Implementare un sistema robusto per la cattura degli screenshot, assicurando l'integrazione con il database per l'archiviazione e l'analisi.	Soddisfatto
O02	Garantire la creazione di una documentazione tecnica completa che supporti sia l'uso che la manutenzione del sistema sviluppato.	Soddisfatto
O03	Creazione di un IA in grado di suddividere in cluster i siti web.	Soddisfatto
O04	Creazione di un IA classificativa in grado di assegnare un punteggio ai siti web analizzati.	Soddisfatto
D01	Aggiungere la valutazione creata dall'IA nel database utilizzato dal CRM.	Soddisfatto
D02	Automatizzare l'invio di email alle aziende che hanno ottenuto una valutazione scarsa.	Soddisfatto
F01	Migliorare la raccolta dei dati delle aziende dal web	Non soddisfatto
F02	Aggiornare automaticamente la valutazione dei siti web	Soddisfatto

Tabella 8.1: Grado di soddisfacimento degli obiettivi

8.1.1 Clustering

I risultati di *clustering* migliori sono stati ottenuti con l'estrazione delle *feature* tramite ResNet50, ma il processo necessita di ulteriori ottimizzazioni per essere funzionale; infatti visualizzando manualmente i cluster ottenuti non è possibile notare delle distinzioni nette tra immagini appartenenti a una o all'altra categoria. Ciò potrebbe essere imputabile alla natura del *dataset*, le immagini di siti web non sono dei soggetti semplici da analizzare, poiché ricche di informazioni come immagini relative al contesto, che non sono utili per giudicare la qualità. Ad esempio prendiamo in considerazione il sito web dell'università e il sito di un centro estetico che offre servizi di manicure (Fig. 8.1). Le due immagini contengono rappresentazioni di mani, di conseguenza potrebbero essere inserite all'interno dello stesso cluster nonostante facciano parte di contesti diametralmente opposti.

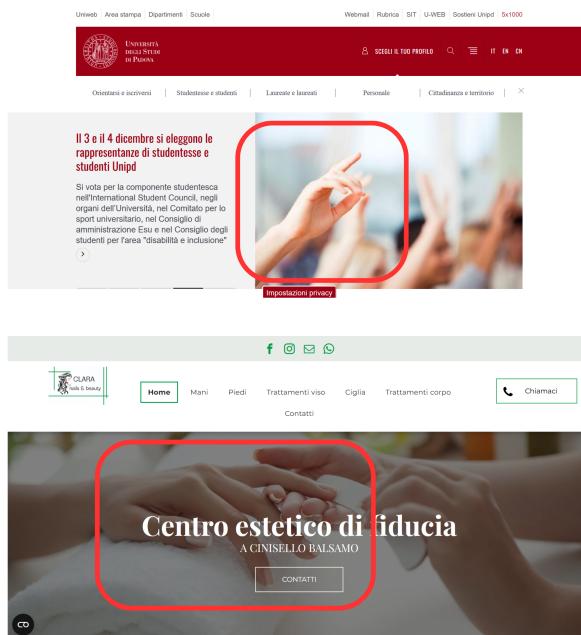


Figura 8.1: Immagini appartenenti a contesti differenti

Un'altra combinazione è dettata da appartenenza alla medesima categoria e differenza di qualità (Fig. 8.2). In questo caso la presenza dei computer potrebbe comportare l'inserimento nello stesso cluster, ma come è possibile notare, ciò non sarebbe utile ai fini del progetto, perché si unirebbe un sito web di buona qualità a uno che utilizza tecnologie molto arretrate.

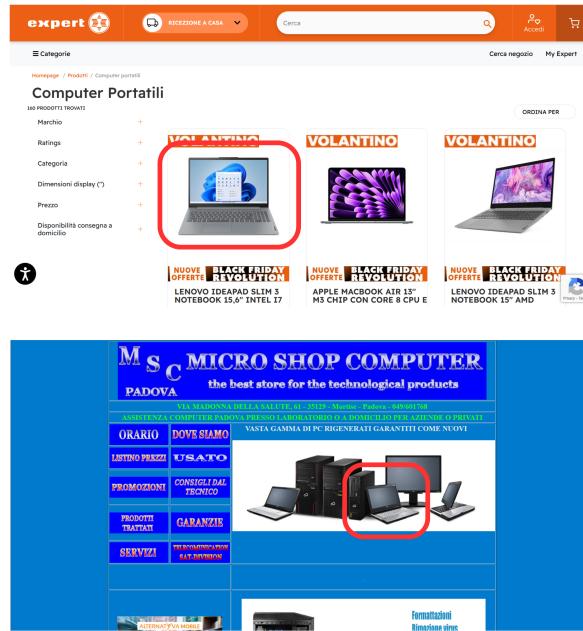


Figura 8.2: Immagini appartenenti allo stesso contesto

Per migliorare il *clustering* ci si potrebbe concentrare solo su parti specifiche del sito web rendendo l'analisi più indipendente da informazioni non utili; inoltre si potrebbe rivalutare l'utilizzo della PCA, effettuando uno studio sulla dimensionalità ottimale dello spazio delle *feature*. (Tale scelta comporta un probabile miglioramento dei risultati ma compromette la facile interpretabilità dei grafici). Per procedere con il progetto risulta anche necessario comprendere più a fondo come utilizzare l'algoritmo K-means al meglio e testare altri algoritmi di *clustering*. Il lavoro svolto può essere considerato un'ottima base di partenza poiché si è concentrato sullo studio preliminare di varie metodologie; arrivando a escludere quelle meno adatte al compito di estrazione delle *feature*.

8.1.2 Valutazione delle immagini

Per quanto riguarda la fase di valutazione, i risultati ottenuti dal *fine tuning* di ResNet50 sembrano essere ottimali, ottenendo un'accuratezza del 92%. La rete neurale è inoltre risultata capace di assegnare valutazioni congrue a quelle che lo sviluppatore si aspetterebbe di visualizzare; ossia voti che vanno da 0 a 100 in un range che spazia

da siti molto antiquati e nettamente migliorabili a siti ottimi che non necessitano modifiche. Il prodotto è comunque migliorabile aggiungendo una quantità più elevata di screenshot al *dataset* e di conseguenza aumentando il range di siti a cui adattarsi.

8.2 Valutazione personale

L'esperienza di stage è stata di grande impatto, ha consentito di sfruttare le conoscenze acquisite e ha messo in risalto numerose lacune da colmare, il tutto in un contesto aziendale stimolante e improntato verso tecnologie all'avanguardia. In particolare, il progetto mi ha consentito di apprendere nuove conoscenze nell'ambito del *machine learning*, argomento che personalmente non avevo mai approfondito. Le scarse conoscenze in ambito IA hanno comportato una grossa sfida che, in seguito a un approfondito studio personale riguardante le stesse e la loro applicazione, sono riuscito a superare, ottenendo un alto livello di soddisfazione.

Glossario

API Le API sono un insieme di regole utilizzate per consentire a software diversi di comunicare tra loro.. [18](#), [36](#)

bot I bot sono dei programmi inseriti all'interno di telegram per eseguire determinate azioni in maniera automatica. [5](#)

computer vision La computer vision si occupa di sviluppare algoritmi in grado di estrarre informazioni da immagini, consentendo così al computer di "vedere". [18](#)

CRM Un software di CRM si occupa di gestire il marketing, il supporto e le vendite dell'azienda in maniera automatizzata. Più in generale è l'insieme di strategie utilizzate per migliorare e organizzare tutti i rapporti con i clienti.. [3](#), [8](#), [50](#)

dataset Un dataset è una raccolta di dati appartenenti a determinate categorie.. [ii](#), [10](#), [15](#), [24](#), [26](#), [27](#), [45](#), [46](#), [51](#), [53](#)

feature Le feature sono le caratteristiche che vengono utilizzate per addestrare un algoritmo di machine learning.. [vii](#), [14](#), [24](#), [27](#), [32](#), [35](#), [38](#), [40](#), [42](#), [44–46](#), [51](#), [52](#)

framework Un framework è un sistema che permette di ampliare le capacità di un linguaggio di programmazione, semplificando e ottimizzando lo svolgimento di determinate azioni da parte del programmatore.. [18](#), [21](#), [22](#)

open source L'open source è una tipologia di software di libera distribuzione, che può essere utilizzato, modificato e redistribuito da chiunque a titolo gratuito.. [5](#), [23](#)

self hosted Con self hosted si intende l'esecuzione del software all'interno di server aziendali. [6](#)

UML L'unified modelling language è un linguaggio di modellazione unificato nato per semplificare la comunicazione di concetti complicati, tramite l'utilizzo di schemi e immagini codificate.. [12](#)

V-RAM È la memoria contenuta all'interno delle schede video, è una componente fondamentale per consentire alla GPU di processare informazioni. [39](#)

web app È un'applicazione utilizzabile tramite il web.. [17](#)

web scraping Il web scraping si occupa dell'estrazione di informazioni da siti web tramite strumenti automatici.. [10](#)

workflow Il termine workflow o flusso di lavoro indica la struttura che si occupa di gestire un insieme di processi da svolgere in un ordine predefinito.. [8](#), [10](#), [12](#), [13](#), [20](#), [37](#), [47](#)

Bibliografia

Riferimenti bibliografici

Goodfellow, Ian, Yoshua Bengio e Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016 (cit. alle pp. 24, 35).

Siti web consultati

Agile. URL: <https://www.atlassian.com/it/agile> (cit. a p. 4).

Conventional Commits. URL: <https://www.conventionalcommits.org/en/v1.0.0/> (cit. a p. 7).

Cos'è il k-means clustering? URL: <https://www.ibm.com/it-it/topics/k-means-clustering>.

Cos'è l'analisi delle componenti principali (PCA)? URL: <https://www.ibm.com/it-it/topics/principal-component-analysis> (cit. a p. 27).

Cos'è una rete neurale? URL: <https://aws.amazon.com/it/what-is/neural-network/> (cit. a p. 28).

CUDA. URL: <https://developer.nvidia.com/cuda-toolkit>.

DDEV. URL: <https://ddev.com/>.

Docker. URL: <https://www.docker.com/>.

Filament. URL: <https://filamentphp.com/>.

Git. URL: <https://git-scm.com/>.

Git Flow. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow> (cit. a p. 7).

Iris dataset. URL: <https://archive.ics.uci.edu/dataset/53/iris> (cit. a p. 24).

Keras. URL: <https://keras.io/>.

Laravel. URL: <https://laravel.com/>.

Machine learning, deep learning e reti neurali. URL: <https://www.ibm.com/it-it/topics/machine-learning> (cit. a p. 24).

Manifesto Agile. URL: <http://agilemanifesto.org/iso/it/>.

Matplotlib. URL: <https://matplotlib.org/>.

MySQL. URL: <https://www.mysql.com/it/>.

Numpy. URL: <https://numpy.org/>.

OpenCV. URL: <https://opencv.org/get-started/>.

phpMyAdmin. URL: <https://www.phpmyadmin.net/>.

Plane. URL: <https://plane.so/>.

Python. URL: <https://www.python.org/>.

RelAI. URL: <https://tessere.spaziodev.eu/> (cit. a p. 3).

Remote Ripple. URL: <https://remoteripple.com/>.

Scikit-learn. URL: <https://scikit-learn.org/stable/index.html>.

Spazio Dev. URL: <https://spaziodev.eu/> (cit. a p. 2).

Telegram. URL: <https://telegram.org/>.

Tensorflow. URL: <https://www.tensorflow.org/?hl=it>.

Understanding ResNet: A Milestone in Deep Learning and Image Recognition. URL: <https://www.ikomia.ai/blog/mastering-resnet-deep-learning-image-recognition> (cit. a p. 42).

Visual Studio Code. URL: <https://code.visualstudio.com/>.

WSL. URL: <https://learn.microsoft.com/it-it/windows/wsl/install>.