



UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA IN INFORMATICA

**STIMA DELLA PRESSIONE SANGUIGNA
DA VIDEO MEDIANTE IL FRAMEWORK
FACEQS**

RELATORE: Prof. Giuliano Grossi

CORRELATORE: Prof. Raffaella Lanzarotti

TESI DI LAUREA DI: Tommaso Amadori

MATRICOLA: 963792

ANNO ACCADEMICO 2021-22

Indice

Introduzione	1
1 La misurazione fisiologica digitale	2
1.1 Medicina telematica	2
1.2 La pressione sanguigna	4
1.2.1 Tecniche di misurazione	5
1.2.2 Pressione sistolica, diastolica e arteriosa	6
1.3 Stato dell'arte	7
2 FaceQs	9
2.1 Struttura Interna	10
2.2 Utilizzi del Framework	12
2.2.1 Eseguire un analisi	13
2.2.2 Personalizzare le analisi	15
2.3 Estensioni di FaceQs	16
3 Misurazione della pressione sanguigna da remoto	17
3.1 pyVHR	19
3.1.1 Estrazione del segnale rPPG	19
3.2 NeuroKit2 come analizzatore del segnale rPPG	21
3.3 Feature selection	24
3.4 Modelli di machine learning	26
3.4.1 Approccio supervisionato	26
3.4.2 Approccio non supervisionato	29
3.4.3 Approccio semi-supervisionato	30
3.4.4 Random Forest Regressor	31
3.4.5 Neural networks	35
3.5 Post-processing applicato al segnale stimato	38
4 Esperimenti e risultati	41
4.1 Datasets	42

4.1.1	V4V dataset	42
4.1.2	MIMIC-II dataset	42
4.1.3	VitalDB dataset	43
4.2	Applicazione dei modelli	43
4.2.1	Preparazione dei dati	43
4.2.2	Fase di training	46
4.2.3	Selezione dei modelli	46
4.2.4	Randomized search, grid search e cross-validation	47
4.3	Risultati ottenuti	50
4.3.1	Risultati tramite segnale PPG	52
4.3.2	Risultati tramite segnale rPPG	63
4.4	Discussione	72
	Conclusioni e sviluppi futuri	74

Introduzione

Negli ultimi anni la ricerca di nuovi strumenti utilizzabili nel mondo della telemedicina è cresciuta notevolmente. A partire dal 2010 è stato registrato un discreto numero di pubblicazioni nel mondo delle analisi di segnali fisiologici a partire da un semplice video. Nella stragrande maggioranza dei casi le pubblicazioni sono relative al segnale fotopletismografico ovvero uno dei principali segnali in grado di analizzare la perfusione sanguigna. In parte molto più ristretta sono stati analizzati altri segnali tra cui la pressione sanguigna. L'obiettivo di questa tesi è consistito nell'esplorare questo specifico segnale ed individuare quindi una possibile soluzione nella stima della pressione sanguigna tramite video ed infine progettare un framework python denominato *FaceQs* in cui vengono racchiuse funzionalità di stima di segnali fisiologici da remoto.

Date le ridotte ricerche pubblicate, sono state esplorate nuove strade per individuare nuovi sistemi in grado di effettuare stime più o meno precise. Le soluzioni allo stato dell'arte prevedono una molteplice serie di passaggi per stimare la pressione sanguigna (sistolica, diastolica e arteriosa). La soluzione adottata in questa tesi non è da meno: è stata definita una *pipeline* che parte dall'estrazione del segnale fotopletismografico dal video, seguita da una analisi morfologica di tale segnale fino ad arrivare ad una selezione delle caratteristiche prodotte da quest'ultima seguite in ultimo da una fase di elaborazione di tali indici tramite dei modelli di machine learning. Il segnale risultante da tali modelli viene infine processato secondo funzioni matematiche in modo tale da renderlo di facile lettura.

La tesi è così articolata: nel Capitolo 1 viene descritto il segnale della pressione sanguigna riportando quali sono le caratteristiche, l'importanza ad esso legata e quali sono i lavori allo stato dell'arte, della stima da remoto; nel Capitolo 2 viene introdotto il framework FaceQs ovvero il pacchetto python in cui è contenuta la ricerca svolta; nel Capitolo 3 viene descritta nel dettaglio la *pipeline* studiata per stimare la pressione, descrivendo ogni strumento interno e esterno utilizzato per raggiungere l'obiettivo; nel Capitolo 4 vengono descritti gli esperimenti su diversi dataset analizzando infine i risultati ottenuti con diverse configurazioni della *pipeline*; nelle conclusioni, infine, vengono riassunti i risultati raggiunti e viene proposto un possibile sviluppo futuro realizzabile mediante questa tecnologia.

Capitolo 1

La misurazione fisiologica digitale

1.1 Medicina telematica

Al giorno d'oggi la nostra vita è fortemente influenzata da Internet e dalla tecnologia. Siamo costantemente circondati da sistemi che monitorano le nostre necessità e rispondono alle nostre esigenze come fanno, ad esempio, i sistemi dell'IoT. Allo stesso modo la tecnologia negli ultimi anni, soprattutto con l'arrivo della pandemia mondiale dovuta al COVID-19, sta permettendo di prenderci cura della nostra salute tramite quella che viene chiamata "telemedicina" o medicina telematica. Quando si parla di medicina telematica si parla dell'uso delle informazioni digitali per accedere, monitorare e gestire lo stato di salute del paziente. Le tecnologie utilizzate in questo ambito possono essere normali comuni computer e/o dispositivi mobili come ad esempio tablet o smartphone. Il fatto che si possano utilizzare sistemi comuni è molto importante perché permette, alla stragrande maggioranza delle persone, di potervi accedere da qualunque posto e senza troppe difficoltà (di natura economica e di accessibilità). Nello specifico caso di COVID-19 la telemedicina è stata fortemente utilizzata riscontrando un grande successo e una grande approvazione da parte della comunità che ne ha fatto uso.

Essa permette inoltre di dare supporto ai classici strumenti di monitoraggio, ad esempio quando questi si inceppano o smettono di funzionare. La "telemedicina" può quindi sostituirsi in maniera piuttosto rilevante alla classica strumentazione fornendo comunque un valido sistema di monitoraggio. Più in generale la medicina telematica presenta diversi vantaggi, tra cui:

- aumentare la disponibilità del servizio sanitario raggiungendo anche le zone più remote,
- rendere l'accesso all'assistenza sanitaria più semplice e immediato (specialmente a chi ha difficoltà di movimento),

- mantenere sé stessi e gli altri più al sicuro da malattie infettive, come ad esempio il COVID-19,
- migliorare la comunicazione e coordinazione tra persone e team medici che si prendono cura del paziente,

Come ogni soluzione adottabile, però, non si hanno mai solo ed esclusivamente vantaggi, bensì sono presenti diversi svantaggi, tra cui:

- l'assistenza fornita dalla telemedicina non sarà mai in grado di sostituire interamente la medicina fisica,
- una visita telematica richiede più tempo per una analisi precisa,
- rischio di malasanità durante la prestazione dell'assistenza medica dovuti a problemi tecnici, ad esempio errori di connessione o violazione delle videoconferenze e quindi violazione della privacy.

La telemedicina si divide in diverse categorie (1), tra cui:

- Tele-nutrizione: costituita dalle conferenze con i nutrizionisti o dottori che guidano il paziente nella corretta alimentazione a seconda delle proprie necessità.
- Tele-infermieristica: realizzata mediante l'uso delle telecomunicazioni per fornire servizi infermieristici tra cui la tele-diagnosi, tele-consultazione, ed altri.
- Tele-farmacia: fornitura di cure farmaceutiche quando non si ha la possibilità di un diretto contatto con il farmacista. Tale cura include la prescrizione di farmaci e il monitoraggio delle assunzioni di tale medicine.
- Tele-neurologia: branca della telemedicina che si occupa di fornire cure per ictus, disturbi del movimento (come ad esempio il morbo di Parkinson), crisi epilettiche ecc.
- Tele-neuropsicologia: sfrutta l'uso delle videoconferenze per valutare e, se necessario curare, disturbi psichici.
- Tele-riabilitazione: parte della telemedicina che si occupa di guidare i pazienti in diversi tipi di riabilitazione tra cui la riabilitazione neuropsicologica, logopedica, occupazionale e fisica.
- Tele-cardiologia: rappresenta la telemedicina più antica nella quale si prevede di trasmettere i dati rilevati dai sensori ECG tramite un sistema wireless.

Parte dello studio condotto durante questa tesi è strettamente legato alla telemedicina in quanto si concentra in modo particolare sulla misurazione della pressione sanguigna a partire unicamente dal segnale video. Il video può essere quello prodotto da una qualunque telecamera, più o meno performante. Questo significa quindi che una comune webcam montata sopra il proprio laptop o un dispositivo mobile quale tablet e smartphone è sufficiente per effettuare tale misurazione. Gli unici vincoli richiesti da tale metodo è che nel video sia ben raffigurato il volto con la dovuta illuminazione. Nel prossimo capitolo verrà esplicata la metodologia per risalire, tramite una sequenza di immagini, alla pressione sanguigna del soggetto raffigurato.

1.2 La pressione sanguigna

La pressione sanguigna (in inglese chiamata *blood pressure* o BP) è la forza esercitata dal sangue circolante sulle pareti dei vasi sanguigni. Essa rappresenta un indice molto importante in quanto è altamente correlato con la salute. Un'alta pressione del sangue (anche chiamata ipertensione), se non controllata, può portare a gravi danni legati al cuore e all'indurimento delle arterie, il quale causa a sua volta una riduzione del flusso di sangue e di ossigeno (2). Una pressione alta, assieme ad un ridotto flusso sanguigno, possono provocare uno o più dei seguenti problemi:

- angina, ovvero un dolore transitorio al petto dovuto al fatto che il cuore non riceve una sufficiente quantità di ossigeno (3),
- infarto, ovvero interruzione improvvisa dell'afflusso di sangue al cuore. Questa interruzione causa la morte delle cellule del cuore per mancanza di ossigeno, inoltre, maggiore è il tempo in cui il flusso sanguigno resta bloccato, maggiore è il danno al cuore,
- arresto cardiaco, ovvero quando il cuore non riesce a pompare abbastanza sangue e ossigeno per il corretto funzionamento degli altri organi,
- battito cardiaco irregolare che può portare a una morte improvvisa.

Analogamente, quando una pressione eccessivamente bassa può provocare altrettanti problemi ma decisamente meno pericolosi di quelli che mettono a rischio i soggetti che soffrono di ipertensione. In questo caso si parla di ipotensione. I problemi principali per cui è a rischio un soggetto che soffre di pressione bassa sono i seguenti:

- vertigini o stordimento,
- svenimento o sincope (forse il sintomo più pericoloso per soggetti di età più avanzata),

- confusione o difficoltà a concentrarsi,
- stanchezza,
- nausea,
- visione sfocata o distorta,
- frequenza respiratoria altalenante,
- agitazione o cambiamenti insoliti nel comportamento.

1.2.1 Tecniche di misurazione

La pressione viene misurata in diversi modi e con diversi strumenti. Il più invasivo è il cateterismo arterioso. Usato esclusivamente in ambiente ospedaliero in contesti di sala operatoria e terapia intensiva, consente di misurare il valore pressorio endovasale con alto grado di precisione. Meno invasiva è la rilevazione della pressione arteriosa mediante sfigmomanometro. È quella più utilizzata e applicabile pressoché in ogni situazione, compresa l'automisurazione a domicilio. Permette di valutare, con buona approssimazione, la pressione a livello del cuore.

Il monitoraggio dinamico della pressione arteriosa (ABPM, più comunemente ma impropriamente detto anche "Holter pressorio") è una metodica diagnostica basata su uno strumento portatile in grado di monitorare la pressione del cuore per 24 o più ore. L'holter pressorio esegue una misurazione ad intervalli (tipicamente di 15 o 30 minuti). Esso trova applicazione in diverse condizioni cliniche, ad esempio per verificare che l'andamento della pressione arteriosa segua il normale ritmo "sonno-veglia", ovvero un orologio biologico della durata di circa 24 ore, che regola l'alternanza tra sonno e veglia che si ripete ciclicamente (4).

Quelli sopra riportati sono gli strumenti principali e più utilizzati al giorno d'oggi. Con l'avanzare della tecnologia in campo informatico e medico stanno nascendo nuove soluzioni per rispondere a diversi problemi di natura medica. In questa tesi viene ricercato, studiato e sviluppato un metodo alternativo per misurare la pressione sanguigna tramite un normale video generato da una comune webcam. Questa soluzione, oltre a portare un discreto contributo nella ricerca, presenta parte dei vantaggi presentati in 1.1. Alcuni di questi, sono, ad esempio, il fatto che per misurare la pressione non sarà necessario l'uso di nessun dispositivo ad hoc e che sono così esclusi errori di misurazione dovuti ad un uso errato dello strumento di misurazione. Inoltre, la possibilità di misurare la pressione tramite un semplice video, rappresenta una soluzione ancor meno invasiva di quanto fatto tramite uno sfigmomanometro. Trova utilità anche in un controllo di massa che può essere effettuato in ambienti ospedalieri

dove non è necessaria la presenza di un assistente sanitario che prepara la strumentazione per misurare la pressione, bensì è sufficiente posizionare il paziente di fronte ad una telecamera in ambiente illuminato riducendo così il lavoro dell'assistente e le tempistiche di misurazione.

1.2.2 Pressione sistolica, diastolica e arteriosa

Quando si parla di misurazione della pressione del sangue si sente sempre parlare di massima e minima, valori che corrispondono rispettivamente alla pressione sistolica e diastolica (5). Nello specifico, si parla di pressione sistolica, in inglese Systolic Blood Pressure (SBP), quando il cuore si contrae e spinge il sangue lungo le arterie che portano sangue e ossigeno agli organi del corpo. Tale pressione dipende quindi dall'efficienza della pompa cardiaca (quantità di sangue espulso ad ogni contrazione) e dall'elasticità delle pareti delle arterie. Quando il lume delle arterie si restringe o diminuisce l'elasticità delle pareti il sangue incontra maggiori difficoltà a scorrere e la pressione massima aumenta oltre i valori normali.

Con pressione diastolica (quindi pressione minima), in inglese chiamata Diastolic Blood Pressure (DBP), si intende il momento in cui, durante un ciclo cardiaco, termina lo svuotamento del cuore e inizia così la fase di riempimento (diastole). In questo periodo il flusso del sangue nelle arterie diminuisce così come la pressione che raggiunge il suo valore minimo un attimo prima dell'inizio della nuova sistole. La pressione arteriosa minima dipende quindi dalla resistenza che il sangue incontra nei tessuti periferici. Tanto più il flusso viene ostacolato e tanto più lentamente la pressione scende. In condizioni normali la pressione sistolica (o massima) è pari a 120 mmHg mentre la pressione diastolica (o minima) è pari a 80 mmHg. In figura 1.1 la fascia verde mostra un intero ciclo cardiaco dove sono evidenziati, tramite croci rosse, i picchi diastolici e sistolici.

Oltre alle due misure sopra citate, ne esiste una terza denominata pressione arteriosa, in inglese Mean Arterial Pressure (MAP), che rappresenta, rispetto all'indice di pressione sistolica, un indicatore più significativo per determinare un adeguata perfusione (6). Analogamente agli indici SBP e DBP esiste anche in questo caso un intervallo significativo di una buona salute che è compreso tra i 70 e i 100 mmHg.

Per determinare il valore della MAP esistono vari metodi. Quello che fornisce una maggiore precisione è ottenuto tramite l'uso di un dispositivo invasivo ad hoc, costituito da un ago inserito all'interno dell'arteria. Altri metodi, meno precisi ma comunque affidabili, sono soluzioni matematiche che sfruttano i valori SBP e DBP. La formula utilizzata varia a seconda dell'età, o del battito cardiaco del soggetto. In

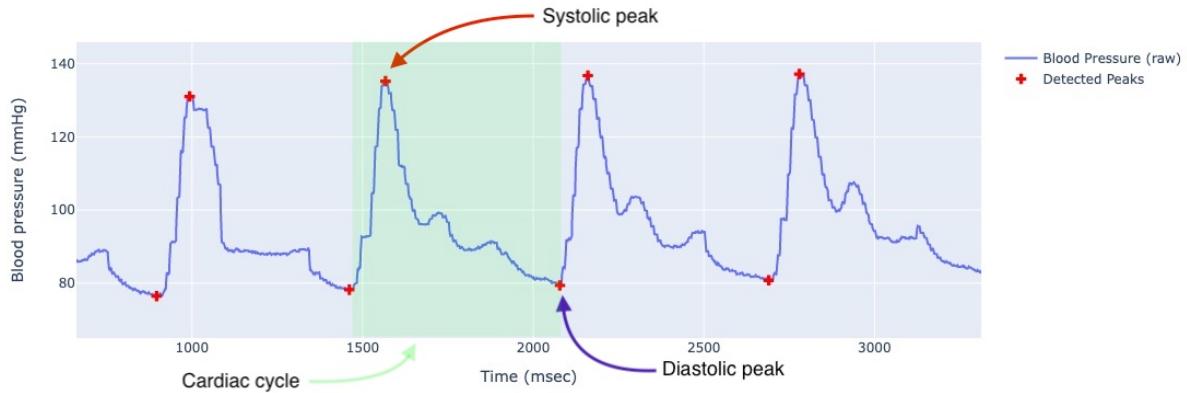


Figura 1.1: Esempio di segnale di pressione sanguigna. Le croci rosse sono posizionate in corrispondenza dei picchi sistolici e diastolici. La fascia verde rappresenta un ciclo cardiaco completo.

generale, però, vale la seguente formula:

$$MAP = DBP + 1/3(SBP - DBP)$$

dove DBP rappresenta la pressione diastolica, SBP la pressione sistolica e MAP la pressione arteriosa.

1.3 Stato dell'arte

Con l'avanzare della tecnologia e della ricerca, gli obiettivi si fanno sempre più complessi e si aprono sfide sempre più ambiziose. In questo studio, come già scritto sopra, l'obiettivo è quello di rendere possibile la misurazione della pressione sanguigna utilizzando esclusivamente un video effettuabile tramite una videocamera che sia di un laptop o di un dispositivo mobile come ad esempio smartphone o tablet. Questo argomento è già stato affrontato da diversi ricercatori e noi non saremo i primi e nemmeno gli ultimi a proporre una soluzione. Proprio per questo di seguito vengono riportati i lavori più seguiti e rilevanti dello stato dell'arte corrente.

Relativamente alla pressione sanguigna le soluzioni che partono direttamente da un video sono in numero ridotto in quanto è più facile trovare lavori e ricerche di predizione che partono dal segnale fotopletismografico, in inglese *photoplethysmography* (PPG).

È stato mostrato che lo studio morfologico del segnale PPG presenta diverse informazioni fortemente correlate con la pressione sanguigna. Per questo motivo tali informazioni vengono usate nella maggior parte degli studi per effettuare predizioni/misurazioni della pressione sanguigna. Un esempio di ricerca che ha fornito risultati significativi mostra come una rete allenata su un segnale PPG (quindi estrapolato

direttamente da un fotopletismogramma) e poi ottimizzata su segnali rPPG (quindi un segnale fotopletismografico estratto dalla videocamera) producono una predizione della pressione sanguigna molto affidabile (7).

Altri ricercatori hanno mostrato nei loro lavori che partendo dal segnale rPPG, estratto dal volto oppure dal palmo della mano, si possono estrarre informazioni come ad esempio il Pulse Transmit Time (PTT), che mostra un'interessante correlazione con la pressione sanguigna. Da qui è quindi possibile applicare il classico machine learning o una rete neurale per produrre stime della pressione interessanti (8) (9). Il problema del PAT così come il PTT è che è strettamente legato al soggetto di conseguenza non si adatta in generale ed è per questo che tale soluzione è stata superata.

In ogni caso i lavori sopra riportati utilizzano dati di piccole dimensioni. Un lavoro di dimensioni decisamente considerevoli è stato svolto da un gruppo di ricercatori che hanno coinvolto 1328 soggetti normo-tensivi. Anche loro, come gli altri ricercatori, sono partiti da un video prodotto tramite uno smartphone da cui hanno poi estratto il segnale rPPG ed infine, tramite avanzati sistemi di machine learning hanno estratto la pressione sistolica, diastolica e arteriosa (10).

Capitolo 2

FaceQs

FaceQs è un framework sviluppato presso il *Perceptual Computing and Human Sensing Lab* (PhuSe Lab) ed è un progetto che nasce dalla necessità di rendere fruibile a tutti, con comodità e semplicità, una serie di strumenti utili nel mondo delle analisi di caratteristiche fisiologiche umane.

Il nome deriva proprio dal termine inglese *face cues*, ovvero segnali facciali. La pronuncia inglese del termine *cues* è stata poi "abbreviata" in *qs* dove la q ricorda proprio il suono del termine *cue*.

FaceQs nasce quindi con l'intenzione di racchiudere diverse analisi effettuabili tramite un video in cui è ben raffigurato un volto in buone condizioni di luce. Si può dire che FaceQs nasce a partire da un progetto già esistente e funzionante il cui nome è pyVHR. Quest'ultimo è un framework sviluppato anch'esso presso PHuSe Lab ma con la specifica funzione di analizzare, sempre tramite video, solamente il segnale fotopletismografico da remoto (anche detto rPPG). Da questo specifico segnale pyVHR permette di definire la frequenza cardiaca del soggetto. Questa funzionalità ha fatto crescere l'interesse all'interno del laboratorio circa la possibilità di sviluppare un framework non solo dedicato al battito cardiaco, bensì ad una serie di altre funzionalità legate all'analisi dei principali segnali fisiologici e non solo. Alcuni di questi segnali fisiologici riguardano la pressione sanguigna, la frequenza respiratoria, il segnale eletromiografico ed altri. Inoltre, tramite FaceQs, si è pensato di aggiungere altri tipi di analisi come ad esempio l'analisi dell'audio, oppure del *gaze* (ovvero dello sguardo) per estrapolare informazioni di vario genere come ad esempio l'umore del soggetto.

Tramite FaceQs è anche possibile estrarre il volto da un video e con esso i vari landmark e la mesh facciale. Queste sono delle funzionalità di base che possono comunque essere utilizzate per svolgere computazioni a partire proprio dalle informazioni basilari di un video raffigurante un volto.

2.1 Struttura Interna

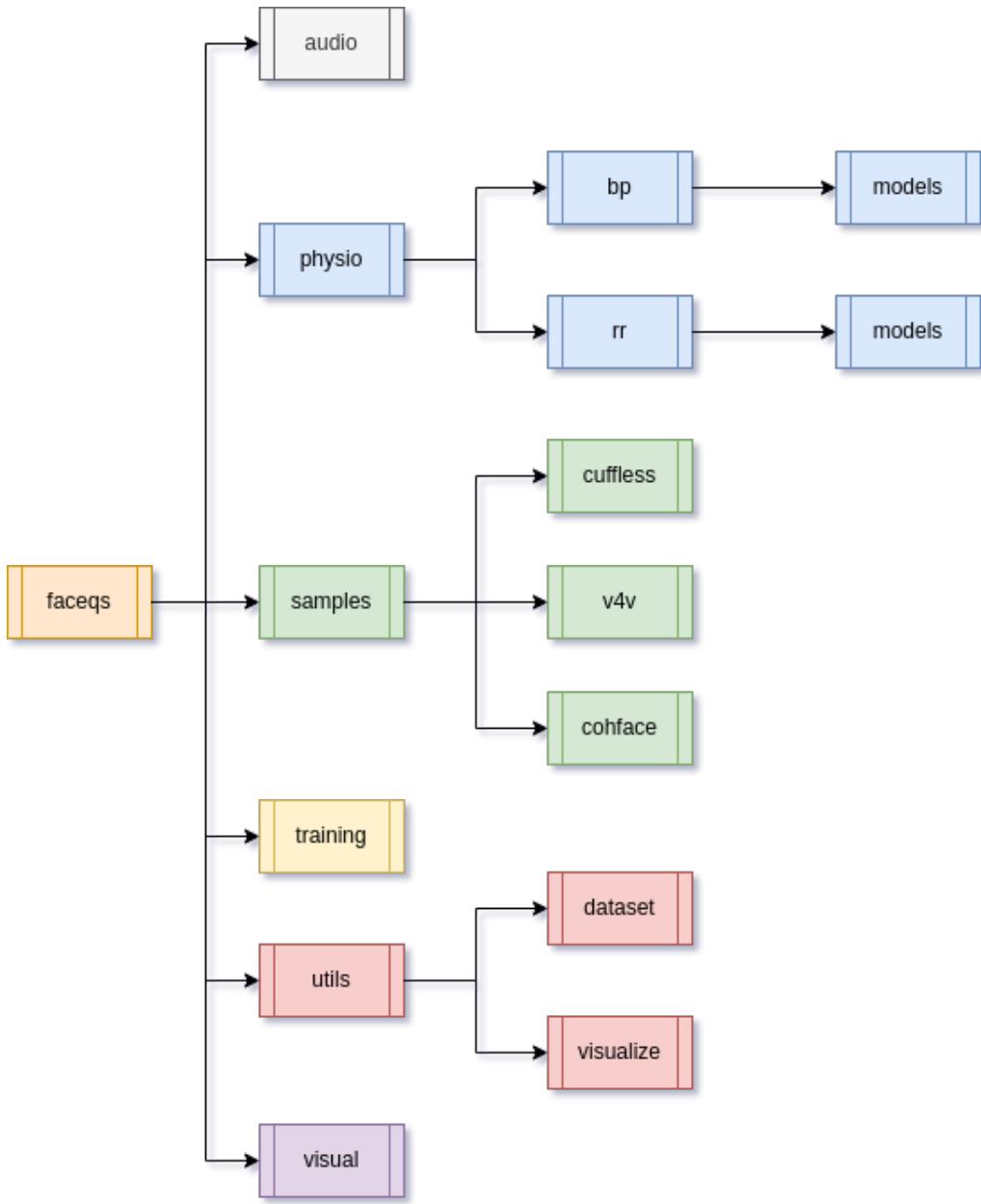
Il framework FaceQs è composto da due sezioni principali:

- libreria FaceQs,
- cartella *notebooks* con file dimostrativi.

All'interno della cartella *notebooks* sono presenti alcuni esempi dimostrativi dell'utilizzo del framework: dalla lettura dei dati all'addestramento dei modelli di machine learning fino al semplice utilizzo di una delle funzionalità disponibili. Inoltre questi strumenti aiutano a comprendere meglio come lavora la libreria oltre che guida per l'utilizzo.

La libreria del framework si suddivide in diversi package e sottocartelle, contenenti il codice sorgente python con cui il tutto è stato implementato:

- *audio*, riservato per utilizzi futuri di integrazione audio nelle predizioni.
- *physio*, dove risiede il codice specifico delle stime implementate.
- *samples*, contenente alcuni campioni di dati preprocessati per test.
- *training*, con il codice utilizzato per la parte di machine learning.
- *utils*, per racchiudere funzionalità generali di appoggio.
- *visual*, in cui si gestiscono i video e i volti in essi presenti.



Utilizzando la figura come riferimento, si procede ora ad una spiegazione più dettagliata e precisa delle componenti del framework.

Come già accennato precedentemente, il package *audio* è riservato per sviluppi futuri del sistema. Al momento vuoto, si prevede l'eventualità di inserire analisi basate su un segnale sonoro anziché puramente visivo, integrando informazioni ad esso associate nelle varie stime.

Il package *physio* si suddivide in due parti (al momento). In ciascuna di esse risiede il codice di algoritmi specificatamente legati a quel tipo di analisi: *bp* per "blood pressure" e *rr* per "respiration rate", con una dedicata sottosezione *models* in cui posizionare i modelli pre-addestrati pronti all'uso.

La cartella *samples* contiene alcuni campioni di dati pre-processati in precedenza che possono essere utilizzati da un utente per fare prove personali e testare il fra-

mework, senza dover partire costruendosi tutto da solo. I campioni sono suddivisi tra i dataset utilizzati nei diversi addestramenti da cui essi provengono: *cuffless*, *v4v*, *vitaldb* e *COHFACE* (si veda il capitolo 4 per una spiegazione dettagliata).

Il package *training* viene utilizzato per raccogliere il codice di machine learning generale, comune alle varie operazioni del framework per l'addestramento dei modelli. In esso non viene fatto esplicitamente riferimento ad un algoritmo specifico di addestramento, che può essere deciso dall'utente, ma si racchiudono le funzionalità algoritmiche di pre e post-processing e i metodi che si occupano del calcolo del modello migliore.

Il package *utils* si suddivide nelle parti *dataset* e *visualize*. Nella prima parte si trova il codice specializzato per la lettura dei vari dataset utilizzati. Esso si pone come interfaccia estensibile tra FaceQs e i dataset utilizzati, con possibilità, in caso di aggiunta di un nuovo dataset, di integrare facilmente codice per la sua lettura. Nella seconda parte sono presenti operazioni di visualizzazione e valutazioni dei dati ottenuti tramite il framework. Si tratta di operazioni di "plot", utili per mostrare grafici sulla performance e accuratezza di un modello, e di calcolo di metriche varie come il classico "MAE" (Mean Average Error) per valutazioni di carattere matematico. Infine abbiamo *visual*, in cui risiede il punto di partenza di utilizzo di FaceQs con la classe *Video*(path) che legge un video e ne espone la possibilità di ottenere dati sulle predizioni o di estrarre i landmark facciali (punti in posizioni rilevanti del volto calcolati per ogni frame del video) attraverso la libreria di Google Mediapipe.

2.2 Utilizzi del Framework

FaceQs ha lo scopo di eseguire analisi di parametri vitali in maniera virtuale attraverso l'analisi di un video. Queste analisi possono essere effettuate all'interno del framework in due modi:

- In modo semplice, tramite strumenti e modelli preconfezionati per un uso immediato, passando direttamente al calcolo computazionale.
- In modo avanzato, in quanto vengono forniti strumenti per utenti esperti allo scopo di addestrare e utilizzare i propri modelli personalizzati coerentemente con la struttura del framework.

Si vedranno ora esempi concreti di chiamate e risultati attualmente implementati e disponibili all'utilizzo.

2.2.1 Eseguire un analisi

Per eseguire una semplice analisi è sufficiente, dopo aver installato la libreria, fare riferimento alla classe `Video` presente nel package `faceqs/visual/video`, costruire un oggetto utilizzando il percorso assoluto riferito al video da analizzare e richiamare il metodo appropriato. Di seguito sono mostrati alcuni esempi pratici. Per un'analisi di frequenza respiratoria:

```
1 from faceqs.visual.video import Video  
2  
3 video_path = "/var/data/VHR/V4V_Dataset/Phase1_TrainingValidationSets/  
    Videos/train/F001_T7.mkv"  
4 rr = Video(video_path).measureRR()  
5  
6 print(" --- Respiratory Rate: {} ---".format(rr))
```

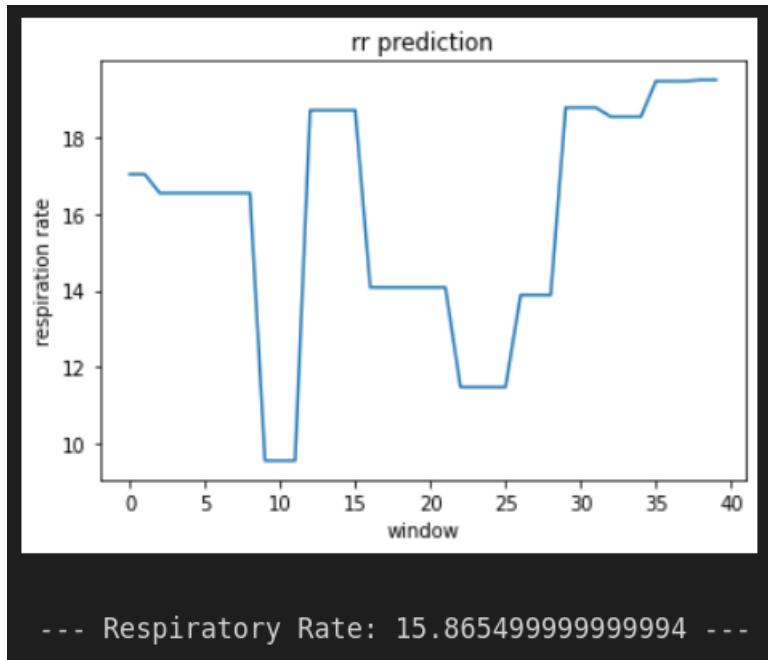


Figura 2.1: Esempio di uso del codice di misurazione della frequenza respiratoria.

Il metodo `measureRR()` restituisce un singolo valore numerico corrispondente alla frequenza respiratoria calcolata. Se il codice viene eseguito in un notebook, verrà inoltre visualizzato a schermo un grafico contenente i valori di respirazione predetti durante l'analisi dei frame del video nelle finestre.

Per un'analisi di pressione sanguigna:

```

1 from faceqs.visual.video import Video
2
3 path = "/var/data/VHR/V4V_Dataset/Phase1_TrainingValidationSets/Videos/
4     train/F001_T1.mkv"
5
6 video = Video(path)
7 video.measureBP()

```

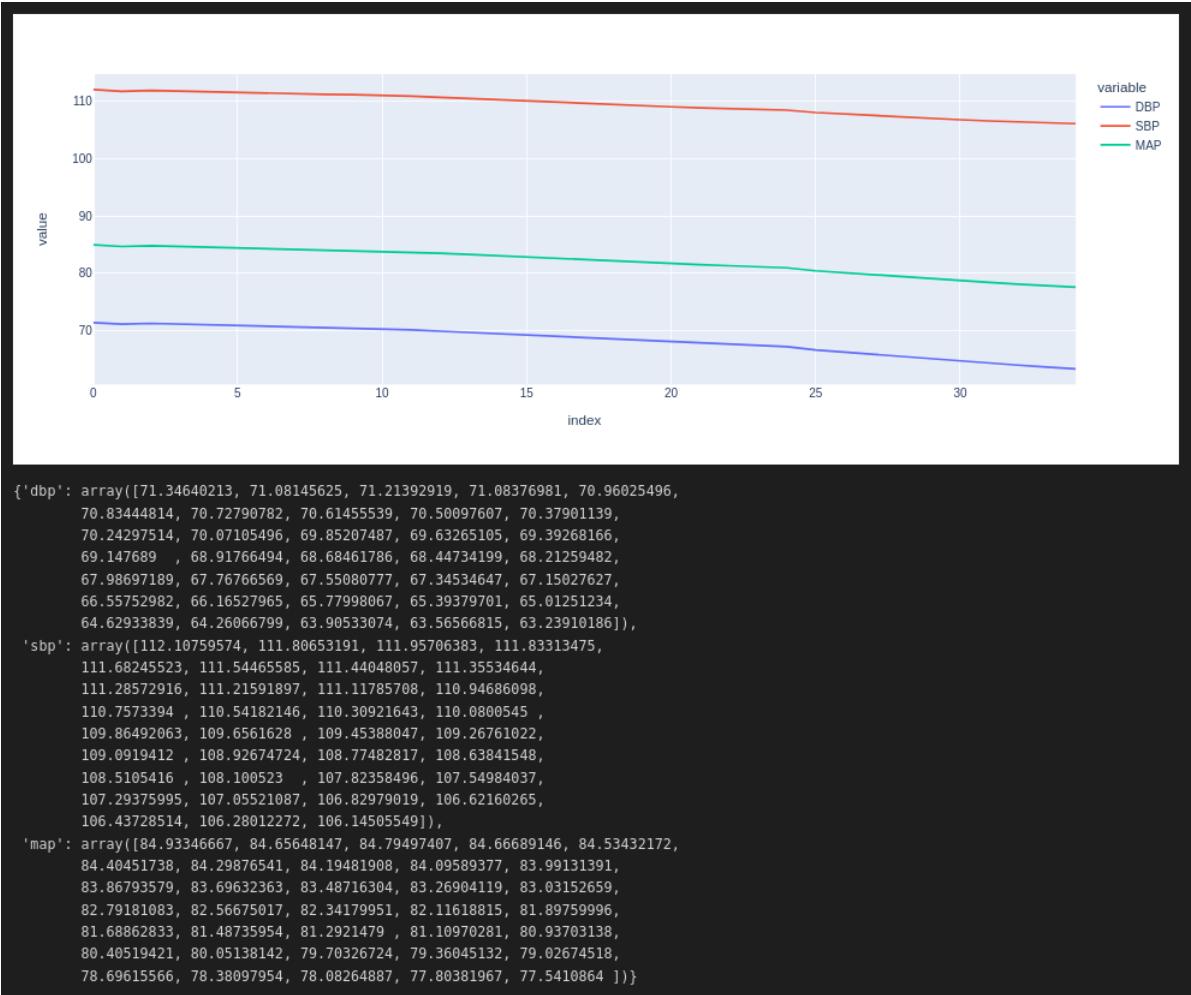


Figura 2.2: Esempio di uso del codice di misurazione della pressione sanguigna.

La frequenza respiratoria è espressa in un unico valore, mentre la pressione sanguigna risulta divisa in tre valori: pressione diastolica, sistolica e arteriosa. Per questo motivo viene restituito come unico oggetto un dizionario con le tre liste di valori associati, oltre ad un grafico dei tre valori.

2.2.2 Personalizzare le analisi

FaceQs fornisce agli utenti analisi tramite modelli preconfezionati con l'algoritmo di Random Forest, assieme ad altri parametri internamente impostati. Per questo motivo può insorgere la necessità di strumentazioni e analisi più specifiche, sia per un utilizzo pratico che per studiare più a fondo l'argomento.

Il framework è stato realizzato in modo strettamente modulare, per permettere alle varie fasi dell'analisi di essere gestite differentemente, con soluzioni ad-hoc definite da un utente esterno. La pipeline di addestramento è la seguente:

1. Pre-processing dei dati, con generazione di file associati. È presente un metodo preimpostato in `faceqs.utils.dataset`. Di seguito è riportato uno snippet per generare dei csv contenenti il segnale rppg analizzato con NeuroKit2 con specifici parametri sul dataset *Vision For Vitals*.

```
1 v4v_data = V4V_Dataset()
2 wsize = 30
3 rppg_extractor = pp.extract_rppg_holistic
4 method = "holistic"
5
6 pp.preprocess_dataset_v4v(
7     "/{}_nk_w{}".format(method, wsize),
8     v4v_data,
9     ppg_extractor=rppg_extractor,
10    ppg_analyzers=["nk"],
11    pp_type=pp.PREPROCESS_TYPE.BP,
12    windowSize=wsize)
13
```

2. Caricamento in memoria di dati precedentemente preparati, suddividendo le feature X dai valori y. Anche qui è presente un metodo preimpostato contenuto nel modulo `faceqs.utils.dataset`.

```
1 X, y = reader.getDataFromDir(
2     root_dir=root_dir,
3     num_labels=1,
4     excluded=[],
5     slice_start=0,
6     slice_end=None,
7     sort=True,
8     verbose=False)
9
```

3. Utilizzo dei valori caricati per un addestramento del modello, da posizionare successivamente nel package relativo:

- `faceqs.physio.bp.models` per modelli di pressione sanguigna.
- `faceqs.physio.rr.models` per modelli di respirazione.

È importante rispettare la nomenclatura dei modelli esistenti.

L'addestramento implementato in FaceQs, varia a seconda del segnale fisiologico. Nel caso di pressione sanguigna sono stati generati modelli Random Forest o Deep Neural Network, nel caso di respirazione sono stati generati modelli Random Forest. La generazione segue la classica procedura di cross-validation partendo prima da una "Randomized-search" seguita da una "Grid-search". Di queste procedure ne viene descritto il funzionamento ai capitoli 3 e 4.

2.3 Estensioni di FaceQs

Ad oggi FaceQs rappresenta un framework in una primissima versione che mette a disposizione le API per l'analisi del battito cardiaco, della frequenza respiratoria e della pressione sanguigna. Essendo un progetto con svariati obiettivi il progetto continuerà ad essere sviluppato portando con sé altre possibili analisi e non solo.

Un prossimo obiettivo che rappresenta un lavoro di non piccole dimensioni è quello di studiare la correlazione tra i movimenti facciali (determinati dai landmark) e alcuni dei principali segnali fisiologici (quali pressione sanguigna e frequenza respiratoria) per determinare lo stato emotivo oggettivo del soggetto. Con stato emotivo oggettivo si intende lo stato d'animo non simulato. Tale lavoro prevede l'utilizzo delle *Graph Neural Network* (GNN) ovvero di particolari reti neurali che si basano sull'uso di grafi dove i nodi corrispondono, in questo studio specifico, ai landmark facciali.

Inoltre, una prossima ricerca si baserà sull'applicazione dell'analisi di tutti questi segnali su video *online*, ovvero video generati al momento (come ad esempio qualunque video prodotto da una webcam durante una video-conference).

Capitolo 3

Misurazione della pressione sanguigna da remoto

Durante lo studio e la ricerca in questo ambito sono state osservate e analizzate diverse strade per ottenere il valore della pressione a partire da un semplice video. La misurazione della pressione non è, in generale, una misurazione semplice. A supporto di questo parere è stata condotta una ricerca scientifica da un gruppo di ricercatori. In tale studio alcuni medici hanno insegnato a 720 pazienti come si utilizzano misuratori di pressione da polso. Le misurazioni rilevate, ogni mattina ed ogni sera per sette giorni di fila, sempre agli stessi orari, hanno denotato una forte imprecisione nei risultati, con divergenze tra i 5 e i 10 mmHg. Questo denota che la misurazione, quando fatta da un gran numero di persone di cui la maggior parte si presume essere inesperta, porta spesso a risultati non troppo precisi.

In questa tesi, sono stati fatti diversi tentativi per giungere ad una stima di misurazione affidabile. Dopo svariate prove è stata adottata una soluzione che non parte direttamente dal video per estrarre il valore della pressione sistolica, diastolica oppure arteriosa. Tale soluzione prevede di seguire una serie di passaggi, (o *pipeline*) illustrati in figura 3.1. Come riportato nell'immagine il primo passaggio consiste nel prelevare dal video un segnale particolare denominato segnale fotopletismografico. Tale segnale indica come varia il flusso di sangue nel corpo, in particolar modo nel viso. Una volta estrapolato questo segnale viene filtrato e pulito per poi essere passato in input ad un analizzatore morfologico del segnale il quale permette di ottenere una serie di indici e valori che rappresentano caratteristiche fisiologiche quali battito cardiaco ed altri valori ad esso legati. Questa serie di indici ottenuti dall'analizzatore viene usata (solo una piccola parte nel caso di Random Forest) per effettuare la misurazione effettiva della pressione sistolica, diastolica e arteriosa. Concretamente, questa soluzione, prevede di generare un segnale fotopletismografico a finestre sovrapposte con dimensione fissata. Questa tecnica di finestratura è dovuta al fatto che per ogni finestra

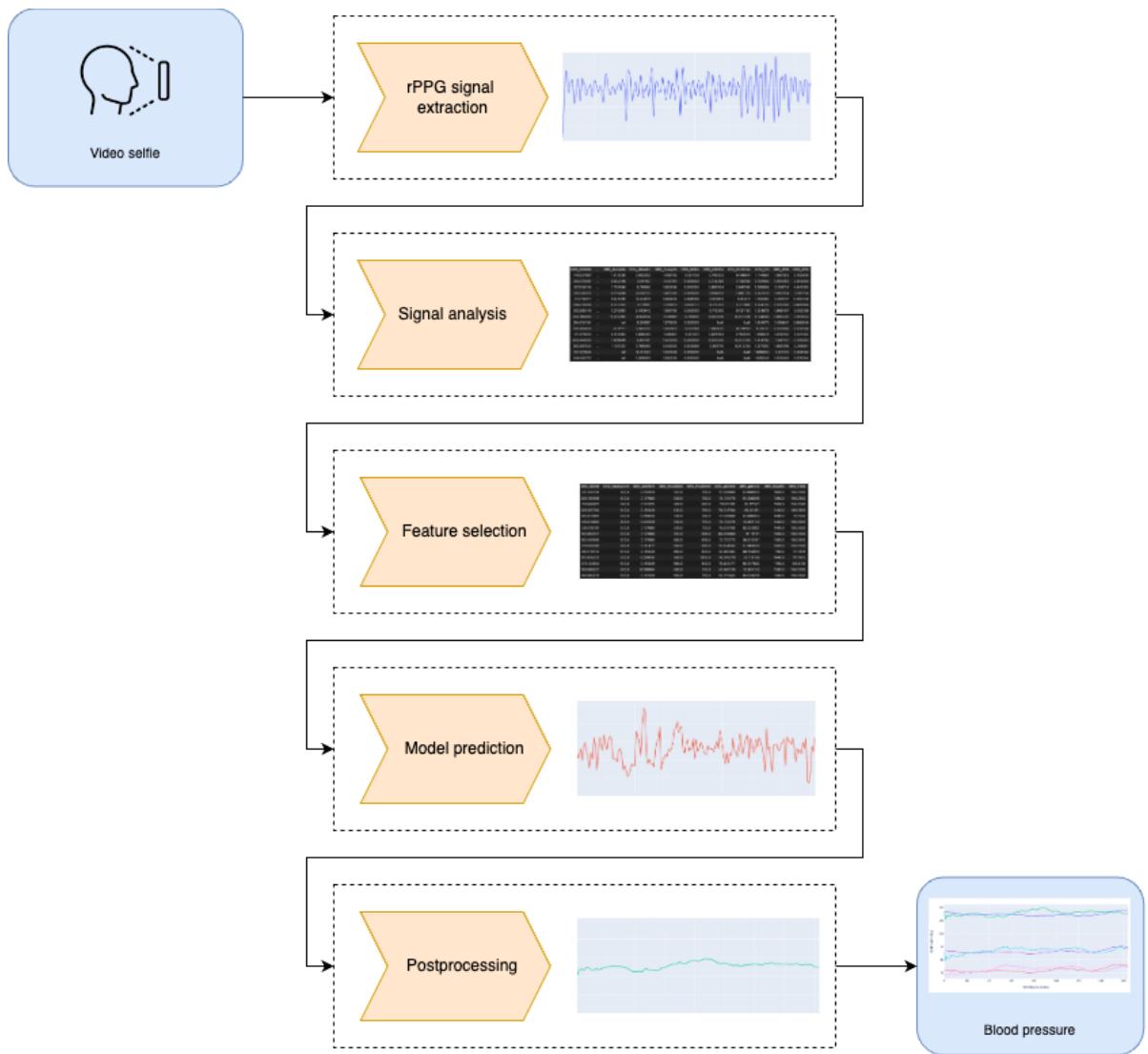


Figura 3.1: Pipeline usata per misurare la pressione sanguigna (sistolica, diastolica e arteriosa) tramite un video.

generata viene riportato un singolo valore di pressione sanguigna. Questo significa che se venisse analizzato un video nella sua interezza, si avrebbe un singolo valore di pressione. Con la finestratura, quindi, si ha la possibilità di definire l'andamento della pressione con una granularità inversamente proporzionale alla lunghezza della finestra, ovvero maggiore sarà la sua durata e minore saranno le misurazioni fatte. Al contrario, minore sarà la finestratura maggiore sarà il numero di misurazioni. Il fatto che siano sovrapposte permette di non scartare una gran parte del video. Un ulteriore necessità dell'uso delle finestre è dovuto al fatto che questo tipo di misurazioni si basano su serie temporali di durata minima, quindi su singolo frame del video non sarebbe possibile stimare il segnale fotopletismografico e di conseguenza tutta la *pipeline* non potrebbe produrre risultati ragionevoli. Sono state analizzate più finestrature del video per capire se, di fatto, la dimensione della finestra possa avere una qualche influenza e in caso positivo quanto questa pesi. Nello specifico sono state analizzate finestre di 20, 30, 40, 50 e 60 secondi. Le scelte degli estremi sono state determinate dal fatto che 20 secondi è il tempo minimo analizzabile mentre 60 secondi è il massimo accettabile per una applicazione che possa avere un uso pratico sensato.

In questo capitolo vengono quindi illustrate la varie fasi di questo processo fino ad arrivare a modelli che leggono i valori ottenuti dall'analisi del segnale fotopletismografico e il successivo post-processing applicato.

3.1 pyVHR

Il framework pyVHR (11), come detto nel capitolo precedente, rappresenta un framework sviluppato presso PHuSe Lab dove sono racchiuse diverse funzionalità conosciute in letteratura per l'estrapolazione del segnale PPG da remoto e il calcolo della frequenza cardiaca. Esso rappresenta il primo punto ed una parte essenziale della *pipeline* per la misurazione della pressione sanguigna.

Nell'uso pratico tale framework mostra la sua utilità solamente nell'estrazione del segnale fotopletismografico in quanto questo sarà il segnale che verrà utilizzato. La stima della frequenza cardiaca a partire da pyVHR invece non avrà nessun utilizzo diretto.

3.1.1 Estrazione del segnale rPPG

L'estrazione del segnale fotopletismografico da remoto è un problema molto conosciuto in letteratura, infatti, come riportato da Daniel McDuff (principal researcher presso Microsoft), la ricerca in questo ambito è la più ricca sia di risultati che di ricerche. In (10) è mostrato un grafico dove sono enumerate le ricerche ed i lavori, suddivisi per anni, di tutto ciò che inerisce all'estrazione dei segnali vitali a partire dalla videocame-

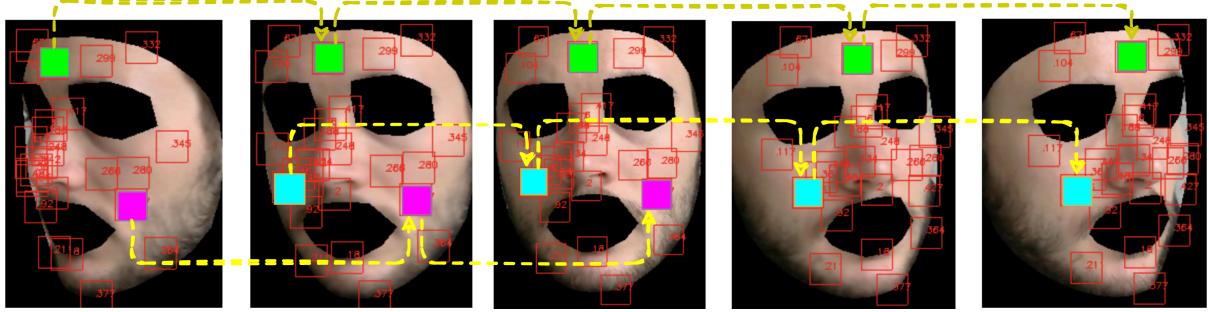


Figura 3.2: Tracking delle patch generate.

ra. Il framework pyVHR racchiude molte soluzioni conosciute per estrarre il segnale a partire da un semplice video. Fondamentalmente le tecniche adottate sono due, tra loro distinguibili: una è la tecnica tradizionale basata sull'analisi del segnale rPPG attraverso un segnale RGB estratto dal volto (o parte di esso), la seconda è una tecnica che si basa sulle reti neurali, nello specifico sulle Deep Neural Networks.

Nel caso della soluzione basata su reti neurali non vi è una pipeline molto complessa. È sufficiente passare come input il video alla rete neurale e automaticamente verrà restituita la stima della frequenza cardiaca.

Nel caso, più comune, della stima tramite algoritmi tradizionali pyVHR segue una pipeline più complessa. Nella prima fase denominata *skin extraction* viene delineato il volto (tramite framework esterni come ad esempio Google Mediapipe (12)) ed eliminato tutto il contenuto circostante. Successivamente viene effettuata la computazione del segnale RGB facciale. Questa procedura può essere processata secondo due diverse tecniche: tecnica olistica e tecnica patch. Queste due tecniche si differenziano per le zone del volto elaborate. Nel caso del metodo olistico, come dice il nome stesso, viene extrapolato il volto nell'insieme. Nel caso di metodo patch, invece, vengono estrapolate differenti zone del volto. In figura 3.2 è mostrato un esempio di come lavorano le patch generate da pyVHR. I riquadri rossi rappresentano le patch computate da pyVHR e inoltre in figura è possibile vedere come vengono tracciate anche in seguito ad uno spostamento del volto. Questa fase appena descritta, denominata *RGB signal processing*, si sviluppa secondo finestre sovrapposte con un salto prefissato, fornendo così un segnale RGB per ogni finestra distinta.

Una volta estratto il volto in una o più zone (a seconda del metodo utilizzato) e processato il segnale RGB, optionalmente a tale segnale possono essere applicati filtri standard come ad esempio:

- filtro RGB: rimuove il segnale che presenta, in almeno un frame, il valore RGB al di fuori di un intervallo prestabilito;
- detrend;
- standardizzazione;

- filtro butterworth.

Tale fase prende il nome di *pre-filtering*. A questo punto segue la fase vera e propria di estrazione del segnale rPPG, denominata *BVP extraction* (dove BVP indica *blood volume pulse*). Essa consiste nell'applicazione di uno specifico metodo rPPG per ogni finestra RGB generata dalle fasi precedenti. I metodi rPPG implementati all'interno di pyVHR sono algoritmi conosciuti in letteratura come ad esempio:

- Green Channel (13),
- POS (14),
- CHROM (15),
- ICA (16).

Una caratteristica particolare di pyVHR consiste nel fatto che la maggior parte degli algoritmi sono stati implementati anche per essere eseguiti su scheda video Nvidia. Questo permette di ridurre drasticamente i tempi di elaborazione.

La fase che segue è, come la fase precedente, una fase di filtraggio del segnale. In questo caso la fase viene denominata *post-filtering* in quanto viene filtrato il segnale computato. I filtri sono quindi applicati non più al segnale RGB bensì al segnale rPPG appena generato. Alcuni dei filtri utili messi a disposizione dal framework sono:

- filtro passa banda: viene filtrato il segnale usando un passabanda di ordine N con uno specifico intervallo di frequenze,
- detrend,
- zero-mean: permette di rimuovere i disturbi inseriti dai componenti elettrici,

pyVHR, essendo un framework in grado di stimare il battito cardiaco, prevede un ultimo passaggio che consiste nel passare dal segnale rPPG al battito cardiaco. Per l'uso che FaceQs fa di pyVHR, questo risulta una parte superflua e pertanto non viene utilizzato.

In figura 3.3 è mostrato uno schema della pipeline adottata da pyVHR.

3.2 NeuroKit2 come analizzatore del segnale rPPG

Una volta ottenuto il segnale rPPG tramite quanto descritto sopra, si passa alla fase "due", ovvero la fase in cui tale segnale viene analizzato. Esistono diversi framework in grado di analizzare un segnale fotopletismografico, tra cui HeartPy, ma NeuroKit2 ha rappresentato la scelta finale perché ben documentato, molto versatile, semplice

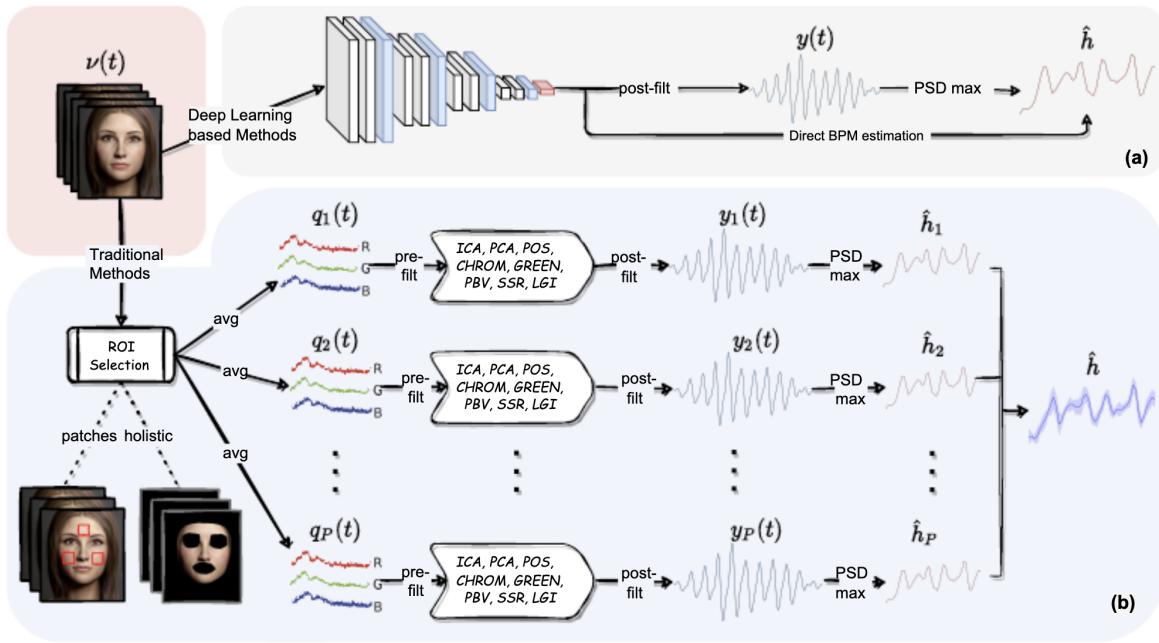


Figura 3.3: Pipeline di pyVHR.

da utilizzare e soprattutto ricco di funzionalità. Questi punti di forza hanno quindi permesso di svolgere la ricerca semplificando il lavoro nel complesso.

NeuroKit2 è un framework scritto in python, in grado di analizzare diversi segnali tra cui ECG, RSP, EDA, EMG e, ovviamente, quello fotopletismografico (PPG). Esso basa le analisi su funzionalità suddivisibili in tre diversi livelli. Nel primo livello, ovvero quello più basilare, si trovano funzionalità comuni a tutte le tipologie di segnale da analizzare. Alcune di queste sono:

- filtraggio del segnale, rimuovendo così le impurità dello stesso;
- interpolazione, ovvero la classica interpolazione tra una serie di valori;
- resampling;
- `find_peaks`, ovvero la capacità di individuare massimi e minimi locali.

Nel secondo livello, vengono di fatto utilizzate le funzionalità basilari in maniera congiunta per ottenere funzionalità specifiche a seconda del tipo di segnale. Alcune di queste analisi sono ad esempio la pulizia del segnale, la classificazione delle varie fasi oppure la capacità di individuare i picchi. Inoltre, per ogni operazione è possibile impostare vari parametri "standardizzati" a seconda del metodo selezionato per effettuare tale operazione. In realtà, però, NeuroKit2 offre la possibilità di specificare i propri parametri in qualsiasi operazione messa a disposizione dal toolbox.

Infine, nel terzo livello, ovvero quello che viene poi utilizzato in maniera più frequente dagli utilizzatori del framework, si trovano le funzionalità per l'elaborazio-

ne e l'analisi dei vari segnali. Queste funzionalità, ovviamente, si basano sull'uso congiunto delle funzionalità di secondo livello.

Nello specifico, il segnale rPPG ottenuto tramite il framework pyVHR viene sottoposto a due diverse funzionalità di NeuroKit2. La prima è una funzione denominata *process* ovvero una funzione che riceve in input un segnale PPG e restituisce un DataFrame contenente diverse informazioni tra cui il segnale *PPG raw*, il segnale *PPG clean*, il battito cardiaco e i picchi PPG individuati. Ogni record del DataFrame rappresenta una "epoca" ovvero un intervallo di tempo dove vengono definite diverse informazioni, in questo caso il battito cardiaco e i picchi. Successivamente, questo DataFrame viene utilizzato come input di un'ulteriore funzione denominata *analyze* la quale permette, appunto, di analizzare le varie epoche. Questa analisi riporta un insieme di indici piuttosto ampio (circa 90 indici) con diverse informazioni che racchiudono caratteristiche fisiologiche relative alla frequenza cardiaca e alla sua variabilità.

Nel concreto, sono stati utilizzati una ridotta quantità di indici selezionati mediante un processo denominato *feature selection* (che viene descritto nella prossima sezione). Alcune di queste sono:

- HRV_MedianNN: la mediana degli intervalli RR(17),
- HRV_MCVNN: la deviazione della mediana assoluta degli intervalli RR divisi per la mediana degli intervalli RR,
- HRV_Prc20NN: il ventesimo percentile degli intervalli RR,
- HRV_Prc80NN: l'ottantesimo percentile degli intervalli RR,
- HRV_pNN50: la proporzione degli intervalli RR maggiore di 50ms, rispetto al numero totale di intervalli RR,
- HRV_pNN20: la proporzione degli intervalli RR maggiore di 20ms, rispetto al numero totale di intervalli RR,
- HRV_MinNN: il minimo degli intervalli RR,
- HRV_MaxNN: il massimo degli intervalli RR,
- HRV_TINN: un parametro geometrico dell'HRV, o più specificamente, la larghezza della linea di base della distribuzione degli intervalli RR ottenuta per interpolazione triangolare, dove l'errore dei minimi quadrati determina il triangolo,
- HRV_SD2: deviazione standard lungo la linea identitaria. Indice delle variazioni HRV a lungo termine

Per una lista completa di tutti i parametri generabili tramite NeuroKit2 è possibile seguire la documentazione relativa (18).

3.3 Feature selection

Come appena detto, prima di dare in pasto ai modelli sopra citati (Random Forest Regressor e Deep Neural Network) i dati generati da NeuroKit2, viene fatta una selezione dei dati da utilizzare. Questa è una prassi molto comune quando si ha a che fare con dati ad alta dimensionalità.

La feature selection rappresenta il processo di riduzione delle dimensioni da utilizzare come input di un qualunque tipo di modello predittivo. Questo processo primario permette di ridurre in primo luogo il costo computazionale durante la fase di modellazione e, in molti casi, di aumentarne le performance. Infatti, quando si hanno tante dimensioni da considerare, la probabilità che alcune di queste dimensioni siano superflue o ridondanti cresce in modo direttamente proporzionale con il numero di feature da considerare. Spesso questo processo è fondamentale anche per motivi di natura tecnica: dover allenare modelli con una grande mole di dati implica avere una grande quantità di memoria sul sistema che elabora l'allenamento del modello.

Nel mondo della *feature selection* esistono due diversi tipi di selezione fondamentali:

- supervisionato: principalmente tecniche che si basano sul peso delle dimensionalità rispetto alla/e variabile/i target;
- non-supervisionato: a differenza di quello supervisionato, non avendo delle variabili a cui fare riferimento, la selezione si basa sulla varianza delle stesse e su quelle che appaiono più significative in base a particolari metriche.

Nel caso di feature selection supervisionato si distinguono principalmente tre differenti metodi:

- *wrapper*: questa tecnica si basa su uno specifico modello di apprendimento. Viene effettuata una ricerca *greedy* valutando un sottoinsieme delle feature applicate al modello specifico. Le metriche per valutare quale dei sottoinsiemi definiti sia il migliore si basa su metriche dipendenti dal tipo di problema affrontato: *mae*, *mse* nel caso di problemi numerici, accuratezza, precisione, recall e simili nel caso di problemi categorici.
- *filter*: vengono filtrate tutte le feature più "significative". L'importanza di una feature è determinata dalla correlazione presente tra la feature e il dato target.

- *embedded*: in questo caso la *feature selection* è, come suggerisce il nome stesso, integrata direttamente nel modello di apprendimento. Questo metodo porta i vantaggi delle tecniche sopra descritte in quanto analizza le feature relativamente ad uno specifico modello di apprendimento, ma lo fa con la velocità che si avrebbe utilizzando il *filter*: nel momento in cui il modello apprende, controlla quali siano le migliori feature che portano a migliori risultati.

La *feature selection* è una tecnica legata alla "riduzione delle dimensionalità". È bene però sottolineare che la *feature selection* e la *feature extraction* sono processi differenti: uno si cura di rimuovere e selezionare solo un certo sotto-insieme di feature, l'altro si occupa di creare una proiezione dei dati su un numero di dimensionalità preciso.

Nella pipeline utilizzata nello svolgimento di questa tesi è stata utilizzata una tecnica supervisionata che prende il nome di *SelectKBest*. Essa è un metodo implementato all'interno della libreria di sklearn (19) nel package di *feature_selection*. *SelectKBest* permette di selezionare un numero k di feature specifico basandosi su funzioni differenti. Di seguito ne sono riportate alcune:

- *f_classif*: utilizzabile nei problemi di classificazione, si basa sul computare la curva di ANOVA sul test di Fisher-Snedecor (anche detto F-test) il quale è volto a verificare che due popolazioni abbiano la stessa varianza,
- *f_regression*: si basa sul calcolo del coefficiente di Pearson usando una formula particolare il cui risultato viene poi convertito in F-score (un punteggio legato allo F-test) e successivamente a un valore-p.
- *mutual_info_classif*: permette di stimare la *mutual information* (MI) per un valore categorico. La *mutual information* tra due valori, rappresenta la dipendenza tra questi ultimi, maggiore è la MI e maggiore è la loro correlazione,
- *mutual_info_regression*: analogamente alla *mutual_info_classif* viene stimata la MI su valori continui per determinare la correlazione tra le variabili considerate.

Il problema della misurazione della pressione sanguigna, essendo un problema di natura numerica, ha implicato l'uso di stime quali: *f_regression* e *mutual_info_regression*. In tutti i modelli predittivi generati la seconda si è rivelata la tecnica migliore, tanto che verrà approfondito nel capitolo 4.

L'uso di *feature selection* è anche una soluzione al problema della predizione di un video "qualunque". Questo perché non si ha certezza che NeuroKit2 sia in grado di calcolare esattamente gli stessi indici per ogni video, o meglio, non si può avere certezza che gli indici utilizzati in fase di allenamento per un qualunque modello siano sempre calcolabili e siano sempre valori finiti (quindi non siano valori *NaN* oppure

inf). Durante lo sviluppo della tesi è stato osservato che dei 90 indici generabili tramite NeuroKit2, un sottoinsieme formato da circa 55 indici è sempre generabile, di conseguenza è stato scelto un numero inferiore a 55 sia per il motivo appena descritto, sia perché le prestazioni fornite dai modelli (specialmente nel caso di Random Forest) degradano notevolmente all'aumentare delle feature selezionate. Nella prossima sezione vengono quindi introdotti i modelli di machine learning utilizzati a seguito di questa fase di *feature selection*.

3.4 Modelli di machine learning

Il Machine Learning (ML) insegna ai computer a compiere attività in modo naturale come gli esseri umani o gli animali: imparando dall'esperienza. In sostanza, gli algoritmi di ML usano metodi matematico-computazionali per apprendere informazioni direttamente dai dati, senza modelli matematici ed equazioni predeterminate. Gli algoritmi di ML migliorano le loro prestazioni in modo "adattivo" mano a mano che gli "esempi" da cui apprendere aumentano.

I tipi di algoritmo del ML differiscono nel loro approccio, nel tipo di dati che utilizzano, che producono e nel tipo di attività o di problema che devono risolvere. Il ML può generalmente essere suddiviso in tre macro categorie:

- supervisionato,
- non supervisionato,
- apprendimento con rinforzo.

A queste si aggiunge solitamente una quarta categoria denominata semi-supervisionato.

3.4.1 Approccio supervisionato

L'approccio supervisionato (o *supervised*) è una tecnica che prevede di lavorare su un insieme di dati associati ad etichette definite dall'utente. Le etichette sono delle classi (o gruppi) entro le quali sono suddivisi i dati. Sapendo che ogni dato è associato a un'etichetta si vuole arrivare a cogliere la relazione che vi è tra dati ed etichette, così da poter predire le etichette a partire dai dati, anche lavorando con dati non visti durante la fase di apprendimento. Questa tecnica può fornire due diversi tipi di risultati: discreti o continui. Per comprendere meglio questo concetto si può fare un esempio.

Si considerino delle diagnosi mediche fatte su una serie di pazienti. Analizzando le diagnosi un medico è in grado di definire se il paziente è in salute o meno. Da qui è possibile estrapolare quindi due etichette differenti per il caso: "in salute" e "malato". Fornendo come input a un classificatore questo insieme di dati con le rispettive

etichette appena definite, il calcolatore, tramite un algoritmo supervisionato, sarà in grado di fornire delle predizioni sulla possibile etichetta da attribuire ad ogni nuova diagnosi. È importante che vi sia a disposizione una quantità consistente di dati in input. In altre parole, è importante che vi siano molti dati da analizzare, perché quando questo non succede la capacità di predizione del sistema che si ottiene è spesso di scarsa qualità.

In questo esempio sono state utilizzate solamente le classi "in salute" e "malato", ma nulla vieta di definirne una terza o una quarta. Ad esempio è possibile etichettare un paziente come "asmatico" o "diabetico". Nel caso in cui non si voglia avere una classificazione della salute del paziente, ma si voglia quantificare l'aspettativa di vita, non è più possibile ricorrere a dei classificatori. Da qui nasce la necessità di passare da un valore discreto ad un valore continuo, pertanto viene utilizzato un modello diverso ovvero i regressori che costituiscono un modello per predire valori continui. Ad esempio, si potrebbe voler quantificare, data una specifica diagnosi, il tempo di guarigione per un paziente malato che per definizione si definisce su una scala di numeri continua. I modelli di ML supervisionati hanno l'obiettivo di eseguire, con la maggior precisione possibile, una predizione su dati nuovi, mai visti prima. Per raggiungere questo scopo è necessario assicurarsi che il modello produca stati lontani sia dal sovra-adattamento (*overfitting*) che dal sotto-adattamento (*underfitting*). L'*overfitting* si verifica quando il modello tende ad adattarsi in maniera eccessiva ai dati che gli sono stati forniti per allenarsi, non permettendo la generalizzazione a nuovi insiemi di dati. L'*underfitting* invece, si verifica nel caso contrario dell'*overfitting*, quando il modello si basa su schemi troppo semplici e poco robusti, il che comporta la definizione di regole con scarsa qualità per la predizione di nuovi elementi.

Per esemplificare si prenda spunto da un caso riportato in (20). Basandosi sulla Tabella 3.1 si noti quali sono i problemi generabili dall'*overfitting* e dall'*underfitting*. Si supponga di voler predire se un cliente vorrà acquistare una barca. Osservando attentamente la tabella si può notare che applicando la regola: "Se un cliente ha più di 45 anni, ha meno di 3 figli o non è divorziato, allora vorrà comprare una barca", tutte le predizioni (su questo dataset) saranno corrette. Ma questo significherebbe anche che se in futuro un cliente C volesse comprare una barca e non rispettasse la regola sopra definita (magari perché ha semplicemente 44 anni o perché ha 4 figli), la predizione del sistema sarebbe "C non vuole comprare una barca", quindi errata. Questo è il caso dell'*overfitting*: stabilire le regole di predizione su troppi dati, in maniera troppo rigida. Lo stesso si può fare al contrario, ossia quando le regole di predizione adottate dal modello si basano su pochi dati e/o le regole sono troppo vaghe. Si supponga che il sistema identifichi un cliente come possibile acquirente di una barca se segue la seguente regola: "Se un cliente ha una casa allora vorrà comprare una barca". È naturale, leggendo la regola, pensare che questa sia eccessivamente generica. Questo

Età	Macchine	Case possedute	Figli	Stato civile	Barca
66	1	Sì	2	Vedova	Sì
52	2	Sì	3	Sposato	Sì
22	0	No	0	Sposato	No
25	1	No	1	Single	No
44	0	No	2	Divorziato	No
39	1	Sì	2	Sposato	No
26	1	No	2	Single	No
40	3	Sì	1	Sposato	No
53	2	Sì	2	Divorziato	Sì
64	2	Sì	3	Divorziato	No
58	2	Sì	2	Sposato	Sì
33	1	No	1	Single	No

Tabella 3.1: Statistica per determinare l'acquisto di una nuova barca

è il caso dell'*underfitting*, ossia il caso in cui si definisce un modello che segue regole troppo vaghe, regole che portano a una scarsa qualità di predizione.

Quindi quello che si vuole trovare è un modello che si posiziona a metà tra l'*overfitting* e l'*underfitting*. La complessità del modello, ossia quante e quali variabili vanno considerate, è un importante aspetto da considerare. Con un modello troppo "semplice" si rischia di utilizzare dati poco significativi per effettuare predizioni. Al contrario, quando è troppo complesso, rischiamo di utilizzare troppe variabili che non permettono di focalizzarci su quelle che sono davvero significative, il che può compromettere la performance del modello.

In Figura 3.4 è possibile osservare quanto appena detto. L'asse orizzontale rappresenta la complessità del modello mentre quella verticale rappresenta l'accuratezza della predizione effettuata dello stesso. La curva blu raffigura l'accuratezza della predizione sui dati di allenamento e, analogamente a questa, la curva verde rappresenta l'accuratezza sui dati che non sono stati usati durante la fase di allenamento. Nel caso di una scarsa complessità del modello avremo un'accuratezza di predizione bassa in entrambi i casi. Con il crescere della complessità del modello (e quindi con il crescere della varietà di dati da poter utilizzare) notiamo che la curva blu aumenta il suo livello di precisione, mentre quella verde raggiunge il picco quando si trova ad un giusto compromesso di complessità, superato il quale torna a decrescere. Tale picco è lo *Sweet spot*, cioè il punto che rappresenta il miglior compromesso tra precisione nella predizione e complessità del modello in caso di dati mai visti. La curva verde decrese quando la complessità diventa eccessiva. Con un numero eccessivo di variabili da considerare si rischia infatti di far perdere rilievo alle variabili più importanti tra tutte quelle considerate.

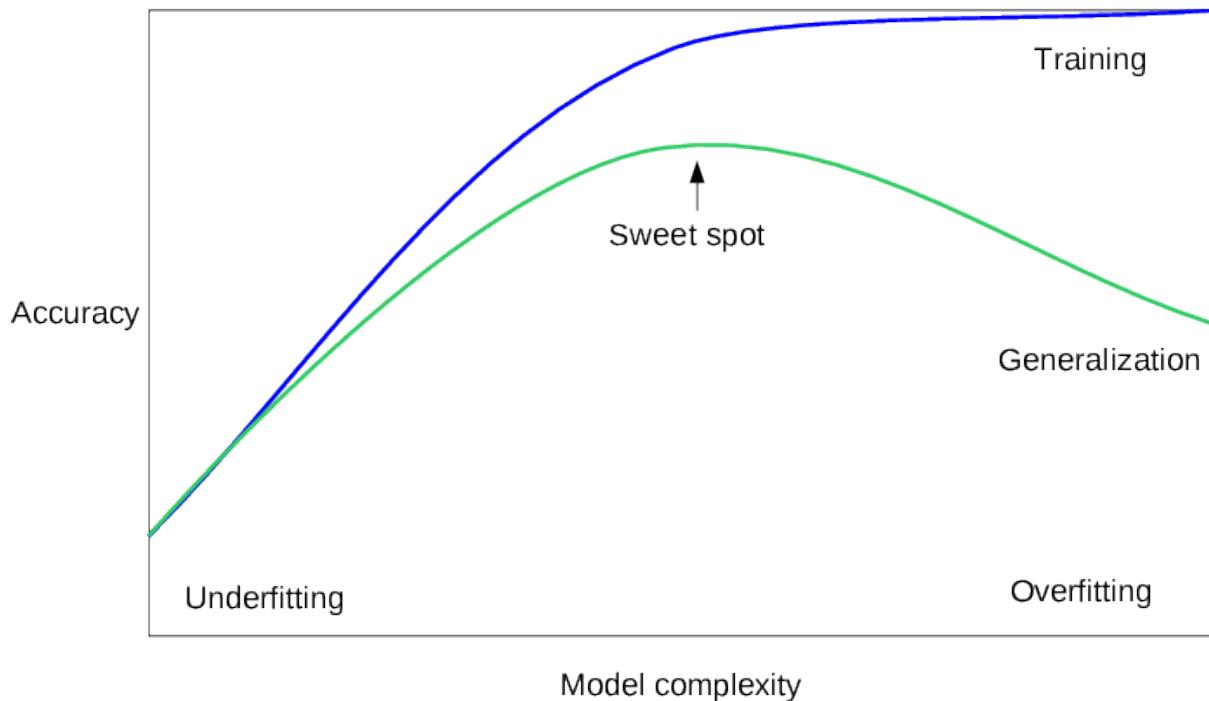


Figura 3.4: Trade-off tra overfitting e underfitting

3.4.2 Approccio non supervisionato

La tecnica non supervisionata (o *unsupervised learning*) è il secondo importante approccio all'applicazione del ML. Cosa si intende dire con "non supervisionata"? Come può una macchina imparare se nessuno la guida nella scelta delle decisioni? Questa è proprio la sfida che si vuole superare con questa tecnica, ovvero far estrapolare al calcolatore delle informazioni "nascoste" all'interno dei dati che gli vengono forniti. Queste informazioni solitamente sono legami, schemi o regole che i dati tendono a seguire. Ci sono diversi utilizzi di ML non supervisionato, in questa sezione ne seguono alcuni. Il principale algoritmo di *Unsupervised Learning* è il clustering, ossia una tecnica in grado di suddividere in gruppi distinti elementi che hanno dati e caratteristiche in comune. Si supponga di aver scattato una serie di fotografie in cui sono raffigurate delle persone e successivamente queste vengano caricate su un social network. Si supponga inoltre che durante il caricamento il social network preso in considerazione visioni le foto e applichi proprio un algoritmo di clustering. In che modo? L'algoritmo non sa né chi siano le persone raffigurate né quante siano. Andrà, però, a cercare tutti i volti nelle fotografie che abbiamo caricato e, successivamente, dopo aver definito una lista di tutti i volti presenti in ogni foto, tramite un algoritmo di clustering cercherà delle somiglianze in questi volti. Alla fine della sua esecuzione l'algoritmo avrà raggruppato le foto dove è presente lo stesso soggetto.

Un altro esempio di utilizzo ricade nell'ambito della sicurezza informatica. Al giorno d'oggi i tipi di attacco conosciuti sono probabilmente solo la punta dell'iceberg.

Ricorrendo, però, a tecniche di clustering possiamo riuscire a individuare e bloccare attacchi tuttora sconosciuti. Si supponga di essere collegati al sito della propria banca che memorizza tutte le operazioni effettuate. Si supponga inoltre, di effettuare quotidianamente delle specifiche operazioni all'incirca alla stessa ora, collegandosi da una determinata località geografica. Tramite il ML non supervisionato queste informazioni potrebbero essere utilizzate per creare dei cluster, in modo che individuino l'identità sulla base delle operazioni. Si supponga ora che un malintenzionato, dall'altra parte del mondo, a un orario differente da quello abituale in cui si svolgono operazioni, riesca ad accedere al profilo bancario. L'algoritmo sarebbe in grado di notare che è in atto qualcosa di anomalo. Nonostante nessuno abbia specificato al calcolatore quali sono le operazioni abituali effettuate dell'utente, il calcolatore, mediante tecniche di clustering, è in grado di riconoscere le operazioni abituali e mandare un messaggio di allarme se individua possibili casi anomali.

3.4.3 Approccio semi-supervisionato

L'approccio semi-supervisionato (o *semi-supervised*), non è un vero e proprio approccio, bensì una tecnica che sta a metà tra le due appena viste: supervisionato e non supervisionato. Questo approccio consiste nel combinare le due tecniche e fornire un risultato basandosi su un input eterogeneo costituito da dati etichettati e dati non etichettati. Questo metodo risulta utile quando si ha una grande mole di dati che vengono etichettati manualmente da utenti specializzati. Nella situazione reale non sempre questo è possibile, proprio perché possono mancare risorse umane competenti o tempistiche adeguate per etichettare tutti i dati. Esistono differenti algoritmi per l'apprendimento automatico mediante un sistema semi-supervisionato:

- *self training*,
- *multi-view training*,
- *self-ensembling*.

Per quanto riguarda il *multi-view training*, si può dire che esso mira a formare diversi modelli con diverse visualizzazioni dei dati. Idealmente queste viste sono complementari e i modelli possono collaborare per migliorare il risultato finale. Queste viste possono differire in diversi modi, ad esempio possono differire nelle funzionalità che utilizzano, nelle architetture dei modelli o nei dati su cui i modelli vengono formati. Il *self-ensembling*, come il *multi-view training*, punta a combinare diverse varianti dei modelli. A differenza di quest'ultimo, però, la diversità nei modelli non è un punto chiave perché il *self-ensembling* utilizza principalmente un singolo modello in diverse configurazioni al fine di rendere più affidabili le previsioni del modello finale. Di seguito

viene descritto l'algoritmo di *self training* che è stato uno dei primi algoritmi a essere sviluppato ed è l'esempio più diretto di come le previsioni di un modello possono essere incorporate nel training del modello. L'algoritmo di *self training* prevede, quindi, di basarsi per quanto possibile su dati che sono stati preventivamente etichettati, avendo anche a disposizione dati non etichettati. Questi ultimi vengono comunque utilizzati, ma in maniera più cauta. Prima di allenare il modello l'algoritmo si concentrerà ad etichettare gli input non ancora etichettati. Come viene spiegato nell'articolo (21), la logica di classificazione dei dati non ancora classificati segue quanto scritto: "Formalmente, l'auto-etichettamento avviene su un modello M avente un insieme L di dati di allenamento etichettati con delle etichette contenute in C e un insieme non etichettato U . A ogni iterazione, per ogni $x \in U$, il modello fornisce delle predizioni su x sotto forma di probabilità $p(x, c)$ ovvero la probabilità che x appartenga alla classe c per ogni $c \in C$. Tra le probabilità appena calcolate, definiamo $P(x, c)$ come la probabilità avente il valore maggiore, allora se P è più grande di una soglia T , x verrà aggiunto a L con l'etichetta c . Questo processo viene ripetuto per un numero fisso di iterazioni o fino a quando non ci sono più dati da etichettare.". (21)

3.4.4 Random Forest Regressor

Relativamente alle soluzioni classiche, sono state analizzate e quindi "allenate" e successivamente "testate" sullo stesso insieme di dati, i classici modelli di regressione come ad esempio Decision Tree Regressor, Random Forest Regressor, Linear Regression, Support Vector Regression ed altri. In seguito ad un'analisi il modello Decision Tree Regressor (in italiano *albero di decisione*) e quindi anche il Random Forest Regressor si sono rivelati decisamente più performanti rispetto a quelli appena citati. L'analisi veloce di fatto è consistita in una valutazione delle performance senza però modificare e quindi senza ottimizzare i parametri che definiscono le caratteristiche del modello.

Successivamente a questa prima scrematura, si è posta l'attenzione sui modelli basati sugli alberi di decisione.

Quando si parla di alberi di decisione si parla di modelli basati su una catena di "decisioni" (da qui il nome del modello), le quali, una volta seguite, portano ad una risposta finale. In generale gli alberi di decisione sono composti da nodi e foglie. I nodi contengono le "condizioni" a cui il dato viene sottoposto per determinare quale sottoalbero di decisioni tale dato dovrà seguire. Nelle foglie dell'albero, invece, si trovano le "risposte" date dalla catena di "decisioni" ad esso collegata. Le risposte possono essere di differente natura a seconda del tipo di problema. Tali tipi di risultati sono di natura categorica oppure di natura numerica.

Nell'immagine 3.5 è mostrato un esempio di un reale albero di decisione allenato e quindi adattato ad uno specifico problema. Il dataset utilizzato per generare questo

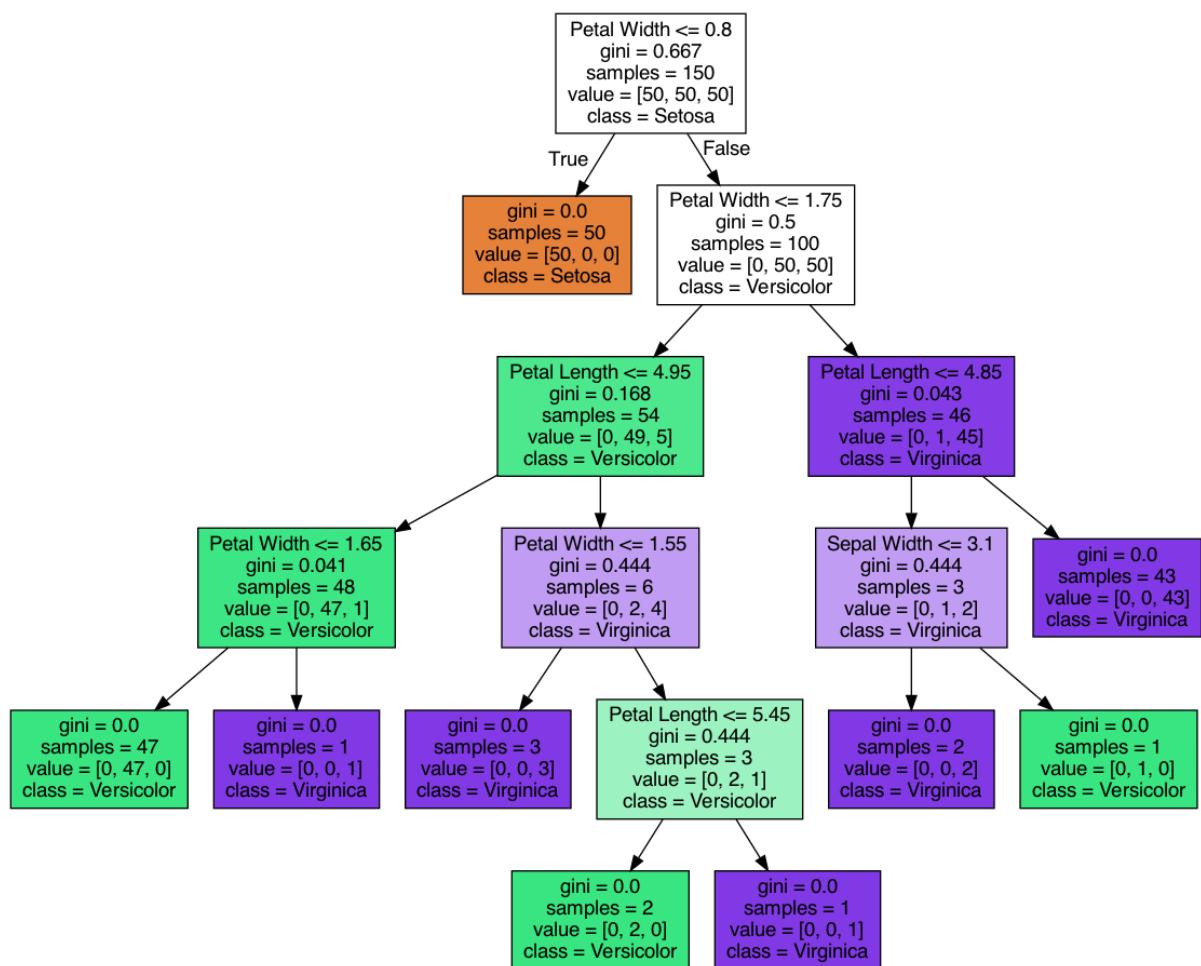


Figura 3.5: Esempio di albero di decisione adattato al dataset iris.

albero è il dataset iris (22) che riporta informazioni relative a 3 tipi di fiori. Esso è quindi composto da una serie di record, che riportano informazioni quali:

- Lunghezza del sepalo;
- Larghezza del sepalo;
- Lunghezza del petalo;
- Larghezza del petalo;
- CATEGORIA DEL FIORE.

In figura è possibile vedere come un albero di decisione lavora per stabilire, a partire da una serie di valori, una classificazione. L'arancione, verde e viola corrispondono rispettivamente a tre categorie distinte di fiore iris: Setosa, Versicolor, Virginica. Ad esempio, considerando solo il nodo radice è possibile determinare, tramite la larghezza del petalo, se il fiore considerato appartenga alla famiglia degli iris Setosa. Nello specifico, questo vale che se "Larghezza del petalo minore o uguale a 0.8" è vero la procedura termina, altrimenti si percorre il sottoalbero di destra e si seguono le condizioni fino ad arrivare ad una foglia la quale conterrà la classificazione di tale fiore. In figura 3.6 è invece mostrato un esempio di albero (non completo) che è stato implementato all'interno del framework FaceQs. In questo caso non è possibile vedere i valori delle foglie a causa della sua dimensione la quale non permette di essere raffigurato nel complesso.

Questo è quanto riguarda gli alberi di decisione. Per quanto concerne i modelli Random Forest Regressor o Random Forest Classifier, diciamo che si basano su quello che viene chiamato *apprendimento d'insieme* (in inglese *ensemble learning*). Con apprendimento di insieme si intende un apprendimento che fa uso di due o più modelli usati congiuntamente per ottenere una migliore prestazione predittiva rispetto ai singoli modelli da cui è costituito. Nell'*ensemble learning* si differenziano 3 categorie, ovvero 3 soluzioni per valutare le predizioni date dai singoli modelli da cui è composto. Tali soluzioni sono:

- *Bagging*: sistema basato su votazione. Ogni modello, all'atto della predizione, voterà circa la propria predizione con pari importanza. Quando tutti i modelli si saranno "espressi", la soluzione avente il maggior numero di voti sarà quella restituita dal modello di apprendimento d'insieme. Nel caso di regressione invece il risultato finale sarà dato dalla media di ogni predizione data da tutti i singoli modelli;
- *Boosting*: analogo al *bagging* con la differenza che ogni modello ha un proprio peso, che varia a seconda delle performance predittive durante la fase di apprendimento. Il risultato sarà, nel caso di regressione, la media ponderata di tutti

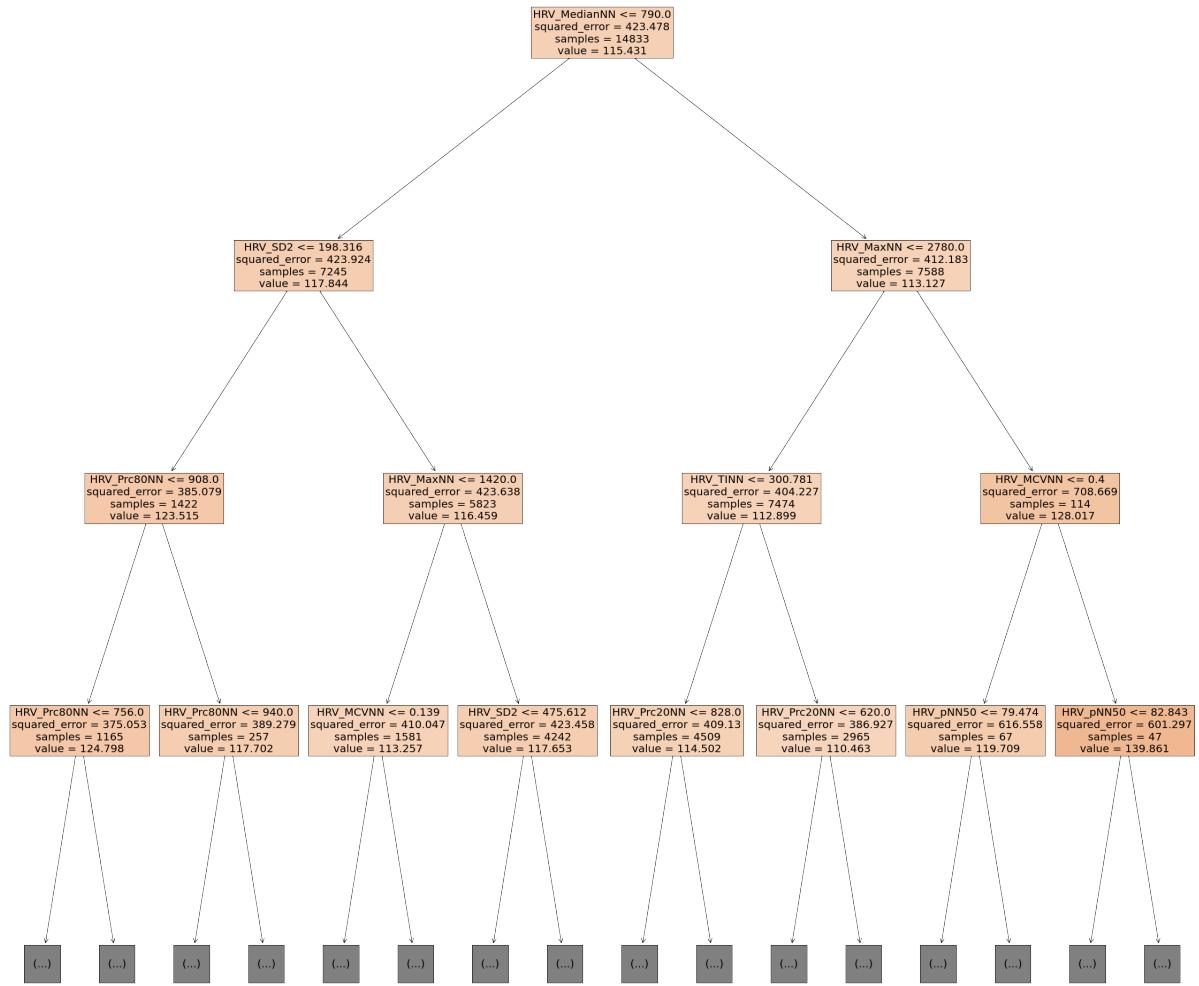


Figura 3.6: Albero implementato realmente dentro FaceQs nella predizione della pressione sanguigna.

i risultati mentre nel caso di classificazione sarà equivalente alla classificazione che avrà il maggior numero di voti ponderati;

- *Stacking*: tecnica che si basa su un meta-classificatore utilizzato per capire come combinare al meglio le predizioni fatte dai singoli modelli.

Nel caso specifico di Random Forest, essendo un modello di ensemble learning basato sugli alberi di decisione, è facile intuire che si tratta di un modello che unisce la potenza di svariati alberi di decisione per fornire una predizione più accurata.

Nell'immagine 3.7 è riportato un esempio di Random Forest composto da più Decision Tree dove è possibile notare che tutti i risultati di ognuno degli albero di decisione convergono in un unico punto, ovvero la funzione che trasforma l'insieme di risultati

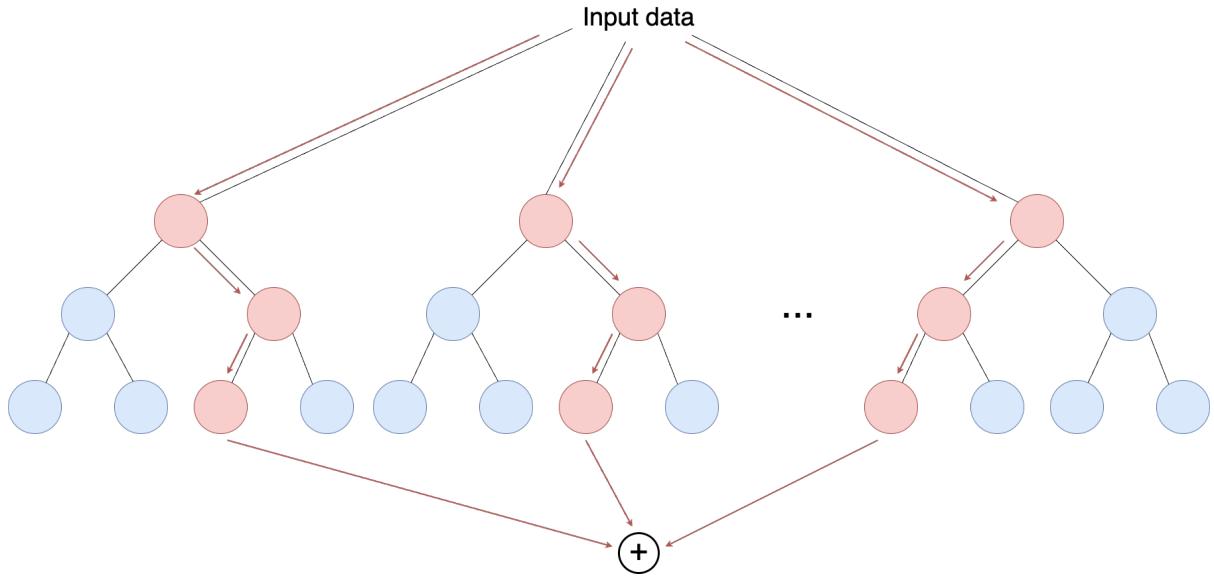


Figura 3.7: Esempio di Random Forest

in un singolo risultato finale. La trasformazione della funzione dipende dalla natura della predizione (quindi categorica o numerica) e dal tipo di tecnica di apprendimento d'insieme applicata.

3.4.5 Neural networks

Le reti neurali rappresentano una soluzione notevolmente più complessa e articolata rispetto ai classici algoritmi di machine learning. Esse hanno l'obiettivo, come qualunque modello di machine learning, di effettuare predizioni di qualunque tipo: si adattano a problemi di regressione (quindi di natura numerica) oppure a problemi di classificazione (quindi di natura categorica).

Queste reti vengono anche dette reti neurali artificiali proprio perché il loro funzionamento tende ad imitare il comportamento della rete dei neuroni biologici del cervello umano. Vengono quindi dette artificiali in quanto replica umana di tale computazione.

Una rete neurale è un grafo diretto $G = (U, C)$ dove U è l'insieme dei vertici e i suoi elementi $u \in U$ sono anche chiamati neuroni. Gli archi invece $c \in C$ sono chiamati connessioni. L'insieme dei nodi U può essere partizionato in tre sottoinsiemi:

- neuroni di input: l'insieme dei nodi che ricevono in maniera diretta l'input,
- neuroni nascosti: l'insieme dei neuroni "nascosti" ovvero che hanno collegamenti solamente con altri neuroni interni alla rete. Essi propagano la computazione verso altri nodi,

- neuroni di output: l'insieme dei neuroni che comunicano con l'esterno e che quindi trasmettono l'output.

Questi insiemi prendono il nome di strati (o *layer*). Questo significa che in una rete esistono 3 diversi tipi di strati:

- strato di input: racchiude tutti i neuroni dell'insieme dei neuroni di input. Ne esiste uno ed uno solo e rappresenta il primo *layer* a cui i dati in input vengono passati;
- strato nascosto: racchiude tutti i neuroni dell'insieme dei neuroni nascosti. Ne possono esistere uno, nessuno oppure un qualunque numero positivo. Essi rappresentano gli strati che caratterizzano nella maggior parte dei casi le reti neurali;
- strato di output: racchiude tutti i neuroni dell'insieme dei neuroni di output. Ne esiste uno ed uno solo e rappresenta l'ultimo *layer* da cui passano i dati propagati all'interno della rete. Da questo strato vengono inoltrati all'esterno i dati computati.

In generale si definiscono reti neurali profonde quelle reti formate da x strati nascosti (con $x > 1$). Le reti senza o al più con un solo strato nascosto vengono dette reti neurali semplici.

Nelle reti ad ogni connessione (u, v) viene assegnato un peso (solitamente definito w) che definisce l'importanza del dato inoltrato dal neurone u al neurone v . Ogni neurone $u \in U$ porta con sè quattro variabili fondamentali: il *network input* net_u , la *activation* act_u , l'*output* out_u e l'*external input* ext_u . Le prime tre variabili sono computate durante la costruzione e apprendimento della rete da funzioni dedicate:

- *network input function* f_{net}^u : rappresenta la funzione che prende in input gli output dei neuroni precedenti (o l'input esterno in caso di neuroni appartenenti all'insieme dei neuroni di input) correlati al proprio peso w e restituisce un valore che viene poi processato dalla funzione di attivazione,
- *activation function* f_{act}^u : funzione che prende in input il risultato dalla *netowrk input function* e determina se il neurone venga attivato o meno,
- *output function* f_{out}^u : funzione che genera l'output effettivo del neurone a seconda che questo sia attivato o meno.

Esistono diversi tipi di reti neurali, ognuna delle quali offre i propri vantaggi (e svantaggi) ed è in grado di affrontare uno specifico tipo di problema. Alcune di queste reti sono ad esempio: *feed forward neural network*, *recurrent neural networks*, *convolutional*

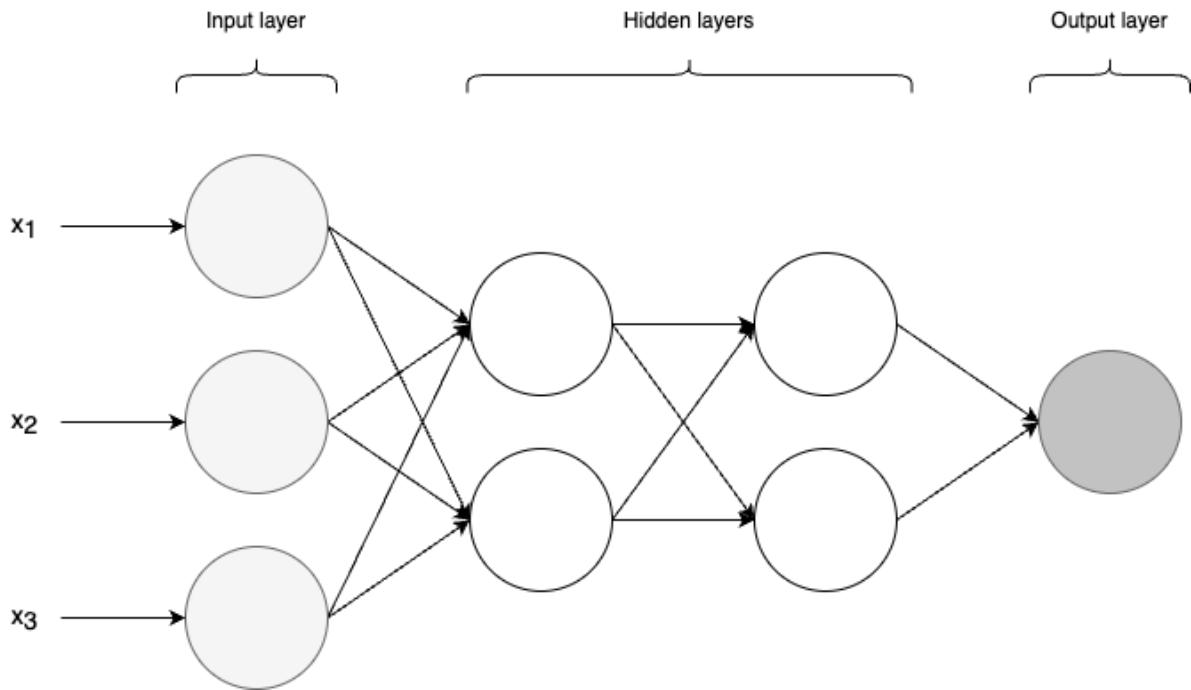


Figura 3.8: Esempio di rete neurale densa con due strati nascosti.

neural network, *graph neural networks* ed altre. Nel primo caso, si definiscono reti *feed forward* quelle reti che sono definibili come grafi aciclici e quindi reti in cui l'input può percorre la strada a partire dall'insieme dei neuroni di input fino ad arrivare all'insieme dei neuroni di output. Nel secondo caso si parla di reti in grado di rispondere a classici problemi come ad esempio il riconoscimento di immagini oppure di problemi di regressione su serie temporali. Nel caso delle *recurrent networks*, come dice il nome stesso, il percorso può essere ricorrente, ovvero sono reti definibili come dei grafi ciclici. Esse trovano il proprio utilizzo nei problemi la cui risposta dipende dal contesto corrente. Vale a dire che per risolvere problemi di questo genere è necessaria una componente di memoria (che è data dalla ricorsività della rete) che permette alla rete di fare predizioni in base agli stati precedenti correlati a quello corrente ed ai vari input.

Durante lo sviluppo di questa tesi, è stato sperimentato uno specifico tipo di reti neurali: *Dense Neural Networks*. Esse sono reti neurali (che possono essere o non essere profonde) caratterizzate dal fatto che tutti gli strati sono definiti "densi". Gli strati "densi" sono interamente connessi con lo strato precedente. Ad esempio, presi due strati densi, entrambi composti da 3 neuroni, il numero totale di collegamenti equivale a $3 \times 3 = 9$ connessioni. In figura 3.8 è possibile vedere una schematizzazione di una rete neurale densa.

In figura 3.9 è mostrata uno schema dell'architettura della rete realmente implementata in FaceQs. Nello specifico è stata utilizzata una rete della stessa architettura per tutti gli esperimenti riportati in questa tesi. Tale rete prevede 5 layer di 256, 128, 32, 32 e 1 neuroni rispettivamente. Ad ognuno di essi, escluso l'ultimo, è stato attacca-

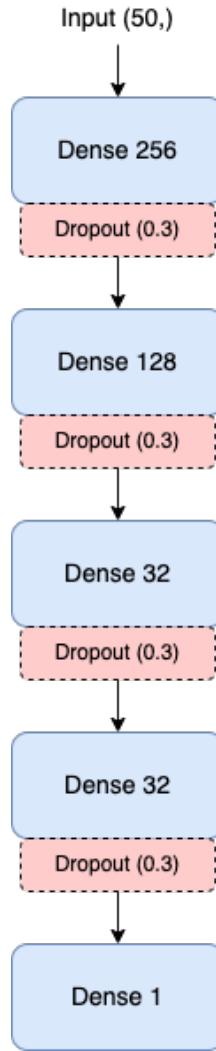


Figura 3.9: Rete neurale densa implementata all'interno di FaceQs.

to un layer di *Dropout* (23). Tale layer permette di evitare l'overfitting durante la fase di allenamento della rete. Infine l'ultimo layer è composto da un unico neurone il cui risultato sarà la stima della pressione (diastolica, sistolica o arteriosa).

3.5 Post-processing applicato al segnale stimato

Come ultimo passaggio nella *pipeline*, avviene uno smoothing del segnale generato dai due modelli presi in esame, Random Forest o Deep Neural Network, mediante calcolo della media mobile. Questo consente di ottenere una migliore leggibilità qualitativa dei risultati e una più ragionevole misura quantitativa, generalmente data in termini di MAE.

Fissata una finestra di dimensione W punti del segnale, si effettua la media di tutti i punti all'interno di essa e si fa traslare la finestra di una punto alla volta, fino alla fine del segnale. Le finestre risulteranno pertanto sovrapposte con uno stride pari a

uno. Per tutti gli elementi di indice minore o uguale a W la media viene calcolata sulla finestra di dimensione $i - 1$ dove i rappresenta l'indice dell' i -esimo campione (con i chiaramente inferiore a W).

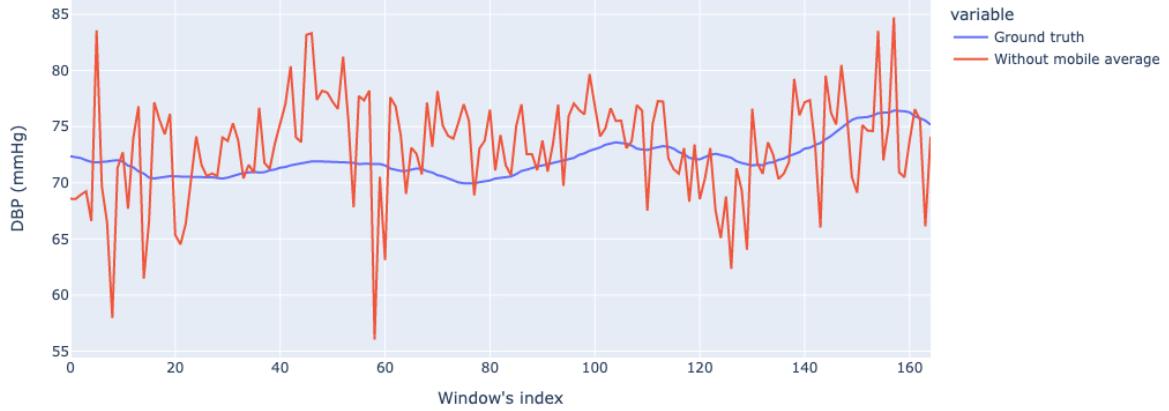


Figura 3.10: Esempio di stima di pressione diastolica di un video di test senza applicare la funzione di *post_processing*.

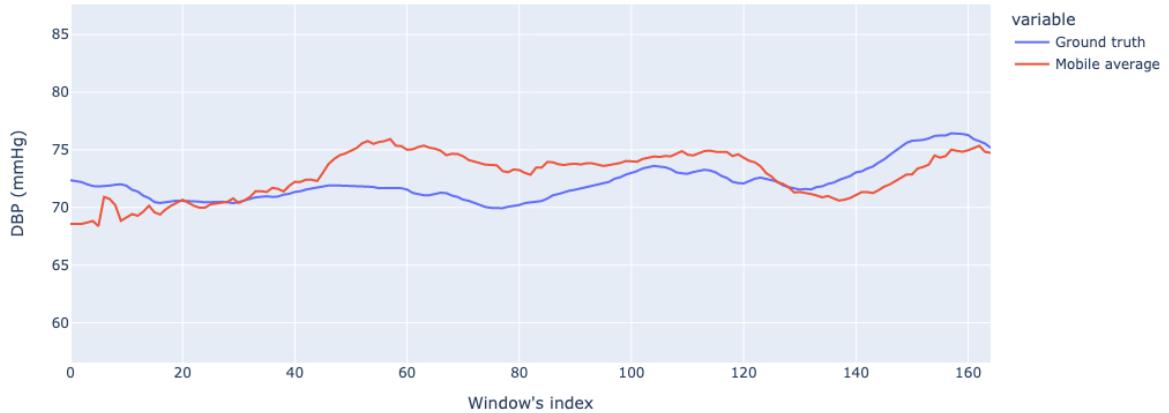


Figura 3.11: Esempio di stima di pressione diastolica di un video di test con l'applicazione della funzione di *post_processing*.

In figura 3.10 è riportato un esempio di predizione del segnale senza computazione della media mobile. Il segnale blu, rappresenta la predizione del modello (in questo caso Random Forest) mentre la linea rossa rappresenta il segnale reale (anche detto *ground truth*) della pressione diastolica del paziente. Il segnale blu denota una somiglianza con il grafico rosso nell'andamento di quest'ultimo. In ogni caso però il segnale predetto resta di difficile lettura.

Nella figura 3.11 è riportato il segnale *ground truth* (GT) del grafico precedente comparato con la media mobile estratta dal segnale predetto dal modello. Nello specifico il segnale blu è la media mobile a finestra di dimensione 25, ovvero di dimensione pari a quella degli fps del video. Il segnale rosso, come nel grafico riportato sopra, rappresenta l'andamento reale della pressione diastolica del soggetto.

Capitolo 4

Esperimenti e risultati

In questo capitolo si riportano i risultati sperimentali ottenuti applicando i modelli proposti nel capitolo precedente, Random Forest Regressor e Deep Neural Network, su benchmark di riferimento. In particolare, sono stati impiegati tre diversi dataset MIMIC-II (anche se solo in piccola parte) (24), VitalDB (25) e Vision for Vitals (V4V) (26) che offrono, rispettivamente nei primi due casi, un segnale fotopletismografico fornito tramite sensore posizionato sul dito correlato alla pressione sanguigna e nell'ultimo caso dei video di diversi soggetti correlati anch'essi alla pressione sanguigna. Nonostante l'obiettivo della tesi sia quello di analizzare la pressione sanguigna a partire da un video come impone la tecnica basata sul segnale fotopletismografico remoto, cioè ricavato da video, sono stati utilizzati due dataset che offrono anche un segnale PPG "puro" (GT), in modo tale da validare la tecnica qui applicata su un segnale fotopletismografico stimato. In questo modo è possibile mostrare in maniera più completa, in base ai risultati ottenuti tramite un segnale PPG, a quale precisione massima si può spingere la tecnica studiata in questa tesi.

Inoltre, è da sottolineare, che come tutti i dataset pubblici disponibili in rete, la distribuzione dei dati al suo interno favorisce i soggetti normo-tensivi, rendendo così più difficile generare modelli di apprendimento (ad alte performance) in grado di stimare la pressione anche a soggetti che soffrono di problemi cardiovascolari. Proprio per questo motivo, quando verranno mostrati i risultati, sarà mostrata la distribuzione dell'errore sui casi di test correlata al tipo di soggetto (ipotensivo, normo-tensivo e ipertensivo).

Verranno descritti inoltre tutti i risultati, commentandoli e traendo infine delle conclusioni su entrambe le metodologie applicate.

4.1 Datasets

4.1.1 V4V dataset

Vision for Vitals (V4V) è un dataset che racchiude video raffiguranti il volto dei soggetti correlato alla misurazione della pressione sanguigna (espressa in mmHg), della frequenza respiratoria ed altre misurazioni. All'interno del dataset sono presenti 140 soggetti dell'università di Binghamton. Di questi 140, 58 sono maschi e i restanti 82 femmine la cui età va da un minimo di 18 anni fino ad un massimo di 66. Per ogni soggetto sono presenti molteplici video in diverse inquadrature ed espressioni facciali. In totale sono presenti 724 video correlati di *ground truth*. Purtroppo, quelli utilizzabili sono al più 488 e al minimo 182. Questo è dovuto al fatto che i video sono troppo corti oppure pyVHR non è in grado di analizzarli. Il numero effettivo di video utilizzabili dipende dal tipo di configurazione del modello allenato, ma questo viene spiegato nella prossima sezione.

I soggetti inclusi nel dataset sono di diverse etnie: neri, bianchi, asiatici, ispanici ed altri. La registrazione dei segnali fisiologici è stata effettuata mediante lo strumento BIOPAC MP150. Nello specifico la pressione è stata registrata mediante Biopac NIBP100D che è composto da un sensore posizionato sul dito collegato ad un bracciale gonfiabile. La frequenza respiratoria, invece, è stata rilevata mediante una fascia posizionata all'altezza del busto in grado di rilevare i cicli respiratori (espressa in battiti per minuto). Entrambe le misurazioni sono state effettuate con una frequenza pari a 1000 Hz. All'interno di questa tesi, con il nome "V4V" verrà definito questo dataset.

4.1.2 MIMIC-II dataset

Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC) II è un dataset ricco dei principali segnali fisiologici. Consiste in migliaia di segnali registrati dai monitor a lato del letto del paziente. Sono stati raccolti dati dal 2001 al 2008 in vari ospedali. Il dataset completo racchiude diversi tipi di segnali ma nello studio di questa tesi ne è stata utilizzata solo una piccola parte, comprendente segnali ECG, PPG e ABP (24) ognuno dei quali è stato campionato con una frequenza pari a 125 Hz. Il dataset, per semplicità di utilizzo, è stato suddiviso in diversi file ognuno dei quali contiene circa 3000 record. Ogni record corrisponde ad una misurazione fatta su un paziente che comprende i tre segnali appena citati. È possibile che per uno stesso paziente siano state effettuate più misurazioni e di conseguenza siano presenti più file per ognuno di essi. Purtroppo non è possibile distinguere i pazienti l'uno dall'altro ma in ogni caso tutte le misurazioni di uno stesso paziente si trovano una dopo l'altra (all'interno del file), il che significa che la fase di allenamento e la fase di testing rischiano di avere

dati appartenenti allo stesso soggetto. Per questo dataset il nome utilizzato nella tesi sarà "CufflessDB".

4.1.3 VitalDB dataset

VitalDB dataset (25) è un dataset ricco di molteplici segnali fisiologici tra cui il segnale fotopletismografico e quello della pressione sanguigna. Esso contiene 6.388 registrazioni di diversi pazienti sottoposti a chirurgia non cardiaca (generalmente, toracica, urologica e ginecologica). I segnali sono registrati con una frequenza pari a 500 Hz e hanno una durata media di circa sessanta minuti.

Essendo una durata eccessiva e non processabile di ogni segnale (in questo caso PPG e di pressione sanguigna) sono stati estratti coerentemente 150 secondi con una frequenza ridotta a 75 Hz. La scelta di 150 secondi è dovuta al fatto che generalmente un video-selfie ha una durata massima di questa misura, una durata maggiore sarebbe eccessiva. Per quanto riguarda la riduzione della frequenza, è stata applicata una analoga riduzione: le telecamera di uno smartphone o laptop raramente superano i 60 fps (ovvero la frequenza di campionamento ottenibile tramite pyVHR).

Queste scelte sono basate sul fatto che prendendo dei segnali quanto più simili possibile si possono ottenere comparazioni coerenti. Questo dataset verrà indicato successivamente con "VitalDB".

4.2 Applicazione dei modelli

Nella prima parte di questa sezione viene descritta la fase di pre-processing dei dati, successivamente viene mostrato come è stato effettuato l'allenamento dei modelli ed in fine i risultati sperimentai ottenuti sui dataset considerati.

4.2.1 Preparazione dei dati

In primo luogo è stato necessario estrarre le informazioni di interesse per questo progetto: il segnale sistolico, diastolico e arterioso. Questi tre indici legati alla pressione sanguigna sono stati estratti dal segnale della pressione continuo. È stata implementata una funzione ad hoc che, con i corretti parametri, è in grado di identificare i picchi sistolici e diastolici per ogni ciclo di pressione sanguigna. In figura 4.1 è mostrato un esempio di estrazione del segnale sistolico (tramite croci rosse) e diastolico (tramite croci verdi).

Questa estrazione è stata fatta per tutti i dataset presi in considerazione. Per quanto riguarda V4V sono stati estratti, tramite pyVHR, i segnali fotopletismografici finestrati tramite i metodi patch e olistico. Nel caso dei dataset CufflessDB e VitalDB non è stato

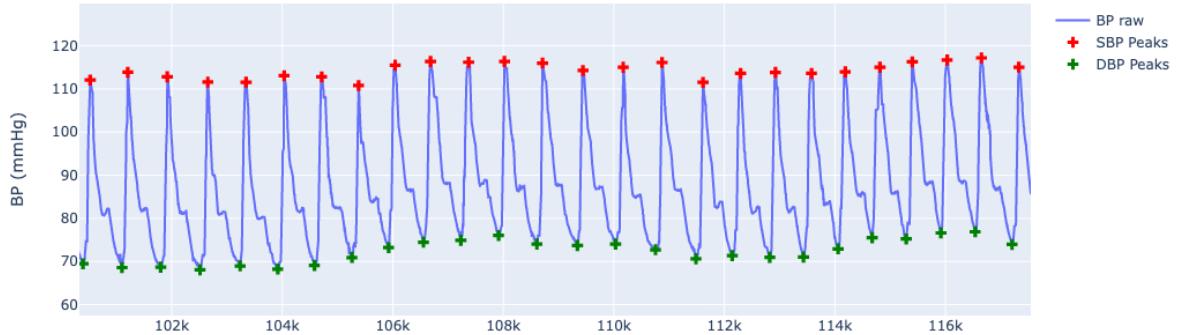


Figura 4.1: Estrazione di segnale sistolico (croci rosse) e diastolico (croci verdi).

necessario estrarre il segnale fotopletismografico in quanto già presente all'interno del dataset.

Successivamente, per ogni dataset, è stato applicato lo stesso procedimento descritto di seguito. Per ogni file sono stati presi i segnali rPPG (o PPG) e il relativo segnale di pressione sanguigna e sono stati finestrati. Ogni finestra del segnale rPPG (o PPG) è stata sottoposta ad un'analisi morfologica tramite NeuroKit2. Con quest'ultimo è stata generata, per ogni finestra, una lista di indici che sono stati messi in correlazione alla media, calcolata sulla stessa finestra, del segnale sistolico, diastolico e arterioso. In figura 4.2 è raffigurato un esempio di finestratura. Sono mostrati i segnali di PP-G/rPPG e di BP entrambi finestrati (tramite i riquadri tratteggiati) ad una dimensione fissata (in questo caso 10 secondi). Ad esempio, i dati generati tramite NeuroKit2 della finestra azzurra, sono messi in correlazione con la media di tutte le misurazioni della pressione contenute all'interno della stessa finestra azzurra (ovvero per la sistolica viene calcolata la media delle croci rosse, per la diastolica la media delle croci verdi e per la arteriosa la media dei punti definiti tramite l'apposita formula).

Le dimensioni delle finestre, come descritto nel capitolo 3, sono misurate in secondi: 20, 30, 40, 50 e 60. Come estremo inferiore è stato scelto 20 secondi perché NeuroKit2 non è in grado di analizzare un segnale fotopletismografico di dimensioni inferiori. Come estremo superiore si è deciso di limitare a 60 secondi perché rappresenta la dimensione massima per un uso pratico sensato (i video selfie di dimensioni maggiori di 90 secondi sono considerate eccessive).

Infine, una volta eseguiti i calcoli sulle finestre dei vari segnali sono stati generati dei *dataframe* contenenti tanti record quante sono le finestre applicabili su quello specifico segnale.¹ Le prime N colonne del *dataframe* corrispondono quindi ai dati

¹In ogni caso, sia per video di V4V o per segnali PPG di CufflessDB o VitalDB le lunghezze del

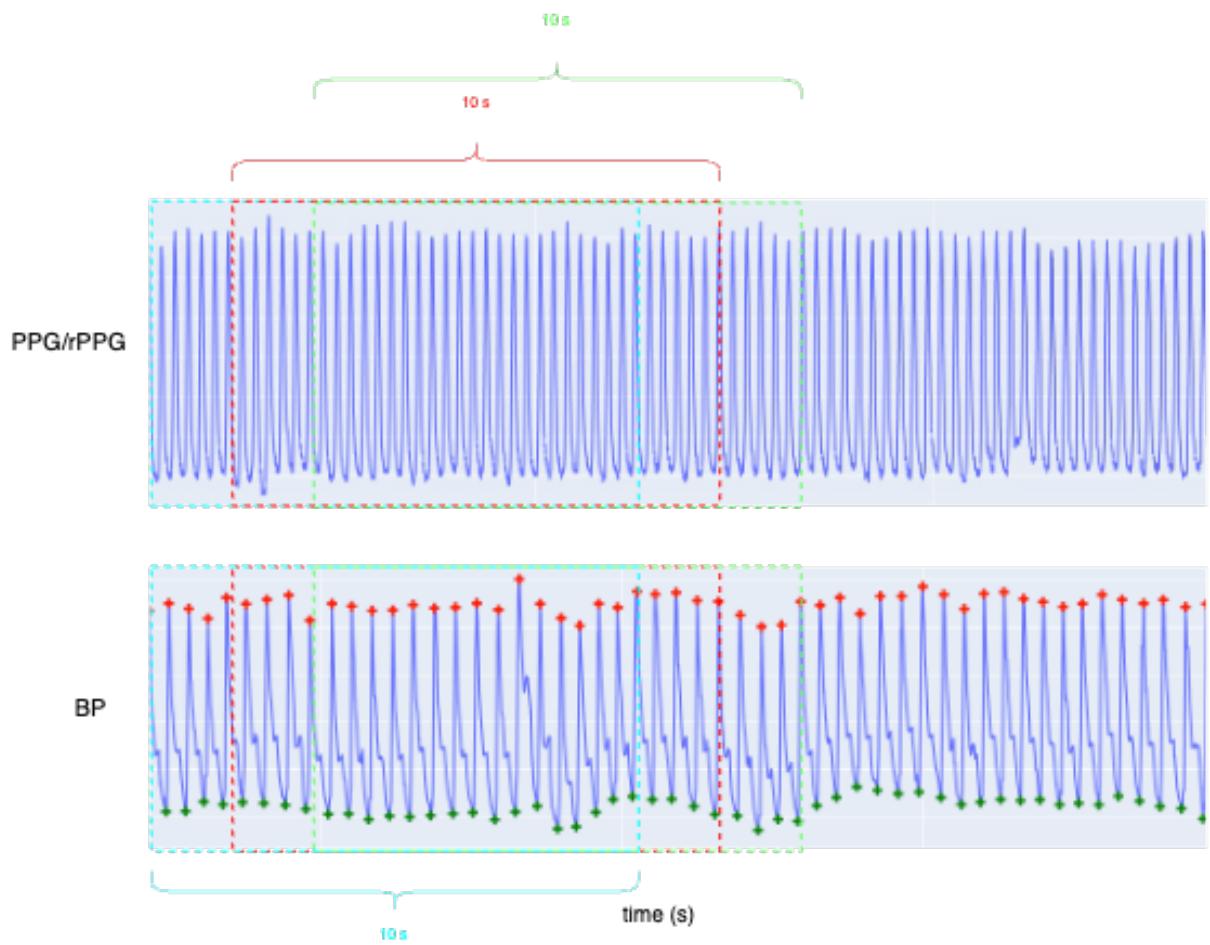


Figura 4.2: Esempio di finestratura del segnale PPG/rPPG coerentemente con il segnale BP.

Finestra (sec)	Video totali	Train	Validation	Test
20	488	440	44	48
30	356	321	33	35
40	268	242	25	26
50	220	198	20	22
60	182	164	17	18

Tabella 4.1: Divisone dei video in base alla finestratura.

restituiti da NeuroKit2, le ultime tre ai valori medi della pressione sistolica, diastolica e arteriosa.

4.2.2 Fase di training

Una volta elaborati i vari segnali *raw* e generati di conseguenza i *dataframe*, è iniziata la fase di training e valutazione del modello. In tutti e tre i dataset è stata applicata la stessa divisione tra dati di train, validation e test. Per la fase di training (e validazione) sono stati utilizzati il 90% di video/registrazioni (di cui 10% di training per la validazione) e il restante 10% per il test.

Essendo che il reale obiettivo di questa tesi è analizzare la pressione sanguigna a partire da un segnale video e non da un segnale fotopletismografico, tutte le scelte sono dipese dal dataset V4V per mantenere quanto più possibile comparabili gli esperimenti. Infatti, la divisione dei dati in questa forma è giustificata dalla ridotta quantità di dati utilizzabili nel dataset V4V. Di conseguenza le scelte per gli altri dataset (relativamente alla divisione dei dati e non solo) sono le stesse prese per V4V. In tabella 4.1 sono mostrati in dettaglio i video utilizzati per la fase di training, validation e test in base alla finestratura dei video.

In generale, per ogni *dataframe* generato, è stata applicata una pulizia dei dati per permettere ai modelli di utilizzarli. Da questi *dataframe* sono stati rimossi tutti i campi che presentano valori diversi da un numero reale (ad esempio quelli che assumono il valore "not a number" oppure "infinito"). Inoltre sono stati rimossi i *dataframe* che presentano misurazioni della pressione sanguigna al di fuori dai valori "normali". Infatti per la pressione diastolica sono stati accettati i valori compresi tra 40 e 110 mmHg, mentre per la sistolica tra i 70 e 160 mmHg.

4.2.3 Selezione dei modelli

I modelli utilizzati per rispondere al problema del calcolo della pressione sanguigna tramite video, ovvero il Random Forest Regressor e le Deep Neural Network sono si-

video/segnale sono sempre coerenti con il relativo segnale della pressione.

stemi complessi che, come ogni modello nel mondo del machine learning, necessitano di una accurata selezione degli iperparametri. Gli iperparametri sono quell’insieme di dati forniti dall’utente che influiscono sul design del modello (che sia di machine learning o che sia di una rete neurale). Tali iperparametri si differenziano dai parametri in quanto questi ultimi sono valori che vengono automaticamente generati dal modello stesso e di conseguenza l’utente non ha nessuna influenza su di essi.

Ad esempio, nel caso di Random Forest, sia nella versione di classificatore che di regressore, ci sono svariati iperparametri che possono essere considerati per generare una differente istanza del modello. Quelli considerati durante questa ricerca sono (27):

- *n_estimators*: rappresenta il numero di alberi che costituiscono la foresta,
- *max_depth*: rappresenta la profondità massima che può essere raggiunta dagli alberi,
- *max_features*: rappresenta il numero di features del dato in input da considerare quando si ricerca il migliore split,
- *min_samples_split*: il numero minimo di samples richiesti per la divisione di un nodo,
- *min_samples_leaf*: il numero minimo di samples richiesti per essere un nodo foglia,
- *bootstrap*: un iperparametro che ammette due valori (*true* o *false*), dove viene determinato se applicare il principio di bootstrap aggregating, ovvero per ogni differente modello pescare casualmente un differente sottoinsieme del dataset oppure se considerare l’intero dataset.

Nel caso di reti neurali profonde gli iperparametri sono sempre un punto critico fondamentale che comprende l’esecuzione di diversi esperimenti alla ricerca dei migliori risultati.

4.2.4 Randomized search, grid search e cross-validation

La ricerca dei migliori iperparametri è una fase tanto importante quanto può essere lungo il tempo richiesto per portarla a termine. È importante che venga fatta cercando di considerare il più ampio spazio possibile di combinazioni di iperparametri in modo tale da non focalizzarsi su un minimo locale. Questo è dovuto al fatto che se ci si concentra su un intervallo relativamente piccolo fin dall’inizio, il migliore modello trovato a partire da quello specifico intervallo di combinazioni di iperparametri, non potrà garantire che sia un modello che minimizza (o massimizza, a seconda del tipo

di problema) il risultato in maniera globale rispetto a tutte le possibili alternative di iperparametri.

Si supponga che l'allenamento di una combinazione richieda una decina di secondi. Se noi avessimo una griglia di combinazioni della dimensione $5 * 50$, significherebbe avere 50^5 possibili combinazioni, ovvero 312.500.000 diverse combinazioni! Se moltiplichiamo questo valore per i dieci secondi richiesti da ogni allenamento del modello si avrebbe un tempo totale necessario per individuare il modello migliore superiore a 3 miliardi di secondi, ovvero oltre 36.168 giorni, vale a dire un centinaio di anni! Quindi per risolvere questo problema garantendo (quanto più possibile) un ampio spazio di combinazioni di iperparametri viene utilizzata la tecnica chiamata *randomized search* (o ricerca casuale). Tale tecnica, come suggerisce il nome, è una tecnica che si basa sulla casualità per la ricerca del migliore modello. In generale questa tecnica prevede che l'utente definisca una griglia di dimensioni come quelle definite sopra e successivamente determini un numero x tale che $x \ll N$ con N pari al numero totale di combinazioni. Questo numero è un parametro fondamentale che permette di determinare quante combinazioni selezionare da tutte quelle possibili. Relativamente a tale tecnica è importante precisare che garantisca una distribuzione tale da coprire (quanto più possibile) lo spazio totale delle combinazioni possibili.

Questa tecnica, quando si effettua una ricerca del modello che ottimizza il risultato, è una prima fase molto importante, ma non l'unica. Successivamente a questa segue una ricerca del modello con una "granularità" più fine, ovvero quella chiamata *grid search* (o *ricerca su griglia*). Tale tecnica, analogamente a quanto fatto da *random search*, prevede che venga definita una griglia di iperparametri. A differenza della tecnica sopra descritta, la *grid search* analizzerà ogni possibile combinazione, quindi è importante che il numero di combinazioni selezionate sia in numero ridotto. Questa tecnica molto spesso segue la *randomized search* proprio perché con quest'ultima si individua quell'intervallo di valori per ogni iperparametro che permette di ottenere la migliore predizione. Successivamente, tramite la *grid search* si effettua una ricerca con "granularità" molto più fine a partire da intervalli basati su un intorno vicino di quelli individuati tramite la ricerca casuale.

Nell'immagine 4.3 è mostrata una pipeline di tuning di un modello generico, ovvero una sequenza di operazioni che mostra quanto appena descritto. L'input è rappresentato dalla tabella relativa dove sono definiti n iperparametri con le relative liste di valori che tali iperparametri possono assumere. Successivamente alla *randomized search* i valori selezionati come ammissibili si riducono ad un insieme molto ristretto per ognuno degli iperparametri. Vengono quindi selezionati, per ogni iperparametro, i valori nell'intorno del valore individuato dalla ricerca casuale. Infine, tramite la *grid search*, si ricerca nel dettaglio la migliore combinazione fino ad ottenere il modello ottimizzato (o in inglese *tuned*) avente un solo specifico valore per ogni iperparametro.

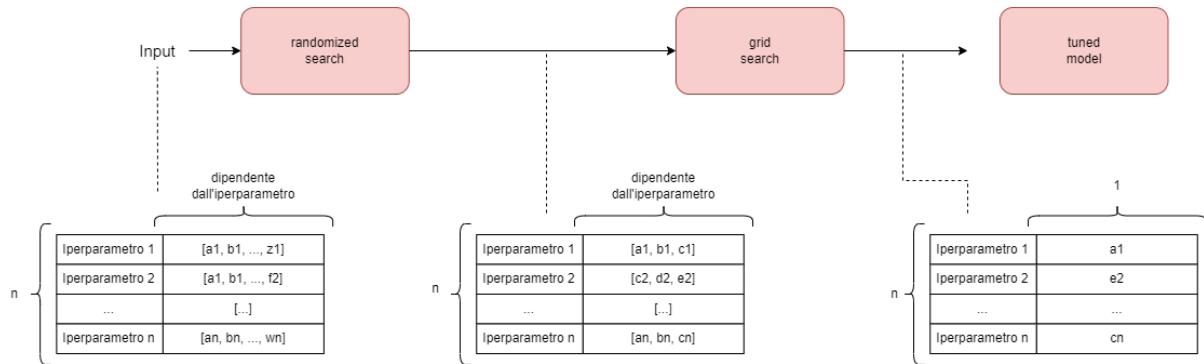


Figura 4.3: Pipeline utilizzata per la ricerca del modello migliore

Quanto precede è ciò che riguarda la selezione della migliore combinazione di iperparametri del modello. Tuttavia quando si può dire che un modello è migliore di un altro? Una semplice risposta potrebbe essere quella di scegliere un modello, allenarlo e poi testarlo e vedere quali sono i risultati rispetto all'insieme di dati dei test. In effetti questa procedura non contiene errori, bensì è una tecnica poco affidabile per garantire che tale modello abbia le performance riportate nella predizione dell'insieme di testing. Questo è dovuto al fatto che quando si genera un modello, oltre agli iperparametri è necessario considerare un ulteriore ingrediente fondamentale. Tale componente è l'insieme di dati utilizzati nella fase di allenamento e di test. Essi giocano un ruolo fondamentale perché sono i dati su cui il modello si basa per le predizioni di dati ad esso sconosciuti. È quindi importante, nella valutazione di un modello, fissati gli iperparametri, che vengano considerate le versioni di quest'ultimo allenate con varie combinazioni di dati di allenamento e di test. Questo processo viene denominato *cross-validation*. Con la *cross-validation* i dati dell'intero dataset vengono partizionati in k gruppi aventi approssimativamente la stessa grandezza, ognuno dei quali è chiamato fold e dove k rappresenta un numero specificato dall'utente. Ad esempio, per un valore k pari a cinque si ha una suddivisione del dataset in cinque diversi fold. Effettuata la suddivisione il modello viene allenato utilizzando il primo fold come test set e i restanti come training set. Successivamente lo stesso modello viene costruito usando il secondo fold come test set e i restanti fold (1, 3, 4, 5) come training set. Questo procedimento è ripetuto utilizzando ogni fold come test set una e una sola volta. Pertanto avremo 5 ripetizioni del dataset in ciascuna delle quali il test set è costituito da un diverso sottoinsieme del dataset stesso. In figura 4.4 è mostrato uno schema di una cross-validation a 5 fold.

L'utilizzo di questa strategia procedurale comporta diversi vantaggi. Un primo importante vantaggio è rappresentato dal fatto che l'accuratezza del modello è calcolata sulla media di tutti gli score ottenuti dall'utilizzo dei differenti test set. Per capire meglio questo vantaggio, vediamo l'esempio seguente. Si supponga di essere "fortunati" e quindi di allenare il modello su un training set che contiene perlopiù dati di difficile

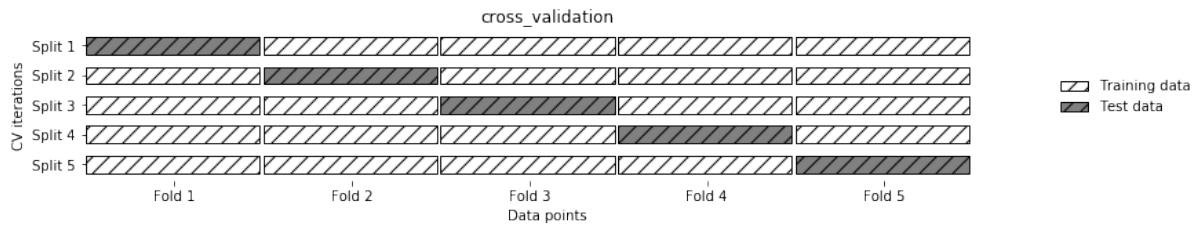


Figura 4.4: Esempio di cross-validation a 5 fold

classificazione (outlier). Di converso il test set avrà probabilmente dati di facile classificazione il che permette al modello di avere prestazioni casualmente molto alte. Al contrario nel caso "sfortunato" si avranno i dati di difficile classificazione all'interno del test set e quelli di semplice classificazione nel training set. In questo modo avremo performance piuttosto basse ancorché aleatorie. Con l'utilizzo di cross-validation però, la performance complessiva sarà difficilmente affetta da una distorsione casuale poiché il risultato finale sarà composto dalla media dei risultati delle varie combinazioni. Un altro vantaggio corrisponde al fatto che oltre a variare l'utilizzo dei dati di allenamento è possibile variare anche il numero di fold che suddividono il dataset. Se, ad esempio, vengono utilizzati dieci fold, si avrà un test set definito da un decimo della dimensione totale del dataset. Analogamente se ne utilizziamo cinque il test set sarà costituito da un quinto del totale. D'altra parte la cross-validation presenta lo svantaggio di allungare i tempi di valutazione del modello. Questo perché maggiore è il numero di fold, maggiore è il tempo richiesto per ottenere un risultato. In altre parole con k fold avremo un tempo pari a k volte il tempo necessario ad allenarlo.

4.3 Risultati ottenuti

In questa sezione finale della tesi sono mostrati i risultati tramite l'uso di differenti tipologie di grafici per capire e dare una migliore visione dei risultati nel complesso. Nello specifico vengono analizzati i risultati ottenuti su entrambi i modelli (RF e DNN) utilizzati per ogni finestratura sul dataset V4V. I risultati sono comprensivi di tutte le finestrature citate sopra per entrambi i modelli con entrambi i metodi di estrazione del segnale rPPG (quindi per il metodo olistico e per il metodo patch).

Questo significa che sono stati generati 20 modelli (10 Random Forest e 10 Deep Neural Network) per ogni tipo di misurazione della pressione (sistolica, diastolica e arteriosa). Anche per i dataset CufflessDB e VitalDB sono stati generati i modelli RF e DNN per le finestrature di 30 e 40 secondi su tutte e tre le misurazioni possibili. In figura 4.5 è raffigurato uno schema riassuntivo dei modelli generati per l'analisi di V4V mentre in figura 4.6 è raffigurato lo schema riassuntivo dei modelli generati per l'analisi di entrambi i dataset PPG (VitalDB e CufflessDB).

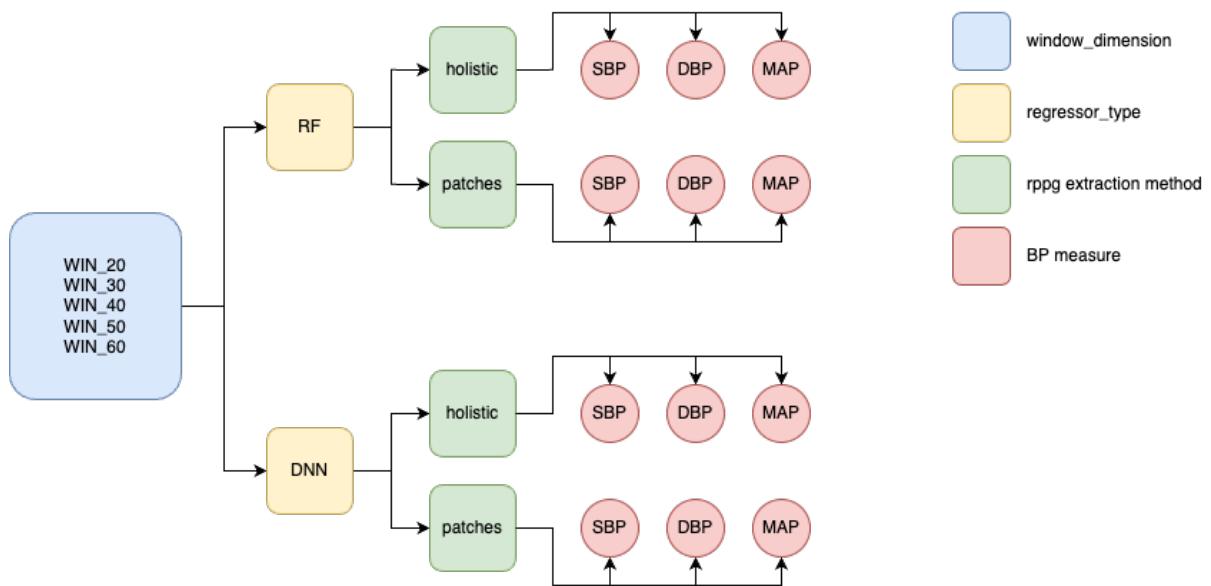


Figura 4.5: Schema dei modelli generati per il dataset V4V.

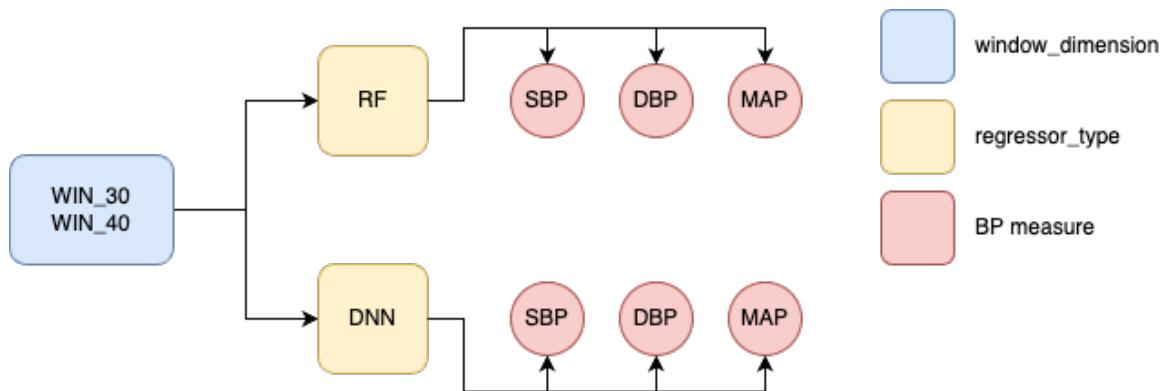


Figura 4.6: Schema dei modelli generati per il dataset VitalDB e CufflessDB.

4.3.1 Risultati tramite segnale PPG

Di seguito vengono presentati i risultati generati dall'analisi del segnale PPG GT. Sono quindi presentati i risultati (su grafici) ottenuti applicando i modelli RF e DNN sui dataset VitalDB e CufflessDB con finestre di 30 e 40 secondi rispettivamente che generalmente forniscono risultati migliori rispetto alle altre.

In entrambi i dataset, specialmente in VitalDB, è ben definita la presenza di soggetti normo-tensivi e questo, come viene mostrato più avanti, permette di avere una performance del modello notevolmente più alta. Nel caso di CufflessDB i pazienti registrati hanno una pressione sanguigna media la cui distribuzione risulta più omogenea con una elevata quantità di soggetti ipertensivi e normo-tensivi. Questa peculiarità della distribuzione permette di mostrare come l'errore generale (inteso come MAE medio) tenda a ridursi.

In figura 4.7 sono mostrati due grafici riassuntivi dell'errore sui casi di test del dataset VitalDB. Nel grafico superiore è mostrata la distribuzione dell'errore in vari box plot raggruppati in gruppi da 10 mmHg. Nello specifico ogni gruppo rappresenta l'insieme dei soggetti con una pressione (diastolica) media di un valore compreso tra x e $x + 10$ dove x è il valore presente sull'asse delle ascisse. Ogni punto affianco ai vari box plot, corrisponde ad un caso di test. Si noti che per ogni intervallo vengono messe a confronto le performance di RF con DNN. Sull'asse delle ordinate è presente il MAE su uno specifico caso di test espresso in mmHg. Nel grafico inferiore invece è presente la distribuzione dei casi di training (identica per RF e DNN), che denota come al crescere dei casi usati per l'apprendimento su uno specifico intervallo, l'errore decresca e viceversa. Infatti è possibile notare, che nel caso di soggetti normo-tensivi, ovvero con una pressione diastolica compresa tra 50 e 70 mmHg si ha l'errore medio più basso. Al contrario, nei casi di soggetti ipotensivi o ipertensivi, avendo un numero inferiore di casi di apprendimento, l'errore tende a crescere notevolmente.

In figura 4.8 e 4.9 è mostrato lo stesso insieme di grafici appena descritto applicato sui dati di pressione sistolica e arteriosa. I box plot presentano le misure standard ovvero il rettangolo è delimitato dal primo e terzo quartile (q_1 e q_3), con la linea indicante la mediana. I baffi sono a distanza $q_1 - 1.5 \times IQR$ e $q_3 + 1.5 \times IQR$ rispettivamente per il baffo inferiore e superiore.

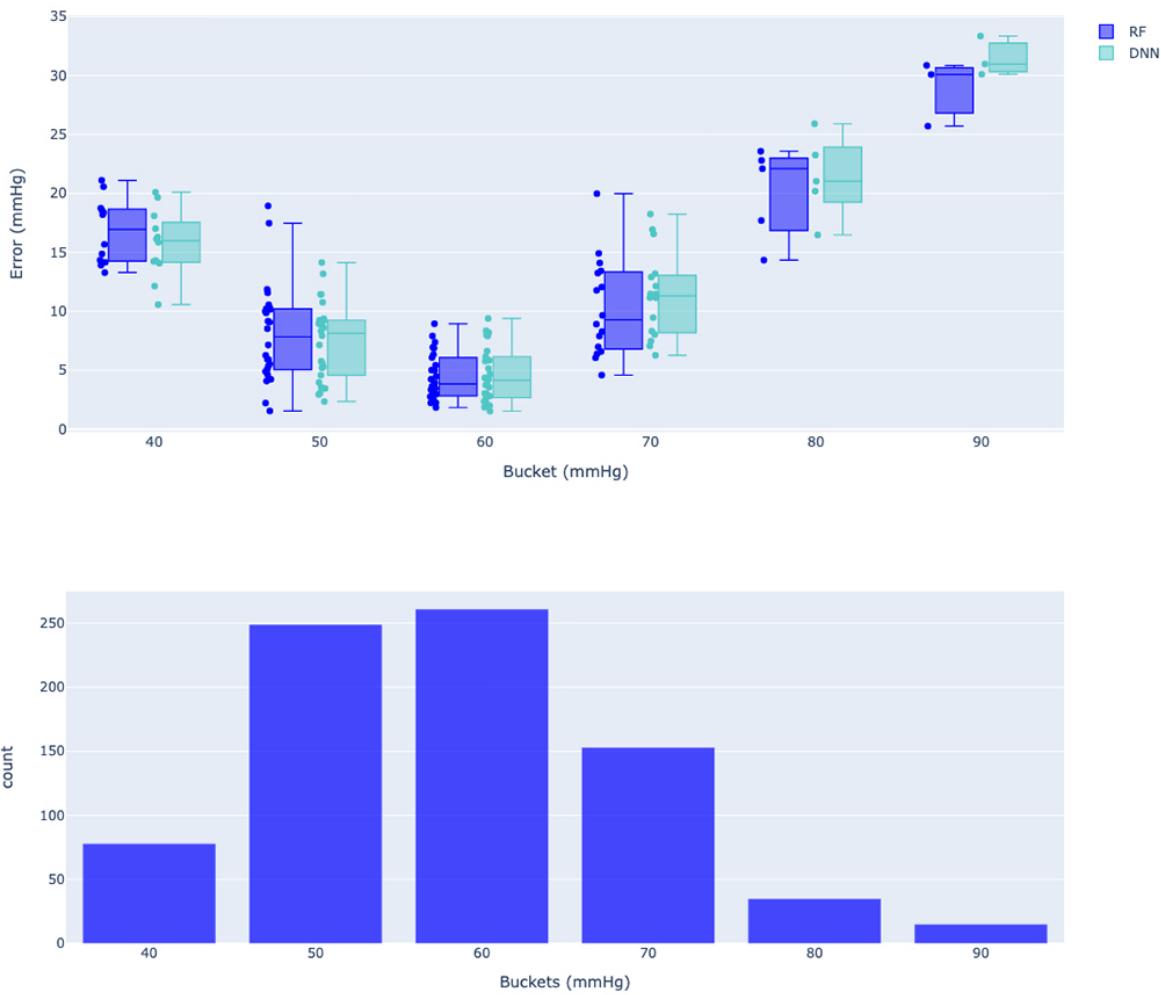


Figura 4.7: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione diastolica su VitalDB con finestre da 30 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione diastolica anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
40/50	16.81	15.70	17.35	18.73
50/60	8.09	7.42	8.18	7.08
60/70	4.44	4.59	4.80	4.65
70/80	10.29	11.36	9.10	10.29
80/90	20.10	21.37	20.07	22.08
90/100	28.88	31.48	30.91	30.65

Tabella 4.2: Media dei MAE di ogni dato di test (mmHg) su intervalli della pressione diastolica basati su VitalDB con finestre da 30 secondi.

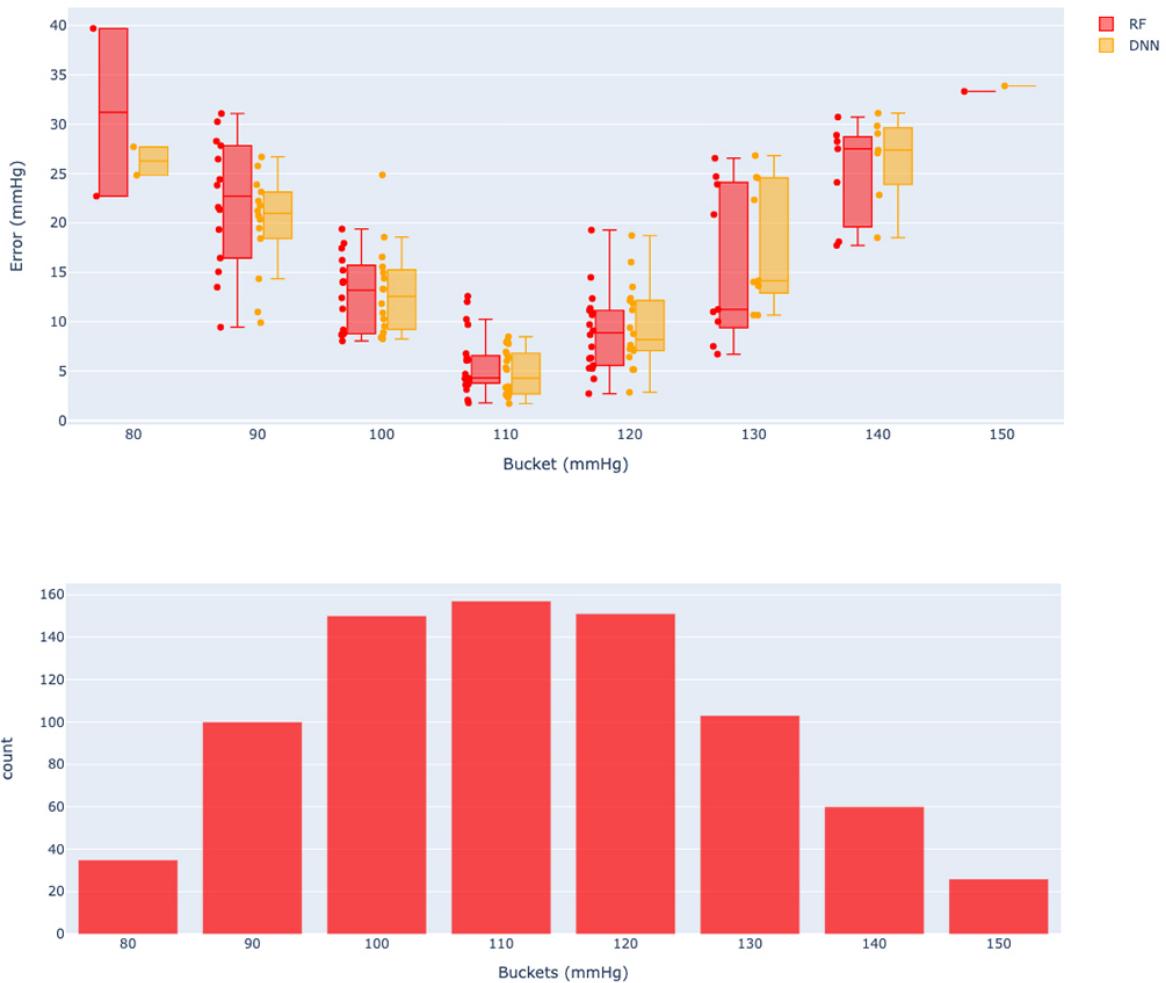


Figura 4.8: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione sistolica su VitalDB con finestre da 30 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione sistolica anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
80/90	31.21	26.27	31.19	28.15
90/100	22.06	19.92	19.13	20.92
100/110	12.76	13.01	13.84	14.57
110/120	5.69	4.82	6.71	6.30
120/130	8.93	9.45	7.38	9.17
130/140	15.83	17.94	16.29	13.57
140/150	25.04	26.54	27.01	25.97
150/160	33.31	33.88	35.43	31.13

Tabella 4.3: Media dei MAE (mmHg) di ogni dato di test su intervalli della pressione diastolica basati su VitalDB con finestre da 30 secondi.

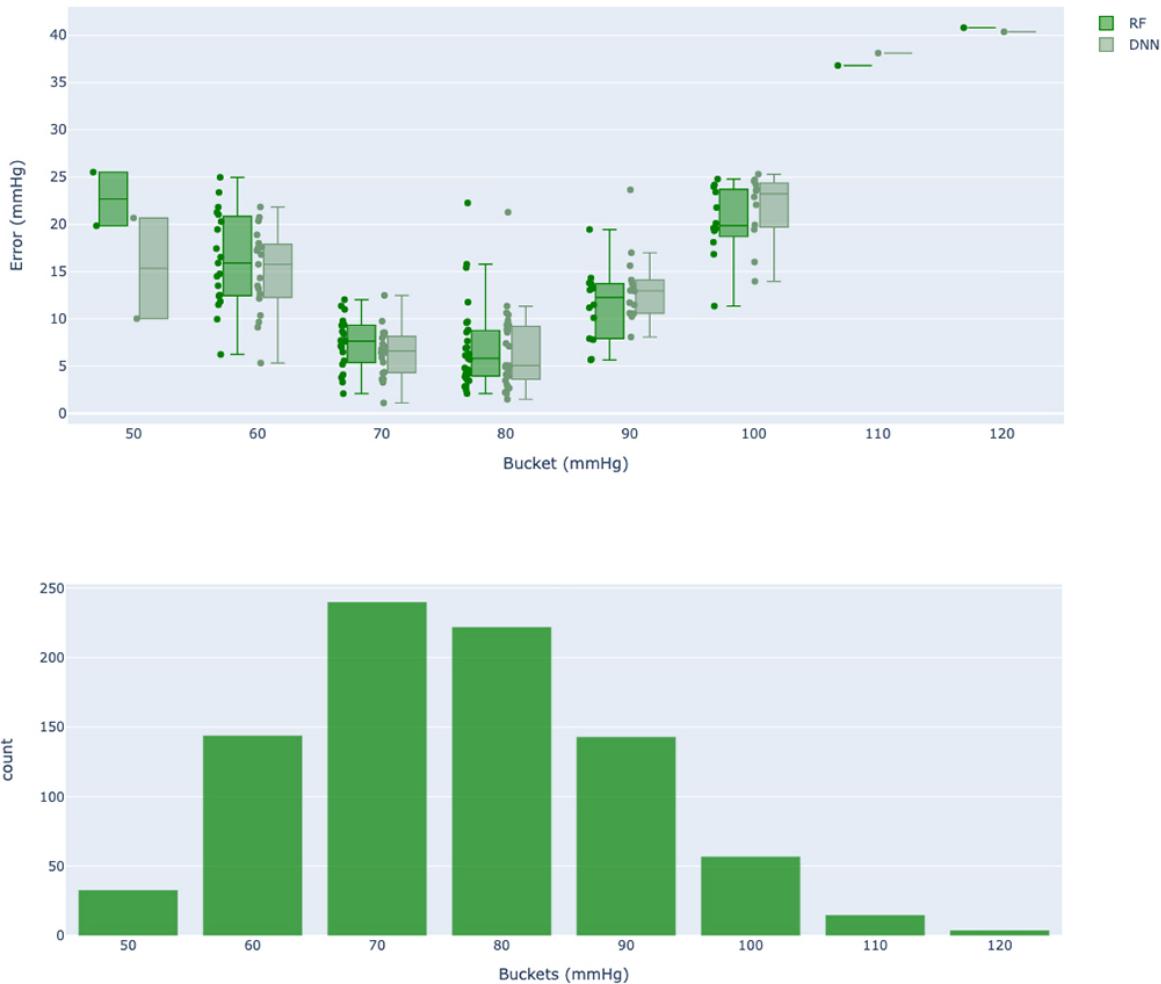


Figura 4.9: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione arteriosa media su VitalDB con finestre da 30 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione arteriosa media anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
50/60	22.68	15.34	23.15	23.51
60/70	16.27	14.99	13.31	14.10
70/80	7.37	6.40	8.19	6.27
80/90	7.06	6.55	6.36	5.98
90/100	11.49	13.30	11.70	11.60
100/110	20.25	21.69	23.01	23.57
110/120	36.80	38.11	32.32	30.95
120/130	40.80	40.37	N.V.	N.V.

Tabella 4.4: Media dei MAE (mmHg) di ogni dato di test su intervalli della pressione diastolica basati su VitalDB con finestre da 30 secondi.

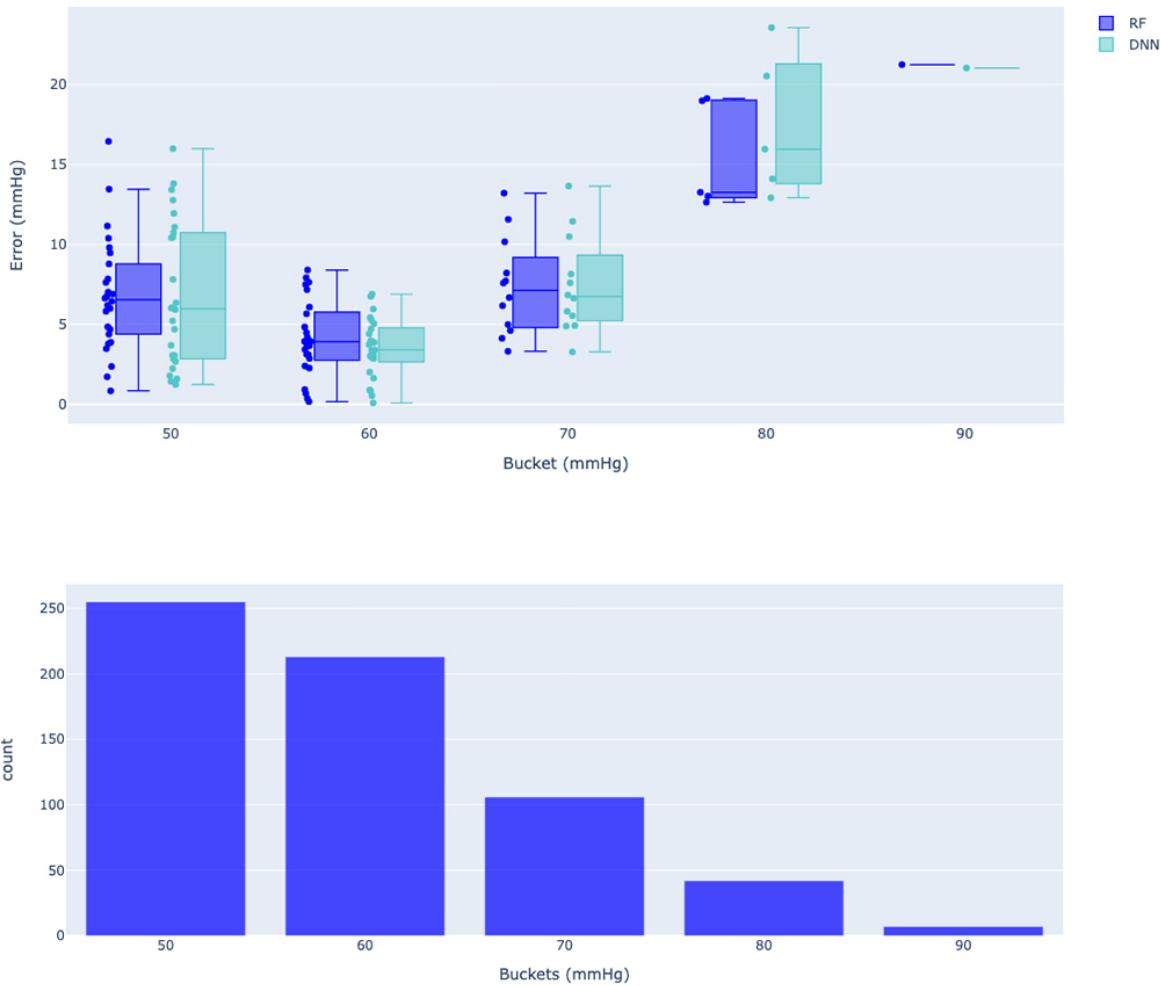


Figura 4.10: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione diastolica su CufflessDB con finestre da 40 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione diastolica anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
50/60	7.53	6.26	6.78	6.77
60/70	3.88	4.40	4.08	3.49
70/80	11.81	11.34	7.35	7.43
80/90	17.49	15.67	15.39	17.40
90/100	26.88	29.22	21.23	21.02

Tabella 4.5: Media dei MAE di ogni dato di test (mmHg) su intervalli della pressione diastolica basati su CufflessDB con finestre da 40 secondi.

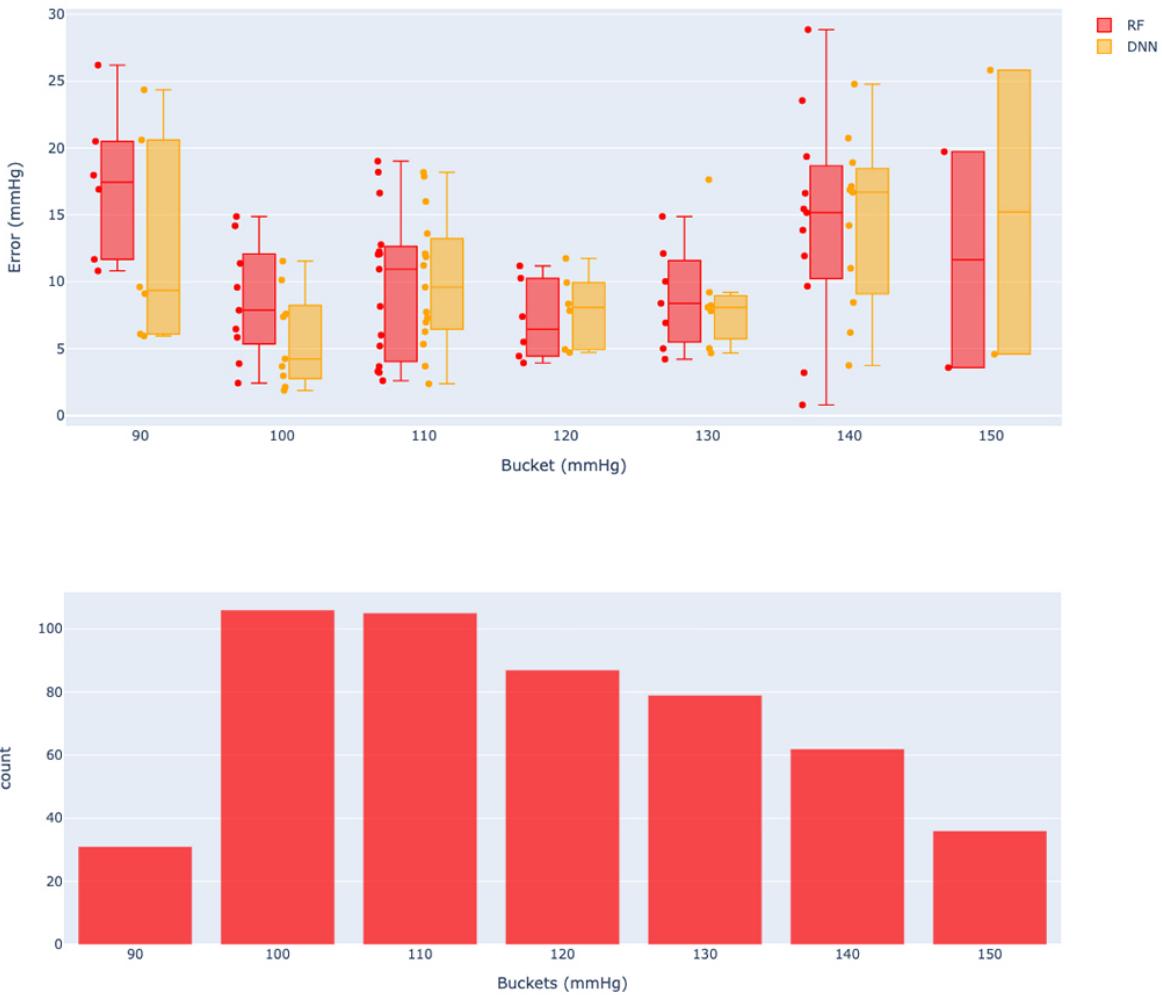


Figura 4.11: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione sistolica su CufflessDB con finestre da 40 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione sistolica anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
90/100	11.30	12.39	17.34	12.62
100/110	11.00	9.22	8.50	5.73
110/120	8.79	8.85	9.74	10.01
120/130	10.24	7.20	7.12	7.92
130/140	14.82	9.14	8.79	8.67
140/150	19.08	15.23	14.40	14.43
150/160	32.13	25.74	11.65	15.20

Tabella 4.6: Media dei MAE di ogni dato di test (mmHg) su intervalli della pressione sistolica basati su CufflessDB con finestre da 40 secondi.

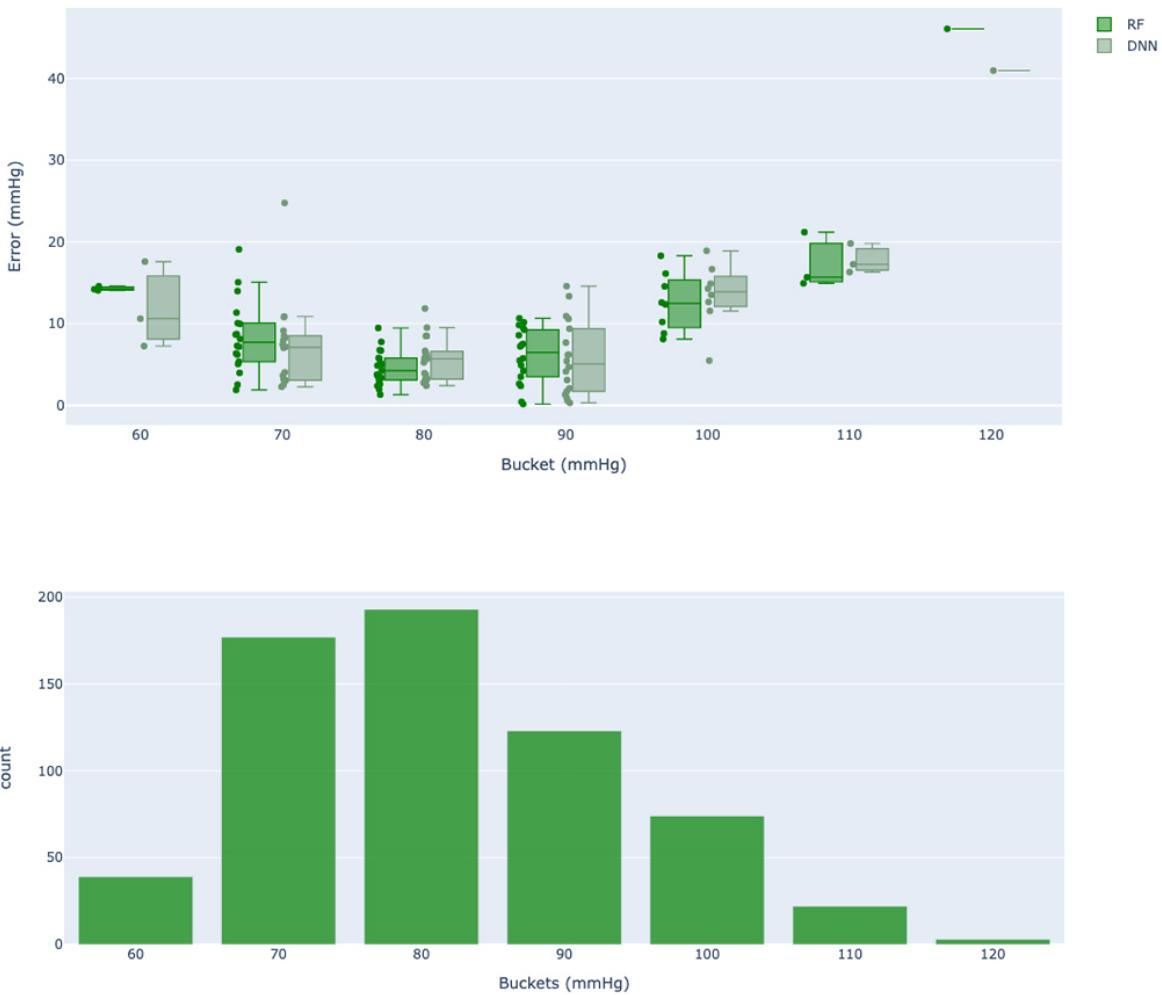


Figura 4.12: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione arteriosa media su CufflessDB con finestre da 40 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione arteriosa media anch'essa su intervalli da 10 mmHg.

Intervallo (sec)	MAE (30 sec)		MAE (40 sec)	
	RF	DNN	RF	DNN
60/70	12.52	8.78	14.27	11.81
70/80	9.97	8.36	8.37	7.04
80/90	6.20	5.44	4.57	5.53
90/100	5.06	5.72	6.08	5.72
100/110	14.95	14.88	12.62	13.48
110/120	21.58	27.20	17.26	17.78
120/130	N.V.	N.V.	46.06	40.95

Tabella 4.7: Media dei MAE di ogni dato di test (mmHg) su intervalli della pressione arteriosa media basati su CufflessDB con finestre da 40 secondi.

Dai grafici sopra riportati è molto chiaro come l'errore sia più basso nei punti in cui la concentrazione dei dati di apprendimento è maggiore. Questo porta a pensare che avendo un dataset omogeneo di soggetti ipotensivi, normo-tensivi ed ipertensivi, i risultati ottenibili siano simili a quelli ottenuti attorno ai 60 mmHg (nel caso di pressione diastolica) e 110 mmHg (nel caso di pressione sistolica). Infine, nelle figure 4.7, 4.8 e 4.9 sono mostrati i box plot (riportanti la mediana) riassuntivi di quelli appena descritti che mostrano la distribuzione dell'errore sui dati di pressione diastolica, sistolica e arteriosa media per finestre di 30 secondi. Per completezza sono riportate anche le tabelle 4.2, 4.3 e 4.4 che mostrano il MAE medio di ogni intervallo rispettivamente per la pressione diastolica, sistolica e arteriosa. Inoltre nella tabella 4.8 è mostrato, in maniera riassuntiva, la media dei MAE per la pressione diastolica, sistolica e arteriosa.

Allo stesso modo, per il dataset CufflessDB sono mostrati i risultati nelle figure 4.10, 4.11 e 4.12. Anche in questo caso sono riportate le tabelle 4.5, 4.6 e 4.7 per ogni misurazione (diastolica, sistolica e arteriosa) ed è facile notare come la capacità dei modelli sia notevolmente migliore nei casi ove la concentrazione dei dati di training è maggiore. Infine nelle figure 4.16, 4.17 e 4.18 sono riportati i confronti mediante box plot (riportanti la mediana) per i modelli RF e DNN per le varie misurazioni (diastolica, sistolica e arteriosa). In tabella 4.9 sono riportate le medie dei MAE dei modelli RF e DNN per la pressione diastolica, sistolica e arteriosa.

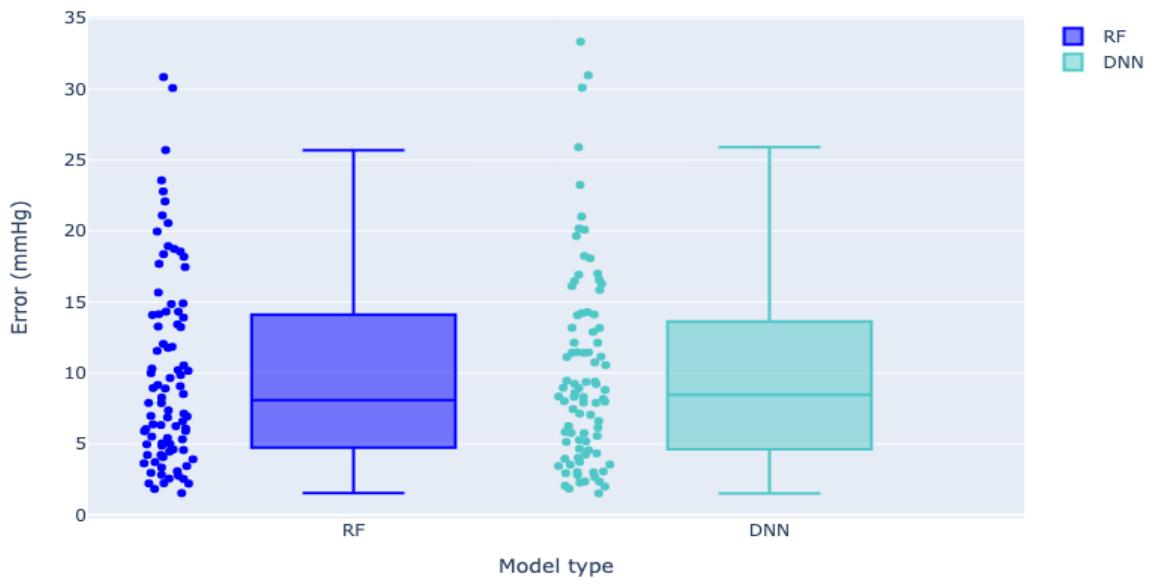


Figura 4.13: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione diastolica su VitalDB con finestre da 30 secondi.

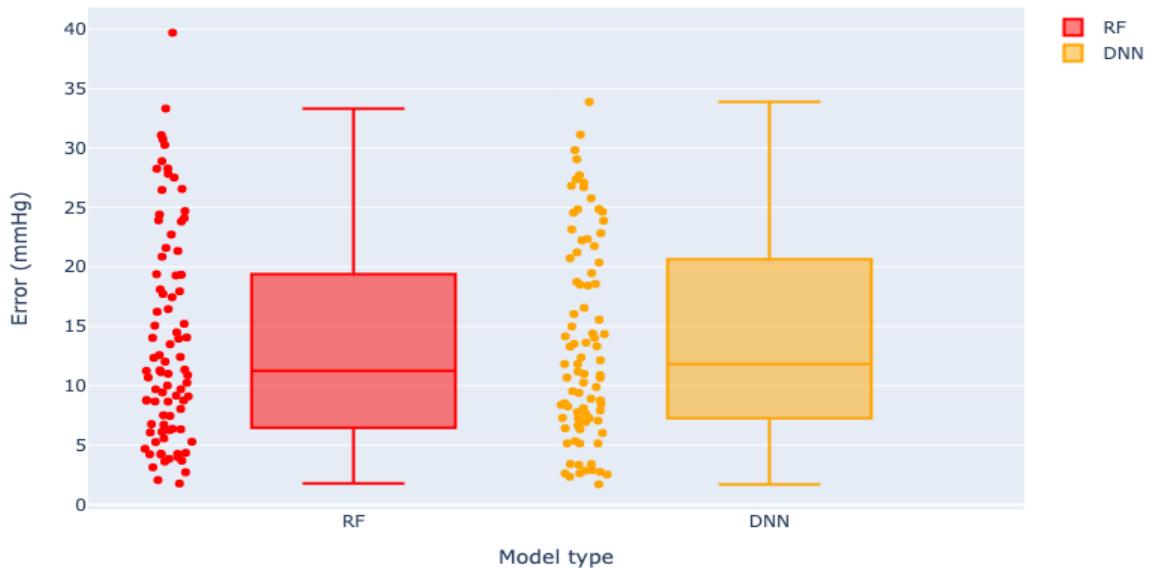


Figura 4.14: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione sistolica su VitalDB con finestre da 30 secondi.

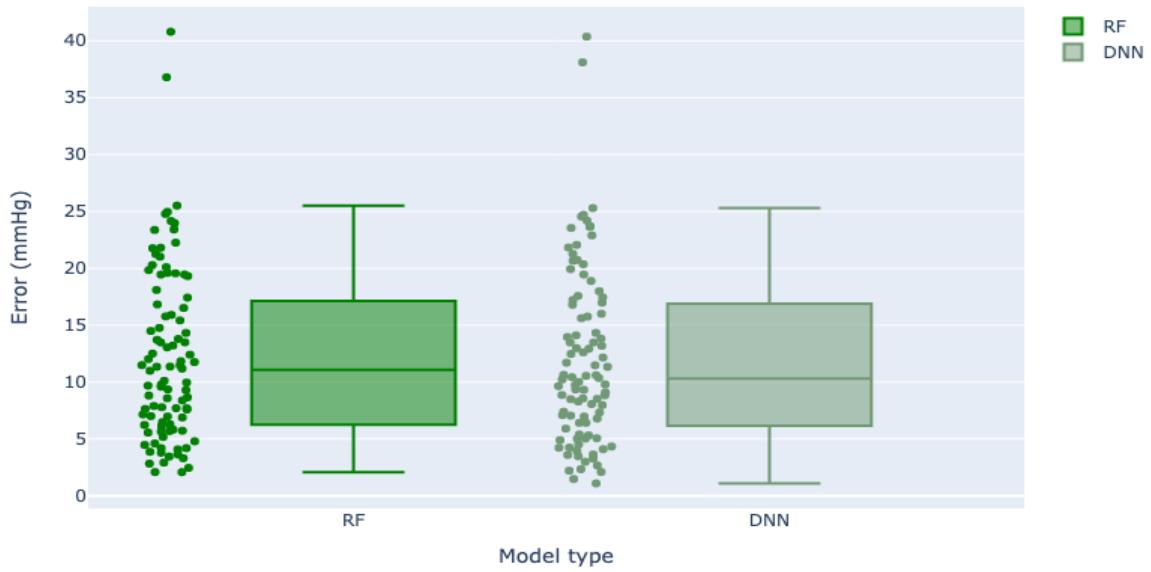


Figura 4.15: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione arteriosa media su VitalDB con finestre da 30 secondi.

Finestra (sec)	MAE DBP		MAE SBP		MAE MAP	
	RF	DNN	RF	DNN	RF	DNN
30	9.99	10.04	13.80	13.64	12.23	11.93
40	9.90	9.98	14.35	14.18	11.82	11.44

Tabella 4.8: Media dei MAE su ogni caso di test (mmHg) su VitalDB per i modelli RF e DNN per ogni misurazione.

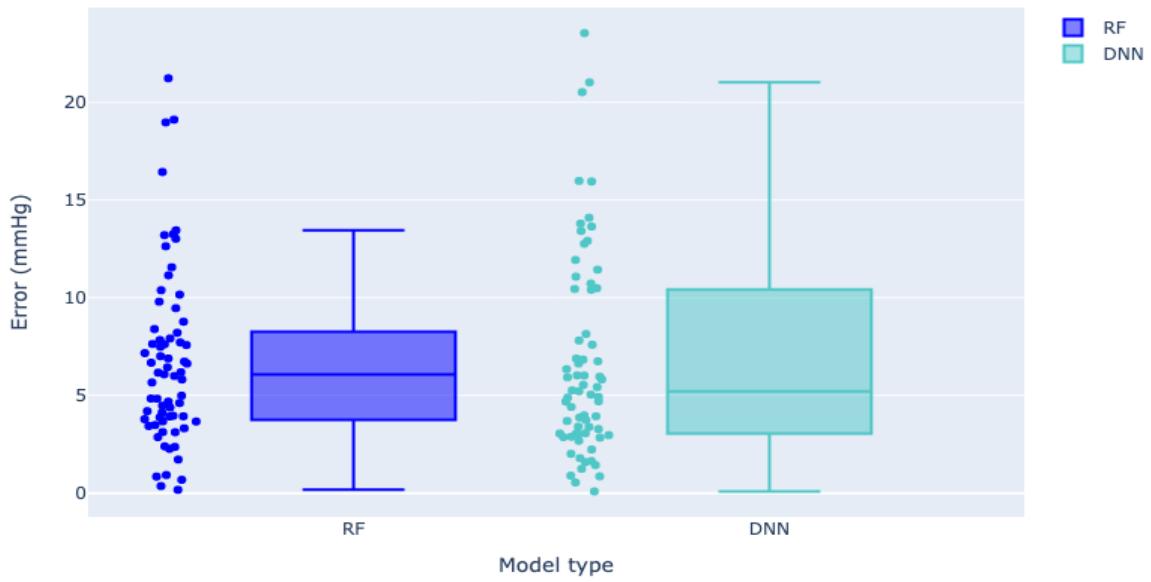


Figura 4.16: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione diastolica su CufflessDB con finestre da 40 secondi.

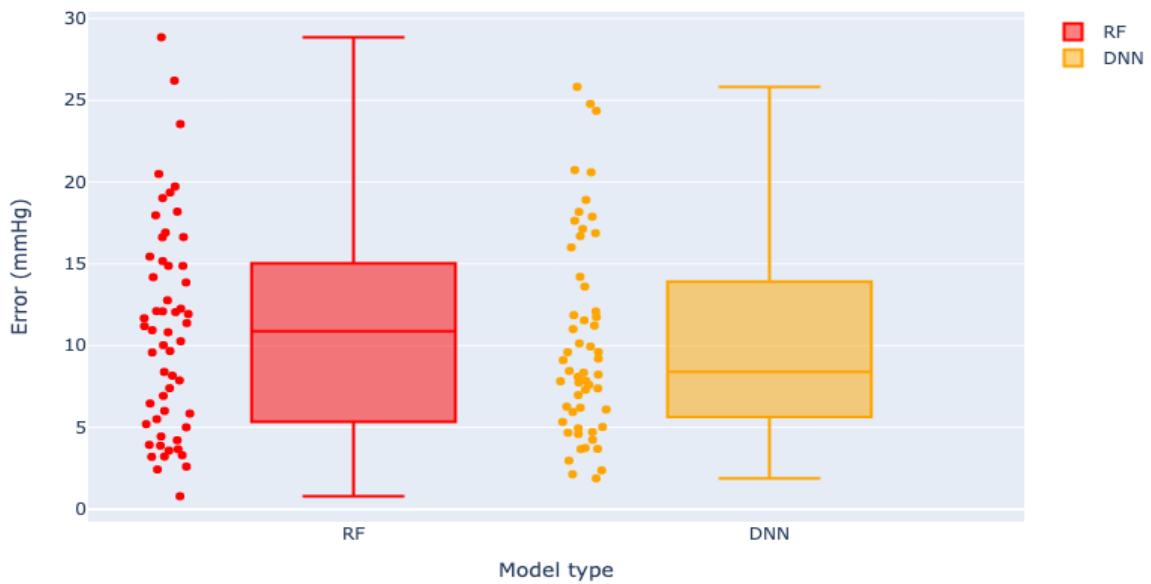


Figura 4.17: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione sistolica su CufflessDB con finestre da 40 secondi.

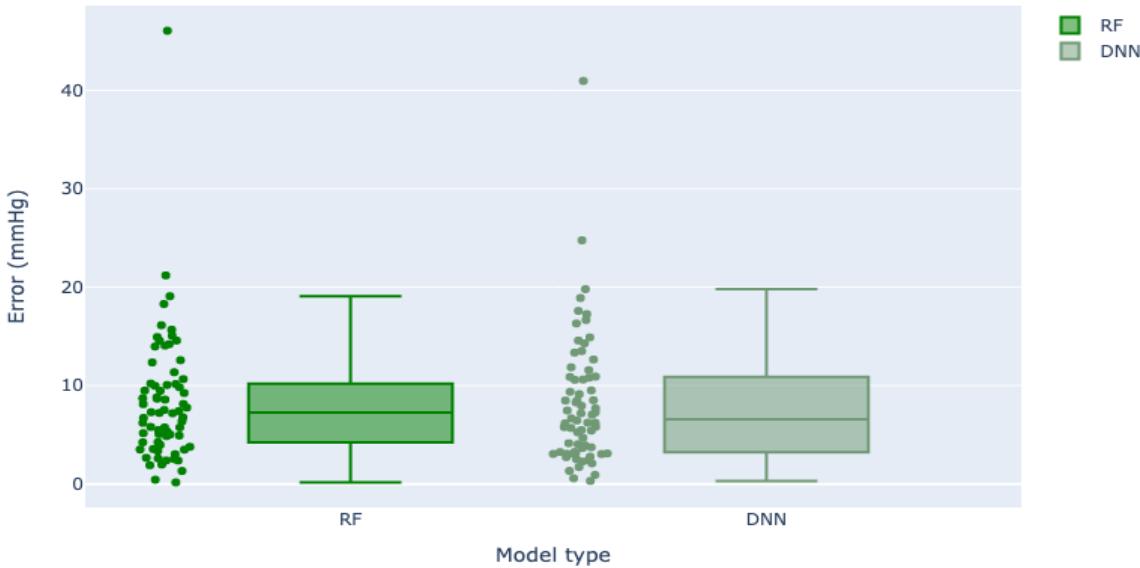


Figura 4.18: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN nel caso di pressione arteriosa media su CufflessDB con finestre da 40 secondi.

Finestra (sec)	MAE DBP		MAE SBP		MAE MAP	
	RF	DNN	RF	DNN	RF	DNN
30	8.14	7.50	14.31	11.41	9.95	9.27
40	6.74	6.67	10.94	10.26	8.41	8.18

Tabella 4.9: Media dei MAE su ogni caso di test (mmHg) su CufflessDB per i modelli RF e DNN per ogni misurazione.

4.3.2 Risultati tramite segnale rPPG

Relativamente al caso del dataset principale, il V4V, le analisi sono state più approfondate in quanto esso rappresenta un dataset completo di dati utili alla ricerca condotta. In questa ultima parte verranno mostrati i risultati sulla finestratura da 20 secondi (che risultano essere quelli più completi grazie alla quantità di dati ad essi riservata) comparando il modello RF con le DNN in entrambe le modalità di estrazione del segnale rPPG.

In figura 4.19 è mostrato il grafico della distribuzione di errore sulla finestratura di 20 secondi per la pressione diastolica. In figura 4.20 e 4.21 sono mostrati l'analogo della misurazione appena descritta rispettivamente per la pressione sistolica e arteriosa. In tutte le figure, le distribuzioni dei dati di training equivalgono alla media tra i dati estratti con il metodo olistico e il metodo patch. Inoltre, per ogni grafico degli errori,

sono presenti tutte e 4 le combinazioni:

- metodo patch analizzato con random forest,
- metodo patch analizzato con DNN,
- metodo olistico analizzato con random forest,
- metodo olistico analizzato con DNN.

Per ognuna delle figure, è riportata la tabella con i risultati per ogni intervallo. Nello specifico la tabella 4.10 per la pressione diastolica, la tabella 4.11 per la pressione sistolica e la tabella 4.12 per la pressione arteriosa.

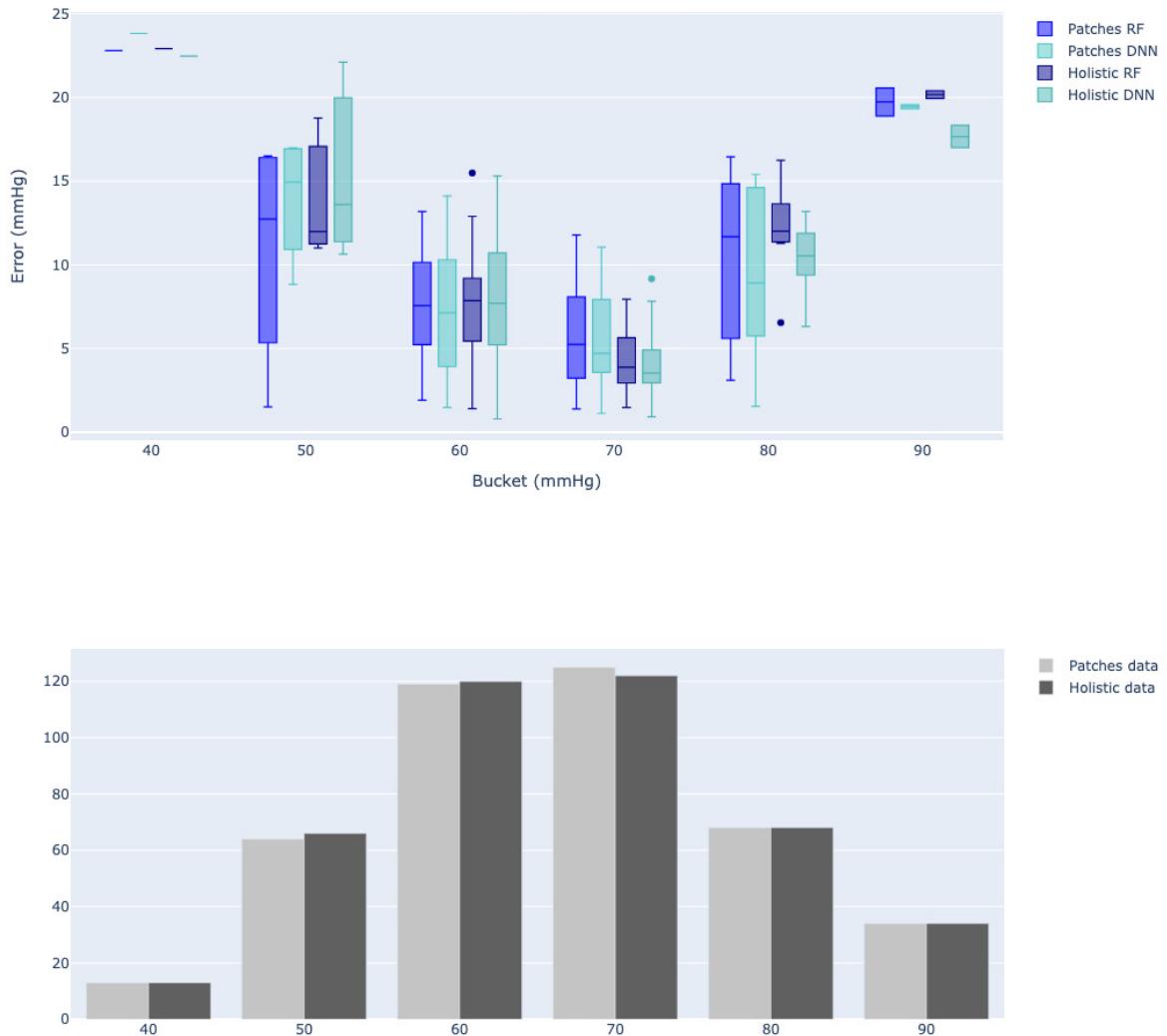


Figura 4.19: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN con estrazione rPPG olistica e patch nel caso di pressione diastolica su V4V con finestre da 20 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione diastolica dei soggetti il cui segnale rPPG è estratto tramite metodo patch (grigio chiaro) e olistico (grigio scuro) entrambe su intervalli da 10 mmHg.

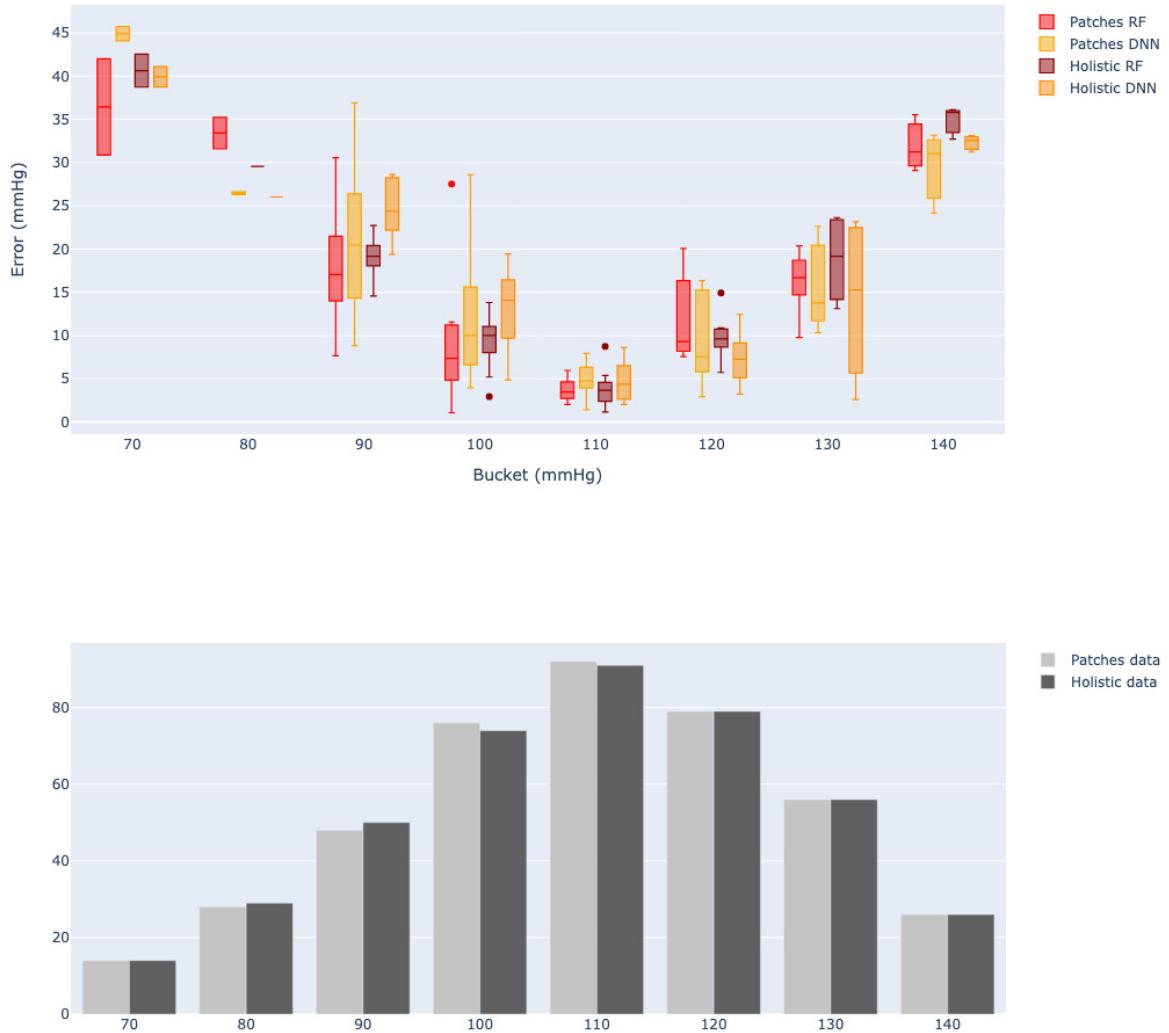


Figura 4.20: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN con estrazione rPPG olistica e patch nel caso di pressione sistolica su V4V con finestre da 20 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione sistolica dei soggetti il cui segnale rPPG è estratto tramite metodo patch (grigio chiaro) e olistico (grigio scuro) entrambe su intervalli da 10 mmHg.

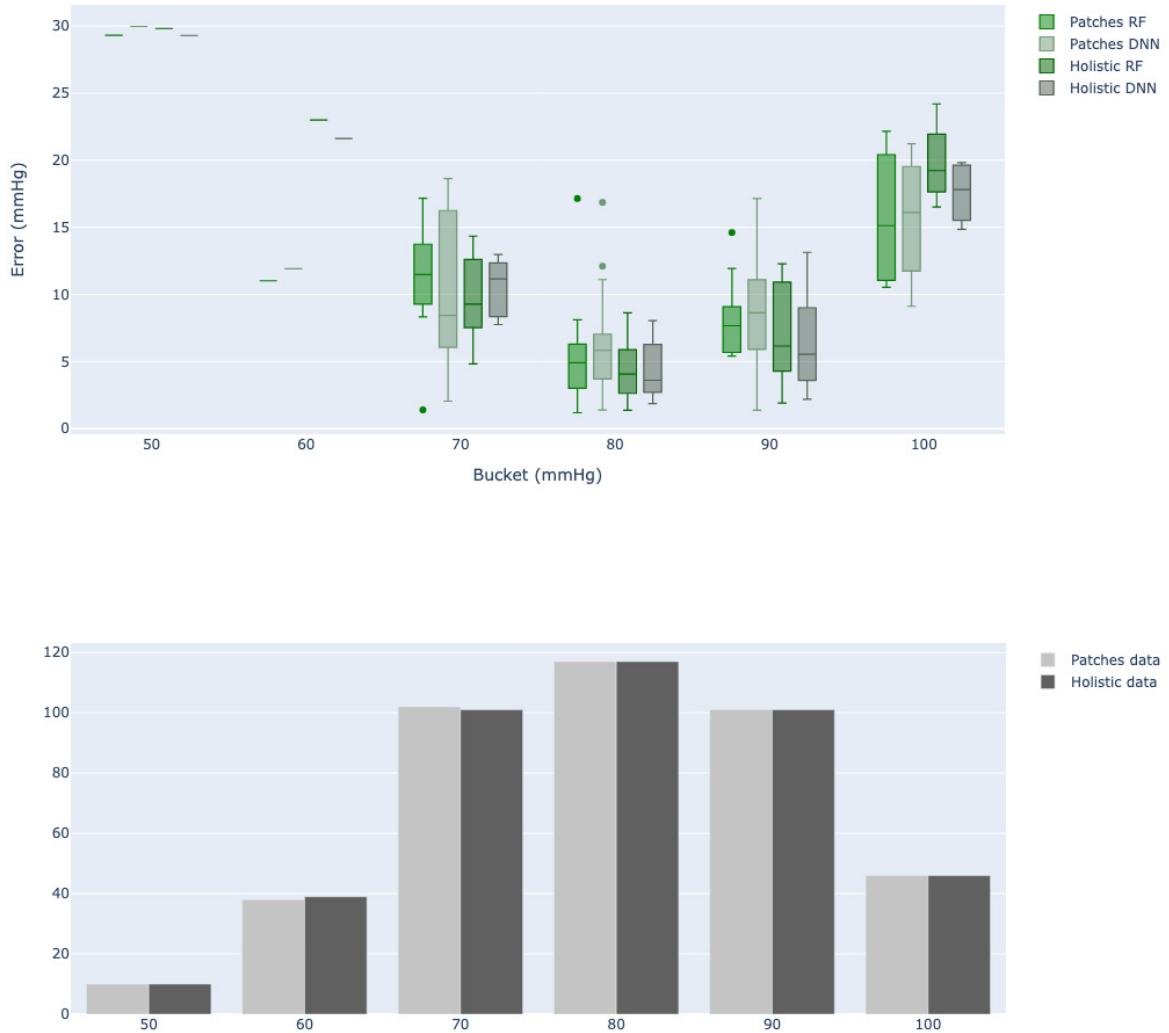


Figura 4.21: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN con estrazione rPPG olistica e patch nel caso di pressione arteriosa media su V4V con finestre da 20 secondi su intervalli da 10 mmHg. Nel grafico inferiore è raffigurata la distribuzione dei dati di training suddivisa per livello medio di pressione arteriosa media dei soggetti il cui segnale rPPG è estratto tramite metodo patch (grigio chiaro) e olistico (grigio scuro) entrambe su intervalli da 10 mmHg.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
40/50	22.91	22.47	22.08	23.31
50/60	13.91	15.44	14.67	12.10
60/70	7.71	8.04	7.15	7.53
70/80	4.25	4.04	4.93	5.45
80/90	12.13	10.37	14.06	13.24
90/100	20.17	17.66	20.32	20.06

Tabella 4.10: Media dei MAE (espressa in mmHg) per ogni caso di test su intervalli da 10 mmHg della pressione diastolica su V4V con finestre da 20 secondi.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
70/80	40.64	39.92	36.65	39.34
80/90	29.56	26.02	29.85	34.43
90/100	19.01	24.54	16.72	18.70
100/110	9.36	13.27	8.27	10.10
110/120	3.83	4.60	3.57	4.28
120/130	9.84	7.34	11.50	10.12
130/160	18.78	14.08	19.63	17.93
140/150	34.87	32.29	34.05	30.44

Tabella 4.11: Media dei MAE (espressa in mmHg) per ogni caso di test su intervalli da 10 mmHg della pressione sistolica su V4V con finestre da 20 secondi.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
50/60	28.59	31.54	29.14	30.11
60/70	23.00	21.61	21.75	24.75
70/80	9.72	10.58	9.83	10.47
80/90	4.51	4.50	4.06	4.11
90/100	7.08	6.45	8.31	8.58
100/110	19.79	17.57	15.46	17.59

Tabella 4.12: Media dei MAE (espressa in mmHg) per ogni caso di test su intervalli da 10 mmHg della pressione arteriosa media su V4V con finestre da 20 secondi.

Anche in questo caso, negli intervalli di pressione normale l'errore si riduce notevolmente rispetto all'errore che si ottiene per i soggetti con disturbi di pressione. Nelle figure 4.22, 4.23 e 4.24, sono riportate le distribuzioni degli errori (in generale) rispettivamente per la pressione diastolica, sistolica e arteriosa. In generale, non ci sono *pipeline* che predominano su altre, infatti nelle tabelle 4.13, 4.14 e 4.15 sono riportati i risultati medi per ogni modello rispetto a ogni finestratura rispettivamente per la pressione diastolica, sistolica e arteriosa.

Complessivamente i risultati non sono perfettamente confrontabili. Questo è dovuto al fatto che la quantità di dati su cui si basano è diversa a seconda della finestratura. Come riportato nella tabella 4.1 la differenza di video di test è netta. Quindi si può dire che nel caso dei risultati su finestre da 20 secondi, in generale, si ha il risultato migliore per tutte le misurazioni. Questo è dovuto al fatto che in finestre da 20 secondi si ha un numero nettamente più alto dei casi di test e di apprendimento. Seppur nel caso della pressione diastolica si ha un MAE minimo pari a 5.64 mmHg, è anche vero che i casi di test sono in numero ridotto (22) contro il MAE del metodo olistico DNN pari a 7.80 mmHg su finestre da 20 con più del doppio dei video di test (48). Inoltre, le finestre da 20 secondi rappresentano un uso molto più pratico e significativo rispetto a finestre da 50 secondi (su finestre così ampie il valore medio rischia di diventare poco informativo). Analogamente per la pressione sistolica non si ha un minimo assoluto, ma quasi. Il MAE minimo ottenuto equivale a 13.66 mmHg contro 13.31 mmHg su finestre da 30 secondi. Anche qui avendo un numero di test maggiore e un errore con uno scarto di 0.3 mmHg si può ritenere una soluzione più performante quella da 20 secondi. Per finire, nel caso di pressione arteriosa, si ha un MAE minimo di 8.32 mmHg che risulta migliore, per i motivi appena descritti, rispetto a un errore pari a 8.14 mmHg tramite finestre da 30 secondi e a 7.84 mmHg tramite finestre da 60 secondi.

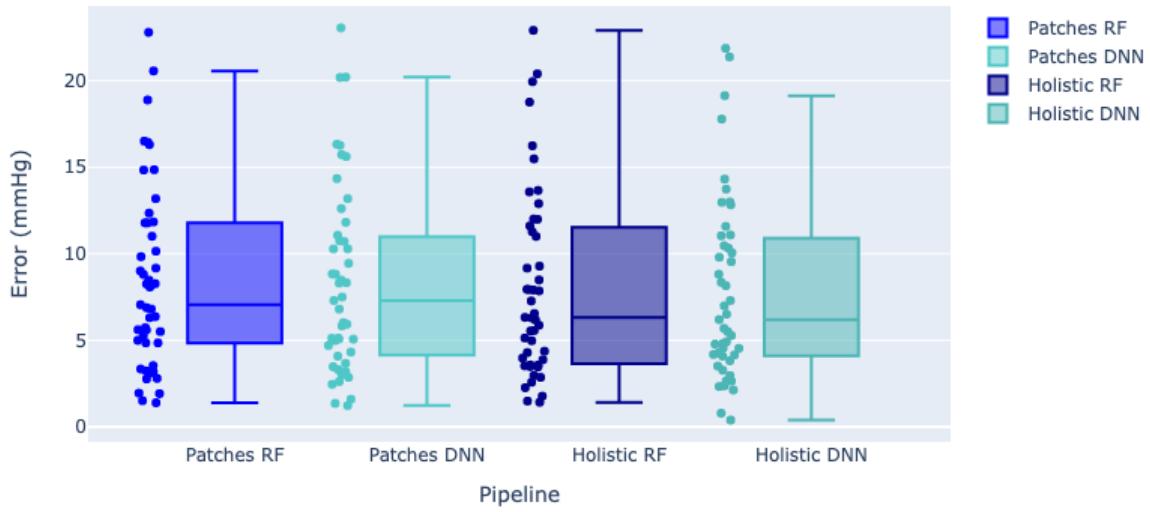


Figura 4.22: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN applicati a segnale rPPG estratto con metodo patch e olistico nel caso di pressione diastolica su V4V con finestre da 20 secondi.

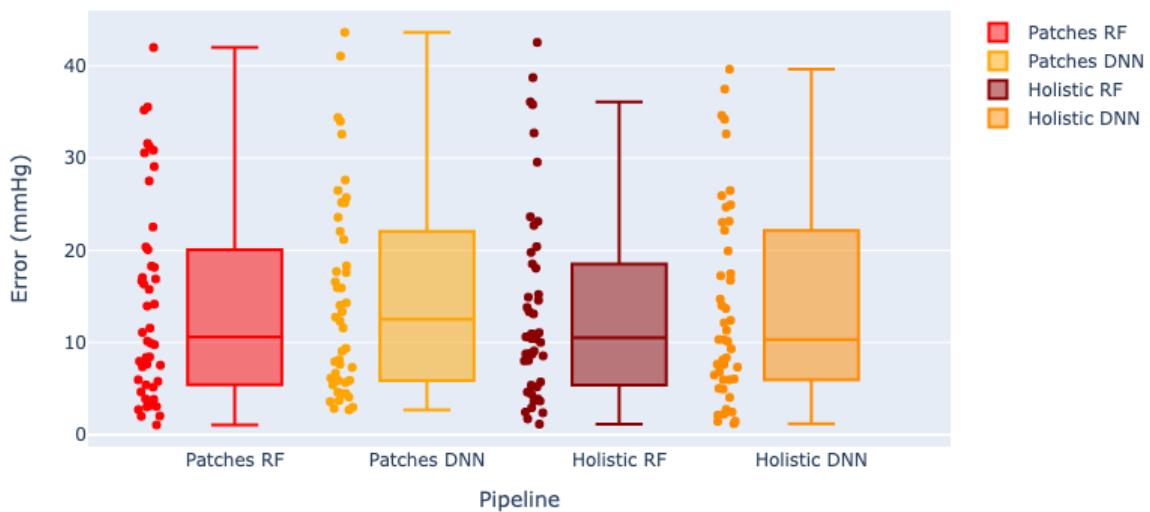


Figura 4.23: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN applicati a segnale rPPG estratto con metodo patch e olistico nel caso di pressione sistolica su V4V con finestre da 20 secondi.

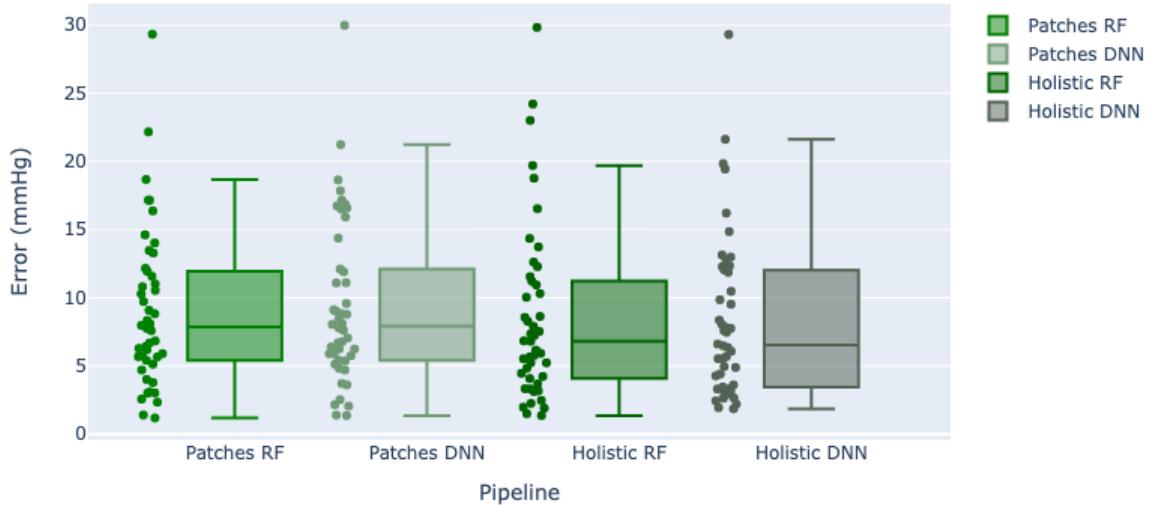


Figura 4.24: Box plot standard (riportanti la mediana) del MAE tramite RF e DNN applicati a segnale rPPG estratto con metodo patch e olistico nel caso di pressione arteriosa media su V4V con finestre da 20 secondi.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
20	8.07	7.80	8.46	8.32
30	7.51	7.48	7.87	8.77
40	6.70	7.02	8.54	8.45
50	6.51	5.64	7.40	8.47
60	8.05	10.19	8.84	9.42

Tabella 4.13: Media dei MAE (espressa in mmHg) ottenuta tramite estrazione rPPG olistico e patch con RF e DNN per ogni caso di test della pressione diastolica su V4V suddivisi per finestre.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
20	13.66	13.85	14.25	14.82
30	13.31	13.81	12.45	12.85
40	14.51	13.99	12.63	16.21
50	13.83	16.00	13.49	14.37
60	13.83	16.48	10.51	13.82

Tabella 4.14: Media dei MAE (espressa in mmHg) ottenuta tramite estrazione rPPG olistico e patch con RF e DNN per ogni caso di test della pressione diastolica su V4V suddivisi per finestre.

Intervallo (sec)	rPPG olistico		rPPG patch	
	RF	DNN	RF	DNN
20	8.54	8.32	9.05	9.39
30	8.14	9.14	8.50	8.41
40	10.24	9.95	9.44	10.91
50	9.12	8.74	8.57	11.75
60	7.84	11.79	7.81	9.85

Tabella 4.15: Media dei MAE (espressa in mmHg) ottenuta tramite estrazione rPPG olistico e patch con RF e DNN per ogni caso di test della pressione arteriosa media su V4V suddivisi per finestre.

4.4 Discussione

Lo studio riportato in questa tesi è consistito nell'individuare strade possibili, ancora inesplorate, per stimare la pressione sanguina tramite video. Per quanto ancora lontana la fedeltà richiesta per l'utilizzo in ambiti sanitari o specialistici in generale, l'analisi sperimentale dei modelli proposti nella tesi ha tuttavia mostrato come sia possibile dare una stima con errore contenuto della pressione sanguigna a partire dal solo video del soggetto.

Nelle tabelle 4.13, 4.14 e 4.15 sono mostrati i risultati ottenuti da diverse combinazioni di strumenti. Si evince che, in generale, l'errore commesso è dell'ordine di quello commesso da classici misuratori "da casa", come ad esempio i misuratori da polso (come anche descritto all'inizio del capitolo 2). Tale errore non permette, secondo lo standard (28) rilasciato da AAMI/ESH/ISO, di considerare tale strumento, uno strumento valido per la misurazione della pressione sanguigna.

Inoltre, è da sottolineare che durante questa ricerca la quantità di dati utilizzabili per allenare e testare le soluzioni proposte è stata piuttosto scarsa a causa dei pochissimi dataset disponibili. Nei casi in cui vi è una maggiore concertazione di dati, però, l'errore medio risulta piuttosto basso e promettente. Questo porta a pensare che utilizzando dataset più ricchi di soggetti ipotensivi e ipertensivi, le soluzioni proposte possano risultare valide ottenendo così un risultato più uniforme e mediamente migliore. Infatti, preso il caso del dataset CufflessDB, dove la distribuzione della è relativamente omogenea, si ottengono i risultati migliori in tutte le misurazioni (nel caso di finestre a 40 secondi).

In conclusione sono stati mostrati i risultati della *pipeline* studiata in questa tesi su diversi dataset. Nel caso dei dataset CufflessDB e VitalDB, ovvero i dataset con il segnale PPG, è stato mostrato, specialmente per CufflessDB, che il MAE medio a partire da un segnale fotopletismografico (quasi) privo di errore il MAE ottenibile è discretamente ridotto rispetto al MAE ottenibile tramite segnale rPPG. Questo significa che se durante l'estrazione dell'rPPG si riesca a ridurre l'errore ad esso legato la *pipeline*

applicata su video può produrre degli errori simili a quelli ottenuti con CufflessDB ovvero ad un MAE medio del segnale pari a 6.67 mmHg nel caso di diastolica, 10.26 mmHg nel caso di sistolica e di 8.18 mmHg per la arteriosa media (per finestre da 40 secondi).

Conclusioni e sviluppi futuri

In questa tesi è stato studiato un sistema specifico per la stima della pressione sanguigna tramite video. Lo studio è stato effettuato su un dataset principale denominato *Vision For Vitals* e altri due secondari denominati *MIMIC-III* e *VitalDB*. Nel primo caso sono contenuti diversi video raffiguranti il volto di soggetti esposti a diverse condizioni di luce e angolature. Nel caso dei dataset secondari è riportato un particolare segnale denominato fotopletismografico estratto mediante un apposito strumento. In tutti i dataset utilizzati è riportato ovviamente anche il segnale della pressione sanguigna.

Il sistema studiato si basa in primo luogo sul segnale fotopletismografico. Per questo motivo sono stati utilizzati dataset comprensivi solamente di tale segnale perchè prodotto da uno strumento apposito privo di errore. Al contrario, nel caso di *Vision For Vitals*, il segnale fotopletismografico è stimato dal video mediante particolari algoritmi e di conseguenza racchiude intrinsecamente un errore. In questo modo, *Vision For Vitals* è stato utilizzato per identificare la precisione di predizione ottenibile realmente, mentre *MIMIC-III* e *VitalDB* hanno permesso di individuare i limiti teorici ottenibili da tale sistema basandosi su un segnale fotopletismografico "puro". Gli esperimenti infine hanno portato a discreti risultati che però, stando alle regole poste da *Association for the Advancement of Medical Instrumentation* (AAMI), non permettono di utilizzare tale sistema come strumento valido per la misurazione della pressione sanguigna.

Nello specifico i risultati hanno mostrato un *mean average error* (MAE) pari a 7.80 mmHg nel caso della pressione diastolica, 13.66 mmHg nel caso di pressione sistolica e 8.32 mmHg nel caso di pressione arteriosa media. Mediante gli altri dataset è stato mostrato che partendo da un segnale fotopletismografico (quasi) perfetto, il MAE si abbassa fino a 6.67 mmHg per la pressione diastolica, 10.26 mmHg per la pressione sistolica e 8.18 mmHg per la pressione arteriosa. Questa ricerca ha permesso di esplorare in maniera piuttosto approfondita il funzionamento della *pipeline* studiata nella stima della pressione sanguigna da remoto. Proprio per il fatto che la quantità di dati utilizzabili è stata ridotta, in futuro sarebbe possibile approfondire lo studio utilizzando dataset più ricchi che abbiano una distribuzione di soggetti ipotensivi, normo-tensivi e ipertensivi omogenea. In tal modo le stime potrebbero risultare più precise su tutte le categorie di pressione migliorando le performance di predizione.

Sul piano egli sviluppi futuri, il framework FaceQs qui presentato per la prima

volta, è nella sua fase preliminare e lo sono anche i metodi di stima della pressione qui mostrati mediante analisi sperimentale. Non v'è dubbio che ci siano ancora molti ambiti in cui innovare e migliorare la stima di questo (ed altri) segnale fisiologico, tutti egualmente importanti per le numerose applicazioni cui si è fatto accennato anche nei capitoli precedenti, con l'indubbio vantaggio di risultare del tutto non invasivo per il soggetto in esame o paziente in cura svincolati da dispositivi medicali indossabili. Sul piano modellistico si potrebbero prendere in esame le molteplici stime di segnale PPG derivanti dal metodo patch che pyVHR produce. Questo potrebbe portare a costruire una molteplicità di stimatori che con tecniche di ensemble opportune darebbero sicuramente stime con confidenza più elevata. Occorrerebbe poi definire nuovi dataset, più ricchi e omogenei (sul piano della distribuzione delle categorie di pressione dei soggetti) per il training in quanto comportano un punto fondamentale per le performance ottenibili tramite il sistema studiato durante questo lavoro di tesi.

Bibliografia

- [1] Telehealth. *Wikipedia*, 2022. <https://en.wikipedia.org/wiki/Telehealth>.
- [2] What is hypertension? *World Health Organization - WHO*, 2021. <https://www.who.int/news-room/fact-sheets/detail/hypertension>.
- [3] Angina (angina pectoris). *MSD Manuals*. <https://www.msdmanuals.com/it-it/casa/disturbi-cardiaci-e-dei-vasi-sanguigni/coronaropatia/angina>.
- [4] Disturbo del ritmo circadiano del sonno: che cos'è e come incide sul nostro sonno. *Sonnocare*, 2022. <https://www.sonnocare.it/disturbo-del-ritmo-circadiano-del-sonno-che-cose-e-come-incide-sul-nostro-sonno-E2%80%A8>.
- [5] Redazione Mypersonaltrainer. Pressione arteriosa, cos'è e come si misura. *My Personal Trainer*, 2020. <https://www.my-personaltrainer.it/Ipertensione/misurare-pressione.html>.
- [6] EMTprep Staff. Map - understanding mean arterial pressure. *EMTprep*, 2017. <https://emtprep.com/resources/article/map-understanding-mean-arterial-pressure>.
- [7] Christoph Aust Georg Osterhoff Fabian Schrumpf, Patrick Frenzel and Mirco Fuchs. Assessment of non-invasive blood pressure prediction from ppg and rppg signals using deep learning. *Sensors*, 2021.
- [8] In Cheol Jeong and Joseph Finkelstein. Introducing contactless blood pressure assessment using a high speed video camera. *Journal of medical systems*, 2016.
- [9] Chenbin Liu Francis Tsow Hui Yu Dangdang Shao, Yuting Yang and Nongjian Tao. Noncontact monitoring breathing pattern, exhalation flow rate and pulse transit time. *IEEE Transactions on Biomedical Engineering*, 2014.
- [10] Daniel McDuff. Camera measurement of physiological vital signs. 2021. <https://arxiv.org/abs/2111.11547>.

- [11] Cuculo V D'Amelio A Grossi G Lanzarotti R Mortara E. Boccignone G, Conte D. pyvhr: a python framework for remote photoplethysmography. 2022. doi: 10.7717/peerj-cs.929. <https://doi.org/10.7717/peerj-cs.929>.
- [12] Hadon Nash Chris McClanahan Esha Ubweja Michael Hays Fan Zhang Chuo-Ling Chang Ming Guang Yong Juhyun Lee Wan-Teh Chang Wei Hua Manfred Georg Matthias Grundmann Camillo Lugaresi, Jiuqiang Tang. Mediapipe: A framework for building perception pipelines. 2019. <https://arxiv.org/abs/1906.08172>.
- [13] J Stuart Nelson Wim Verkruyse, Lars Svaasand. Remote plethysmographic imaging using ambient light. 2008. doi: 0.1364/oe.16.021434. <https://pubmed.ncbi.nlm.nih.gov/19104573/>.
- [14] Sander Stuijk Gerard de Haan Wenjin Wang, Albertus C den Brinker. Algorithmic principles of remote ppg. 2017. doi: 10.1109/TBME.2016.2609282. <https://pubmed.ncbi.nlm.nih.gov/28113245/>.
- [15] Vincent Jeanne Gerard de Haan. Robust pulse rate from chrominance-based rppg. 2013. doi: 10.1109/TBME.2013.2266196. <https://pubmed.ncbi.nlm.nih.gov/23744659/>.
- [16] Rosalind W Picard Ming-Zher Poh, Daniel J McDuff. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. 2010. doi: 10.1364/OE.18.010762. <https://pubmed.ncbi.nlm.nih.gov/20588929/>.
- [17] Paola A. Lanfranchi and Virend K. Somers. Chapter 20 - cardiovascular physiology: Autonomic control in health and in sleep disorders. In Meir H. Kryger, Thomas Roth, and William C. Dement, editors, *Principles and Practice of Sleep Medicine (Fifth Edition)*, pages 226–236. W.B. Saunders, Philadelphia, fifth edition edition, 2011. ISBN 978-1-4160-6645-3. doi: <https://doi.org/10.1016/B978-1-4160-6645-3.00020-7>. URL <https://www.sciencedirect.com/science/article/pii/B9781416066453000207>.
- [18] <https://neuropsychology.github.io/NeuroKit/functions/hrv.html>.
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

- [20] Sarah Guido Andreas C. Müller. Introduction to machine learning with python: A guide for data scientists. 2016.
- [21] Sebastian Ruder. An overview of proxy-label approaches for semi-supervised learning. <https://ruder.io/semi-supervised/>.
- [22] R.A. Fisher. The use of multiple measurements in taxonomic problems. 1936.
- [23]
- [24] Mohammad kachuee. Cuff-less blood pressure estimation. <https://www.kaggle.com/datasets/mkachuee/BloodPressureDataset>.
- [25] Yoon SB Yang SM Park D Jung CW Lee HC, Park Y. Vitaldb, a high-fidelity multi-parameter vital signs database in surgical patients. 2022. doi: 10.1038/s41597-022-01411-5.
- [26] Zheng Zhang, Jeff M. Girard, Yue Wu, Xing Zhang, Peng Liu, Umur Ciftci, Shaun Canavan, Michael Reale, Andy Horowitz, Huiyuan Yang, Jeffrey F. Cohn, Qiang Ji, and Lijun Yin. Multimodal spontaneous emotion corpus for human behavior analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [27] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>.
- [28] George S et al. Stergiou. A universal standard for the validation of blood pressure measuring devices: Association for the advancement of medical instrumentation/european society of hypertension/international organization for standardization (aami/esh/iso) collaboration statement. *Journal of hypertension*.