Università degli studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in

Scienze Statistiche

Relazione Finale

L'effetto delle misure di contenimento a Venezia: riconoscimento e analisi di sequenze di immagini

Relatore Dott. Bruno Scarpa
Dipartimento di Scienze Statistiche

Laureando Andrea De Vita
Matricola N 1241716

Anno Accademico 2021/2022

Indice

Elenco delle figure

Elenco dei codici

Elenco delle tabelle

Introduzione

31 gennaio 2020.
"Il presidente del consiglio, Giuseppe Conte, conferma i primi due casi di contagio riscontrati in Italia. Il primo ministro dichiara l'emergenza sanitaria nazionale".

L'avvento del Covid-19 all'inizio del 2020 ha sconvolto la vita degli italiani e di ogni cittadino del mondo, costringendo ciascuno di noi a cambiare le proprie abitudini. Da un giorno all'altro ci siamo ritrovati costretti a rimanere a casa, senza la possibilità di andare a lavoro o a scuola e costretti a fare allenamento da casa. La pandemia ci ha sconvolto e resterà per sempre nell'immaginario comune, un periodo storico che verrà ricordato.

Lo scopo di questo elaborato è studiare, anche se in misura ristretta, che effetti ha avuto il persistere della paura del contagio nelle persone, congiuntamente alla istituzione, lungo il corso dei mesi interessati dall'emergenza sanitaria, delle diverse zone di rischio, sulle abitudini e sui comportamenti dei cittadini italiani, in particolare coloro residenti nella laguna di Venezia. Per fare ciò, si è deciso di analizzare le fotografie della stessa "calle" di Venezia scattate, in diverse fasce orarie, lungo tutto il periodo di convivenza con il Coronavirus. Attraverso l'utilizzo di tecniche per il riconoscimento di immagini è stato possibile processare un gran numero di foto e studiare l'andamento nel tempo del numero di persone presenti lungo la via, focalizzandosi sulla formazione di gruppi, analizzando eventuali differenze riscontrate sulla base di una diversa situazione di rischio alla quale era sottoposto il comune di Venezia.

Nel primo capitolo di questo elaborato è analizzato il funzionamento delle *Convolutional Neural Network* e le logiche che risiedono alla base di questa

architettura. Inoltre vengono introdotte le tecniche per il riconoscimento delle immagini utilizzate ai fini dell'analisi.

Il secondo capitolo fornisce una vista sulle caratteristiche principali delle fotografie oggetto di analisi, trattando in breve la fase di *pre-analisi* e *analisi* di queste all'interno di Python, con un riferimento alle modalità di applicazione delle tecniche descritte nella prima parte del testo.

Infine, nel terzo e nel quarto capitolo è presentato lo svolgimento delle analisi su, rispettivamente, il numero di persone e il numero di gruppi, esponendo le caratteristiche più rilevanti dei due dataset, congiuntamente alla modellazione proposta ai risultati ottenuti.

Capitolo 1

Convolutional Neural Network e EfficientDet

Per la prima parte dell'analisi si è ricorso all'utilizzo di tecniche per il riconoscimento di immagini che saranno caratterizzate nel corso del presente capitolo.

1.1 Convolutional Neural Network

La *Convolutional Neural Network*, anche conosciuta come CNN o ConvNet, è una classe di reti neurali specializzata nell'analisi di dati che presentano una struttura a griglia, come per esempio le immagini. Un'immagine digitale è una rappresentazione bidimensionale di dati visivi. Essa contiene una serie di pixel organizzati in una matrice all'interno della quale ogni cella è costituita da un valore che denota quanto luminoso e di che colore è il pixel di riferimento.

Una ConvNet è capace di catturare le dipendenze spaziali e temporali di un'immagine con estrema accuratezza, attraverso l'applicazione di filtri differenti. La particolare architettura della rete permette un adattamento all'immagine efficace, grazie ad una riduzione nel numero dei parametri coinvolti e alla possibilità di riutilizzo delle matrici dei pesi.

Se invece utilizzassimo una rete neurale tradizionale questa prenderebbe l'immagine, rappresentata da una matrice di pixel, e la trasformerebbe in un vettore, considerando i valori dei pixel come *features* per la previsione dei numeri nell'immagine. Questo causerebbe, da una parte, la totale perdita dell'organizzazione dei pixel all'interno dell'immagine, e dall'altra, l'utilizzo di una quantità enorme di parametri che condurrebbe velocemente a casi di sovradattamento al crescere delle dimensioni dell'immagine (Gupta 2017).

1.1.1 Architettura delle CNN

L'architettura di una CNN Ghosh et al. 2020 riprende lo schema di connessione all'interno del cervello umano ed è stato inspirato dall'organizzazione della corteccia visiva. I singoli neuroni rispondono agli stimoli solo in una regione ristretta del campo visivo, conosciuto come "campo recettivo". Una raccolta di tali campi si sovrappone per ricoprire l'intera area visiva.

La struttura base di una rete neurale convoluzionale riprende lo schema appena descritto ed è composta dall'alternarsi di *Convolutional Layers*, *Pooling Layers* e *Fully Connected Layers*.

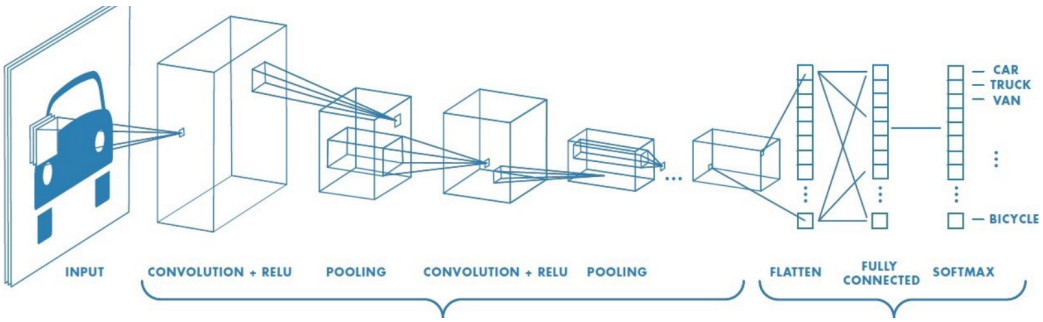

Figura 1.1: Architettura di una CNN (Saha 2018)

Convolutional layer + ReLU

Lo strato convoluzionale (in inglese *Convolutional Layer*) è la componente iniziale della rete ed è la più importante, in questo strato avviene ciò che si definisce "convoluzione" da cui prende il nome la CNN. L'obiettivo di questa

sezione della rete è quello di estrarre *features* dall'immagine di input (Tsang 2019).

In matematica, in particolare nell'analisi funzionale, la convoluzione è un'operazione tra due funzioni in una variabile, che consiste nell'integrare il prodotto tra la prima e la seconda, traslata di un certo valore. La Figura 1.2 aiuta a comprendere meglio questa operazione in realtà molto semplice, attraverso un esempio nel caso di un input bidimensionale.

In questo esempio viene passata una matrice di dimensione 2×2 (chiamata *filtro* oppure *kernel*) lungo tutta la superficie dell'input, rappresentato da una matrice 4×4, con spostamenti sia orizzontali che verticali compiendo un prodotto scalare ad ogni passaggio tra il filtro e la porzione della matrice interessata, generando così uno scalare da ciascuna di queste operazioni. Il processo continua finché tutta la superficie non è stata completamente ricoperta; questo produce quella che viene chiamata mappa di attivazione (o *features map*, la matrice a destra nella Figura 1.2), la quale costituisce di fatto il primo strato nascosto della rete.

Nel caso di immagini con canali multipli (come per esempio immagini RGB), il filtro ha la stessa profondità dell'immagine di input. Il prodotto tra matrici sarà compiuto tra ciascun *kernel* ed il rispettivo canale di input; i diversi risultati vengono poi sommati tra di loro per andare a comporre la mappa di attivazione, come mostrato nella Figura 1.3.

Dagli esempi appena mostrati possiamo ricavare due proprietà fondamentali delle *Convolutional Neural Network*:

- *Sparse Connectivity*: In una rete neurale *fully connected* ciascun neurone di ogni strato è connesso con ogni neurone dello strato successivo mentre nella CNN è presente un numero minore di pesi tra due strati consecutivi, rappresentati dai filtri. Come risultato, il numero di pesi di cui necessitiamo per l'intero processo è minore e di conseguenza anche lo spazio di memoria dove immagazzinarli.

- *Weight Sharing*: Nelle CNN non ci sono filtri dedicati tra due neuroni di strati adiacenti e invece i neuroni appartenenti alla stessa mappa di attivazione condividono sempre lo stesso insieme di pesi, ossia quelli del filtro che li ha generati.

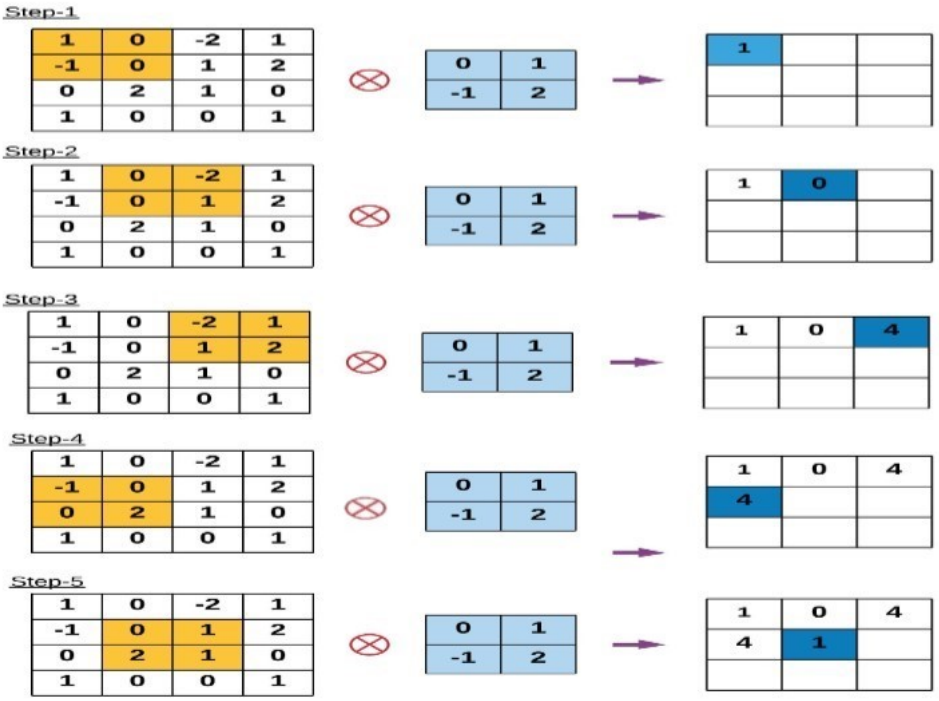

Figura 1.2: Operazione di convoluzione ()Ghosh et al. 2020)

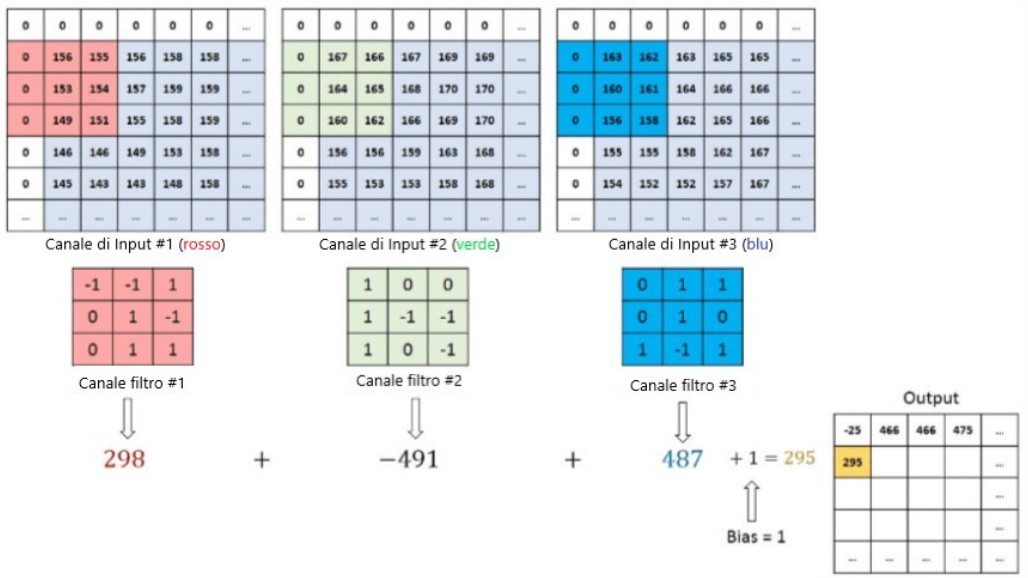

Figura 1.3: Operazione di convoluzione nel caso di immagini RGB (Saha 2018)

Finora abbiamo sempre visto la convoluzione di un singolo filtro ma generalmente un *Convolutional Layer* è formato da un set di filtri molto più numeroso, tutti con la stessa estensione spaziale, e pertanto vengono prodotte mappe di attivazione in egual numero, ciascuna delle quali fornisce la risposta del relativo filtro in ogni posizione spaziale. La loro concatenazione lungo la terza dimensione produce l'output dello strato di convoluzione; se per esempio vengono applicati dieci filtri differenti all'immagine, ognuno dei quali alla ricerca di caratteristiche diverse nel volume di input, saranno di conseguenza prodotte in output dieci mappe di attivazione tra di loro concatenate.

Il risultato derivante dall'operazione di convoluzione è controllato da due iperparametri: numero di filtri (di cui è già stato discusso precedentemente), *stride* e *zero-padding*.

- *Stride*: definisce il numero di pixel con cui viene traslato il filtro ad ogni spostamento. Più questo valore sarà alto, maggiori saranno gli spostamenti e di conseguenza il volume di output sarà minore.

- *Zero-padding*: questo può essere utile nel caso in cui si voglia preservare la dimensione dell'immagine di input, inserendo un perimetro di zeri intorno alla matrice di partenza (come è stato fatto per le matrice nella Figura 1.3).

Le dimensioni spaziali dell'immagine di output possono essere misurate tramite la seguente formula:

$$O = ([W - F + 2 \cdot P]/S) + 1 \tag{1.1}$$

dove W corrisponde alla dimensione del volume di input, F alla dimensione del filtro, P al numero di *padding* applicato ed S al numero di *stride*. La formula ritornerà le dimensioni, larghezza e altezza, del volume di output (O), con profondità pari al numero di filtri utilizzati.

Dopo ogni strato di convoluzione viene inserita un'operazione non lineare chiamata ReLU (*Rectified Linear Unit*), mostrata in Figura 1.4, che ha come obiettivo quello di introdurre la non linearità all'interno della CNN (Wu 2017). La funzione ReLU restituisce il valore massimo tra l'input e lo zero,

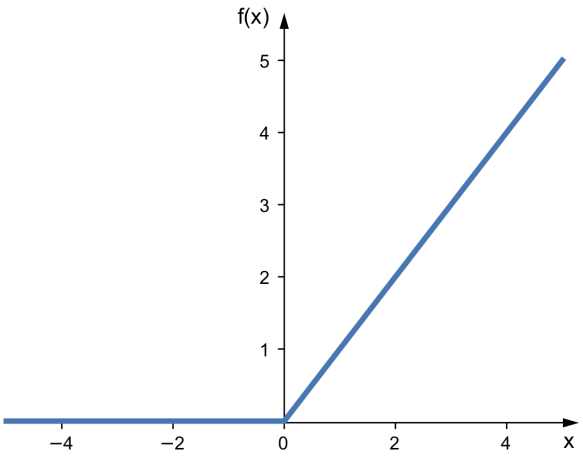

Figura 1.4: Funzione ReLU

sostituendo così, tutti i pixel negativi della mappa di attivazione con il valore nullo (formula 1.2).

$$ReLU(z) = max(0, z) \tag{1.2}$$

Esistono altre funzioni non lineari che possono essere utilizzate nell'architettura di una *Convolutional Neural Network* come la funzione sigmoide e la funzione tangente iperbolica. La ReLU ha, però, due importanti vantaggi:

1. È molto semplice da calcolare, dato che coinvolge solo un paragone tra il valore di input e lo zero

2. La sua derivata assume sempre valore pari ad uno o pari a zero

L'ultimo, in particolare, ha importanti implicazioni nell'algoritmo di *backpropagation*, utilizzato nel processo di addestramento, dato che, il calcolo del gradiente di un neurone in questa fase, ha un costo computazionale praticamente nullo.

Inoltre, la funzione ReLU risolve il problema della dissolvenza del gradiente, non andando a saturare i neuroni. Questo problema si riferisce alla tendenza del gradiente di avvicinarsi allo zero per valori di input elevati, che nel caso della ReLU non si manifesta dato che la sua derivata rimane costante

e pari ad uno.

$$\frac{ReLU(z)}{\partial z} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0. \end{cases} \tag{1.3}$$

Pooling layer

Lo strato di *Pooling* ha lo scopo di ridurre la dimensionalità della *feature map*, derivante dall'operazione di convoluzione, e al contempo conservare le informazioni più importanti di ognuna di esse. Questa operazione porta, quindi, dei benefici, sia in termini di potenza computazionale richiesta nel processare i dati, sia nell'operazione di estrazione delle *feature* dominanti, rendendo la rete invariante a piccole trasformazioni e distorsioni.

Esistono due tipi di *Pooling*: Max Pooling e Average Pooling (Figura 1.5). Il Max Pooling restituisce il valore massimo dalla porzione dell'immagine ricoperta dal filtro, mentre l'Average Pooling restituisce il valor medio della sezione interessata. Il Max Pooling è la tipologia più utilizzata, dal momento che riesce anche a compiere un'azione di "soppressione del rumore".

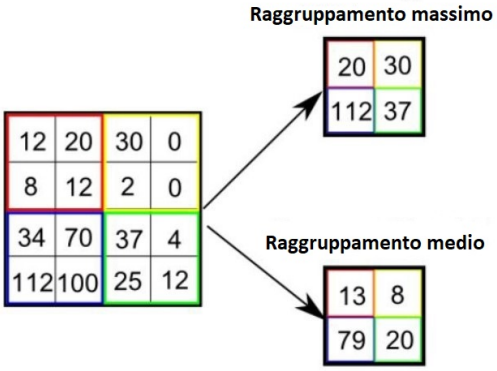

Figura 1.5: Operazione di *Pooling*

Il *Convolutional Layer* e il *Pooling Layer* si alternano tra di loro e la numerosità di questi strati può dipendere dalla complessità delle immagini. In aggiunta, il numero di queste due componenti può essere aumentato al fine di riuscire a catturare dettagli più piccoli, con, però, un maggior costo dal punto di vista della potenza computazionale richiesta.

Fully connected layer

L'ultima parte di ogni CNN (usata per la classificazione) consiste in un *fully connected layer*, dove ogni neurone all'interno di uno strato è connesso con ciascun neurone appartenente allo strato precedente e a quello successivo. La funzione principale di tale strato è quella di eseguire una sorta di raggruppamento delle informazioni ottenute nelle sezioni precedenti, esprimendole attraverso un numero, che sarà necessario per la classificazione finale.

1.1.2 Architettura generale della rete

Fino ad ora sono stati elencati i singoli strati che vengono impiegati all'interno dell'architettura di una CNN. Come per le reti neurali tradizionali, anche qui non esiste una linea guida per la scelta degli iperparametri o della sequenza di strati da adottare, né tantomeno per la decisione del numero di ciascuno di essi.

Ricapitolando ed integrando quanto appena descritto, il processo di addestramento di una *Convolutional Neural Network* si articola nei seguenti passaggi (Karn 2016):

- Primo passo: Si inizializzano tutti i filtri e gli altri parametri con valori casuali.
- Secondo passo: L'immagine di input viene fatta passare attraverso il primo strato convoluzionale. I filtri applicati aiutano ad estrarre le *feature* rilevanti dall'immagine per poi passarle agli strati successivi. Tra un strato convoluzionale e l'altro possono essere posti strati di *pooling*, in modo da ridurre il numero di parametri. Diversi *convolutional layer* e *pooling layer* sono aggiunti prima che venga effettuate la previsione; più la profondità della rete aumenta maggiori sono le *feature* specifiche che vengono estratte.
- Terzo passo: Le *feature* ricavate dallo step precedente sono passate all'ultimo strato, il *fully connected layer*, che restituisce le probabilità per ciascuna classe. Nel primo ciclo di addestramento le probabilità stimate saranno casuali dal momento che i pesi sono stati inizializzati in modo casuale.

- Quarto passo: Viene calcolato l'errore totale, sommato per tutte le classi presenti:

$$ErroreTotale = \sum_{k=1}^{K} 1/2 \cdot (p(k) - \hat{p}(k))^2 \tag{1.4}$$

- Quinto passo: Si utilizza l'algoritmo di *backpropagation* per calcolare i gradienti dell'errore, rispettivamente ai pesi della rete, e si usa il *Gradient Descent* per aggiornare i filtri ed i pesi, in modo da minimizzare l'errore di stima. I pesi vengono aggiustati in proporzione al loro contributo all'errore totale. Parametri come il numero di filtri, dimensione delle matrici o numero di strati sono stati fissati precedentemente e non sono quindi coinvolte nel processo di addestramento.

- Sesto passo: I passaggi 2-5 vengono ripetuti per ciascuna immagine del dataset.

1.2 EfficientDet e EfficientNet

In questa sezione saranno trattate le principali funzionalità e caratteristiche di nuove architetture per il rilevamento di oggetti nelle immagini, che prendono il nome di "EfficientDet", a loro volta costruiti sulle "EfficientNet", che verrano successivamente utilizzate per l'identificazione delle persone nella strada di Venezia considerata.

1.2.1 EfficientNet

Le ConvNets vengono comunemente sviluppate con un "budget" di risorse fissato a priori, che viene poi ridimensionato per una migliore precisione, se vengono rese disponibili risorse aggiuntive. Le dimensioni principali di una CNN che ne regolano il funzionamento sono: l'ampiezza, data dal numero di filtri applicati all'immagine nello strato convoluzionale, la profondità, pari al numero di *Convolutional Layer* e *Pooling Layer* che si alternano tra loro, e la risoluzione dell'immagine. Molto spesso, per raggiungere un'accuratezza migliore nella previsioni, questi parametri vengono scalati singolarmente.

Quello che viene fatto nelle *EfficientNets* (Tan e Le 2019) è il cosiddetto *compound scaling method*, ossia una tecnica che scala in modo uniforme le tre dimensioni della rete mediante un insieme di coefficienti fissati. Se, per esempio, vogliamo usare 2^N volte le risorse computazionali, è sufficiente incrementare la profondità, l'ampiezza e la risoluzione dell'immagine di, rispettivamente, α^N, β^N e γ^N. La Figura 1.6 mostra le differenze tra gli usuali metodi di *scaling* e quello proposto per lo sviluppo delle *EfficienteNets*.

Il successo di questa nuova metodologia è dovuto al fatto che, intuitivamente, immagini con una risoluzione più alta necessitano di una maggiore profondità della rete, così che un maggior numero di strati aiuti a catturare diverse *features* ciascuna delle quali formata da più pixel. Di conseguenza, è opportuno aumentare l'ampiezza della rete, in modo da catturare più *patterns* dettagliati, sull'immagine più grande, mediante una quantità più elevata di filtri.

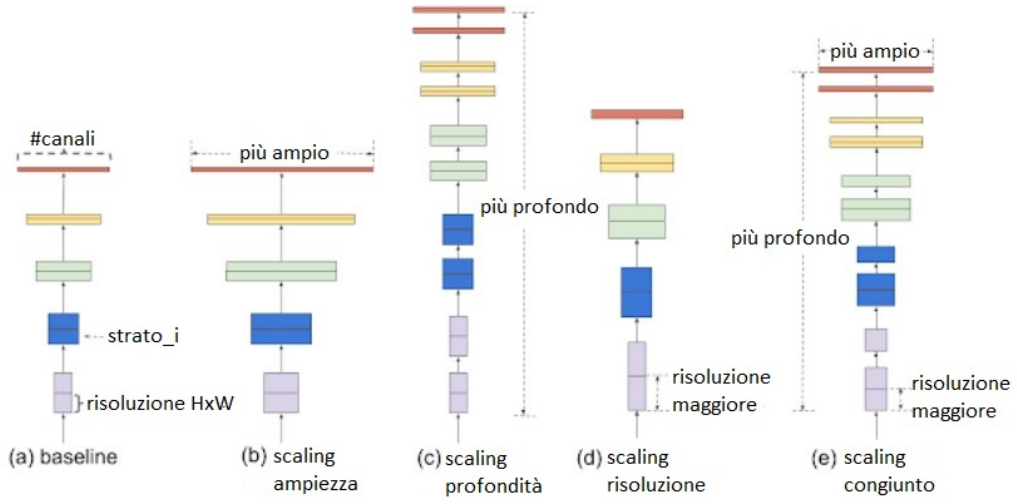

Figura 1.6: Diffenti metodi di scaling paragonati. Diversamente dai metodi di scaling tradizionali (b)-(d) che si concentrano su una sola dimensione della rete, il metodo della EfficientNet scala in modo uniforme tutte le dimensioni contemporaneamente (Tan e Le 2019).

L'efficacia del modello è dovuto, oltretutto, alla rete di base. Senza addentrarsi ulteriormente nelle specificità di questa architettura, questo elaborato

si limiterà a mostrare brevemente le caratteristiche principali dello schema della *baseline network*: questa utilizza una convoluzione del tipo *Mobile Inverted Bottleneck*(MBConv) che riesce ad ottimizzare sia l'accuratezza che l'efficienza (in termini di FLOPS, *FLoating point Operations Per Second*, numero di operazioni in virgola mobile eseguite in un secondo dalla CPU), attraverso uno schema più semplice orientato ad una maggiore facilità di *scaling* e generalizzazione. La Tabella 1.1 riassume il modello della rete utilizzato per l'implementazione della *EfficientNet-B0*.

Stage	**Operatore**	**Risoluzione**	**N° Filtri**	**N° Strati**
1	Conv3x3	224 × 224	32	1
2	MBConv1,k3x3	112 × 112	16	1
3	MBConv6,k3x3	112 × 112	24	2
4	MBConv6,k5x5	56 × 56	40	2
5	MBConv6,k3x3	28 × 28	80	3
6	MBConv6,k5x5	14 × 14	112	3
7	MBConv6,k5x5	14 × 14	192	4
8	MBConv6,k3x3	7 × 7	320	1
9	Conv1x1 e Pooling e FC	7 × 7	1280	1

Tabella 1.1: EfficientNet-B0 baseline network. Ogni riga descrive una fase del processo con relativo numero di strati e numero di mappe di attivazione in output.

1.2.2 EfficientDet

L'*EfficientDet* c è una nuova famiglia di metodi per il rilevamento di oggetti nelle immagini, costruita a partire dalla rete neurale *EfficientNet*, descritta nella sezione precedente.

In generale, gli *object detectors* hanno tre componenti principali:

- Una *backbone* che estrae le *feature* dall'immagine di input.
- Una *feature network* che prende i diversi livelli delle *feature* dalla *backbone* come input e ritorna una lista di *feature* fuse tra loro, che rappresentano le caratteristiche salienti dell'immagine.

- La *class/box network* che utilizza le *feature* fuse per prevedere a quale classe e a quale posizione spaziale appartengono i vari oggetti nell'immagine.

In merito alla rete di base (*backbone network*), questo modello implementa una rete *EfficientNet*, che risulta molto più efficiente di altre tecniche come *ResNets* e *AmoebaNet*, sia sotto il livello di accuratezza sia per la potenza computazionale richiesta.

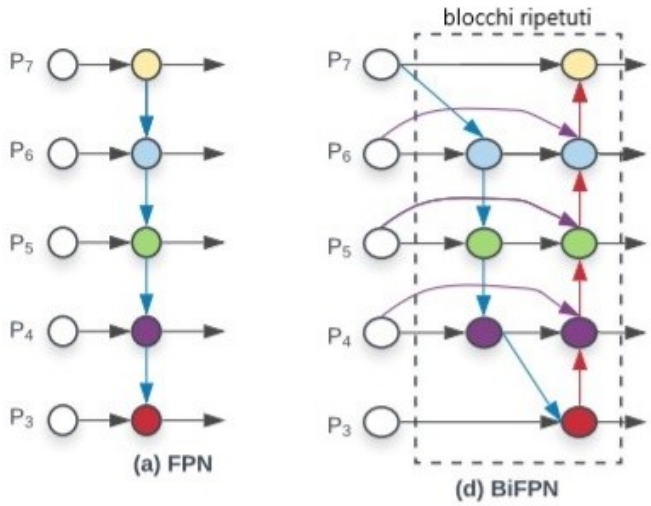

Figura 1.7: Confronto tra FPN e BiFPN. La BiFPN permette sia un flusso di tipo "top-down", come la FPN, sia un flusso "bottom-up" (Tan, Pang e Le 2020).

Una ulteriore ottimizzazione che è stata proposta riguarda la *feature network*. Mentre la maggior parte dei metodi conosciuti utilizza la FPN (*Feature Pyramid Network*), che semplicemente applica una fusione delle *feature* dall'alto verso il basso, l'*EfficientDet* propone una nuova *feature network* bidirezionale (BiFPN), che, invece, rende possibile il fluire delle informazioni, sia dall'alto verso il basso che dal basso verso l'alto (Figura 1.7). Inoltre, i metodi precedenti utilizzavano tutte le *feature* di input senza alcuna distinzione, nonostante fossero caratterizzate da una diversa risoluzione. Si è osservato, però, che *feature* con diversa risoluzione contribuiscono alle *feature* di output in maniera differente. Quindi, per affrontare il problema ed incrementare

maggiormente l'efficienza del processo, la BiFPN aggiunge dei pesi per ogni *feature* in input, basandosi sulla loro diversa risoluzione, così da permettere alla rete di imparare l'importanza di ciascuna di esse.

Un ultimo perfezionamento coinvolge l'utilizzo del "compound scaling" per ogni componente della rete: *backbone*, *feature network* e *box/class prediction*. Come visto per le *EfficientNets*, viene proposto un coefficiente composto in modo da poter aumentare congiuntamente tutte le dimensioni, e rendere estremamente più semplice la determinazione di quale valore deve assumere il fattore di scala, dato il vincolo di risorse disponibile. In Figura 1.8 è riassunto lo schema di funzionamento della *EfficientDet*.

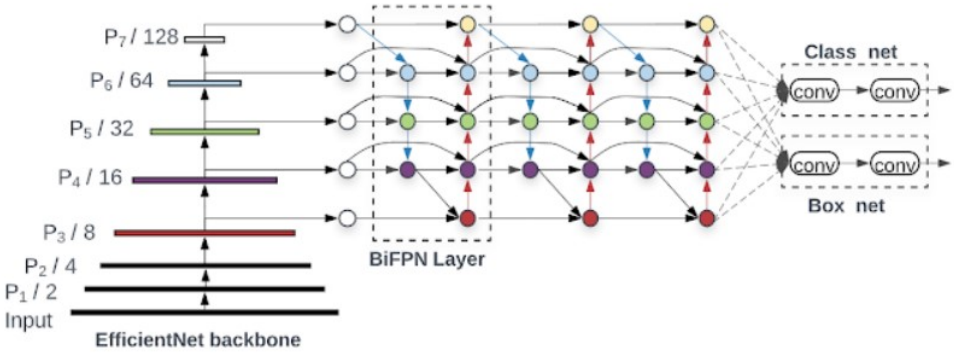

Figura 1.8: Architettura della *EfficientDet* (Tan, Pang e Le 2020).

Capitolo 2

La raccolta di immagini

2.1 Raccolta di immagini

La raccolta di immagini si compone di 854 foto scattate tra l'8 marzo 2020 ed il 23 agosto 2021 in Strada Nova a Venezia, per la precisione nella tratta che va da Campo San Felice fino ad arrivare a Campo Santi Apostoli (esempio in Figura 2.1). Le immagini non presentano una regolarità temporale; infatti, può accadere che in alcune giornate venga scattata un'unica foto, mentre in altre è possibile arrivare fino ad un massimo di tre fotografie, una per la mattina, una per la fascia pomeridiana ed una per la sera, per altre ancora, invece, non si ha a disposizione alcuna immagine. In aggiunta a quanto appena detto, tra il 5 luglio 2020 e l'1 novembre del medesimo anno, non è disponibile alcuna immagine, derivante anche dal fatto che, nel periodo di cui sopra, si è potuto assistere ad una mitigazione delle misure restrittive messe in atto.

2.2 Tensorflow in Python

Tramite l'utilizzo della libreria *TensorFlow* in Python è possibile applicare diverse metodologie per il riconoscimento di immagini. TensorFlow è una libreria software *open source* sviluppata da Google nel progetto "Google Brain (AI)" ed utilizzata per la creazione di app nell'apprendimento automatico

Figura 2.1: Esempio di una foto appartenente al dataset.

(*machine learning*). *TensorFlow* è un *toolkit* matematico simbolico che esegue una varietà di attività, tra cui l'adattamento della rete neurale profonda utilizzando il flusso di dati e la programmazione differenziabile.

Per il caso in esame è stata inizialmente utilizzata una serie di modelli reperiti all'interno del "TensorFlow Hub", uno spazio dove è possibile navigare alla ricerca di modelli pre-stimati e dataset per problemi quali il riconoscimento di audio e video, analisi di testi, fino a quelli relativi al riconoscimento di immagini. Per lo scopo della tesi la modalità di modellazione è stata ricercata all'interno del TensorFlow Hub relativo al rilevamento di oggetti (*TensorFlow Hub Object Detection*), contenente una serie di modelli per la rilevazione di oggetti, adattamento all'insieme di dati "COCO 2017". Il dataset COCO (Common Objects in Context) è stato rilasciato nel 2014 ed è spesso utilizzato per paragonare la prestazione di algoritmi per problemi di *Object Detection* e per adattare i diversi tipi di modelli. COCO 2017 è composto da 330000 immagini, divise in insieme di stima, verifica e convalida, di cui più di 200000 etichettate con 80 categorie di oggetti, come per esempio persone, macchine, cibi ecc., fino ad un totale di 1.5 milioni di oggetti.

Prima di elaborare le foto mediante queste tecniche, sono state compiute delle operazioni di *pre-analisi* su ciascuna di queste. Innanzitutto, tutte le fotografie appartenenti al dataset sono ritagliate nelle medesime coordinate, così da considerare in ognuna di esse la stessa porzione di strada; per la precisione, in questa analisi si è voluto tenere conto solamente delle persone presenti all'interno del Campo San Felice, per due motivi principali: il primo è dovuto all'eventualità che potessero sorgere delle complicazioni legate al riconoscimento delle figure con una distanza dal punto di scatto molto elevata, mentre il secondo è legato al fatto che, utilizzando questa metodologia, si riducono le problematiche relative alla misurazione delle distanze interpersonali, che saranno di grande rilevanza nell'analisi dei gruppi, trattata nel capitolo 4.

Dopo aver elaborato le immagini, queste sono ricondotte automaticamente ad una stessa risoluzione prefissata, così da poter mantenere le stesse proporzioni tra le immagini evitando che eventuali differenze possano inficiare la misura delle distanze. In Figura 2.2 è mostrato un esempio di un'immagine sottoposta allo step di *pre-analisi*.

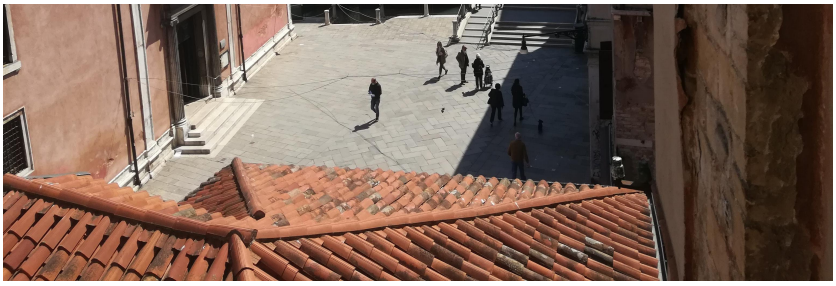

Figura 2.2: Immagine in Figura 2.1 dopo gli step di *pre-analisi*.

A questo punto, il dataset è composto da fotografie rappresentanti la stessa porzione dello spazio e aventi la medesima risoluzione, ed è quindi pronto per essere sottoposto all'applicazione dei modelli di *object detection*, contenuti all'interno del "TensorFlow Hub Object Detection". Alle immagini sono stati applicati numerosi modelli, tra cui *Faster-RCNN*, *Retinanet* e *Mask-RCNN*, per i quali si è poi confrontata l'accuratezza nel riconoscimento delle figure umane e infine, attraverso un'analisi qualitativa dei risultati, deciso di

utilizzare quello che si adattava in modo più soddisfacente alle foto del dataset. Il modello che più accuratamente è riuscito a cogliere i *detection boxes* delle persone (Figura 2.3) è l'*EfficientDet*, descritto all'interno del paragrafo 1.2 nel capitolo 1.

Figura 2.3: Immagine in Figura 2.1 dopo l'applicazione del modello *EfficientDet*.

L'applicazione del modello *EfficientDet* restituisce come risultato l'immagine di input comprensiva dei *box* di contorno di ciascuna persona riconosciuta, con la relativa accuratezza di classificazione (definita in percentuale). In aggiunta a ciò, è disponibile un dizionario contenente una serie di variabili utili ai fini delle analisi presentate nei capitoli successivi, riportate nella tabella che segue (Tabella 2.1).

Variabile	**Tipo**	**Descrizione**
num detections	int	numero di oggetti rilevati nell'immagine
detection boxes	vector	posizione spaziale dei vertici del *detection box*
detection classes	int	categoria a cui appartiene l'oggetto rilevato
detection scores	float	accuratezza della classificazione

Tabella 2.1: Variabili restituite in output successivamente all'applicazione del modello *EfficientDet*.

Tra queste, la variabile *detection classes* è utilizzata per filtrare i soli *box* con etichetta pari a *person* (la tabella con i numeri assegnati a ciascuna istanza è reperibile online), evitando in tal modo di estrarre qualsiasi altra categoria di oggetti. *Detection scores* viene utilizzata per impostare una soglia

al di sotto della quale l'oggetto riconosciuto non viene considerato; in questa fase dell'analisi la soglia di accettazione viene scelta mediante uno studio qualitativo basato sul confronto della classificazione di una serie di immagini ottenute con differenti soglie. Infine, *num detections* e *detection boxes* saranno parte integrante dei dataset utilizzati rispettivamente, nelle analisi al capitolo 3 e al capitolo 4.

In appendice A sono riportati gli *script* Python per l'applicazione delle tecniche per il riconoscimento di oggetti e per la creazione ed elaborazione dei due datasat utilizzati per la modellazione nei capitoli successivi.

2.3 Complicazioni nel riconoscimento delle persone

L'applicazione del modello *EfficientDet* non sempre restituisce un risultato ottimale in termini di accuratezza nelle previsioni. Infatti, può succedere che in alcune immagini con caratteristiche specifiche tale algoritmo non riesca ad identificare determinati soggetti, come per esempio nel caso in cui due persone siano posizionate in modo tale da mascherare la presenza di uno dei due, oppure in quelle fotografie scattate in situazioni di poca visibilità. Logicamente una tecnica automatizzata, come applicato in questa tesi, non può essere sempre infallibile nella classificazione, soprattutto per quei particolare che sarebbero difficile da cogliere anche per l'occhio umano. Inoltre, in alcune immagini, è possibile che la tecnica di *object detection* riconosca erroneamente determinati oggetti o animali, classificandoli come persone e, di conseguenza, influenzando positivamente il calcolo del numero di soggetti. Resta da evidenziare che le casistiche appena descritte, grazie al buon adattamento del modello utilizzato, si verificano con una frequenza molto ridotta e non sono quindi da considerare determinanti ai fini dell'analisi presentata successivamente in questo elaborato.

In appendice B viene approfondito questo tema, mediante l'ausilio di alcuni esempi di immagini elaborate con le caratteristiche appena descritte.

Capitolo 3

Numero di persone: analisi e modelli

Lo scopo iniziale del progetto è quello di analizzare la serie del numero di persone presenti nelle foto e riuscire a capire se effettivamente esiste una correlazione tra questo numero ed il susseguirsi dei diversi colori di zona lungo tutto il lasso di tempo preso in considerazione.

3.1 Uno *sguardo* al dataset

Il dataset generato dall'estrazione delle figure ha un numero di righe pari alla quantità di foto elaborate ed è composto da una serie di variabili riportate all'interno della Tabella 3.1.

Tutte le informazioni relative al momento temporale dello scatto della foto, sono state ricavate a partire dal nome dell'immagine, che contiene all'interno la sua data e orario. Per la codifica della variabile *parte della giornata* si è deciso di operare nel seguente modo, assegnando la modalità:

- Mattina, per le foto scattate tra le sei e le otto di mattina.
- Pomeriggio, per le osservazioni con *ora* compresa tra le undici e le sedici.
- Sera, per quelle rimanenti, a partire dalle diciassette.

Variabile	Esempio estrazione
Nome	IMG_20201230_121805
Giorno	30
Mese	12
Anno	2020
Giorno della settimana	Wednesday
Ora	12
Parte della giornata	Pomeriggio
Numero persone	9
Colore	Arancione

Tabella 3.1: Variabili presenti all'interno del dataset con relativo esempio di estrazione.

È importante che l'analisi tenga conto della variabile riferita al momento della giornata, unitamente a quella relativa al giorno della settimana, dato che, la correlazione tra la numerosità di persone ed il colore di zona potrebbe essere influenzata dalla valorizzazione di queste variabili. Si presuppone, per esempio, che nella fascia mattutina dei giorni feriali, la maggior parte della gente esca di casa per recarsi sul luogo di lavoro; di conseguenza la numerosità in questo caso è possibile non sia strettamente influenzata dalle restrizioni applicate.

Per la variabile *colore*, si è fatto riferimento al livello di rischio assegnato alla regione Veneto nel corso del periodo analizzato, mediante l'utilizzo dei colori di zona ("zona rossa", "zona arancione", "zona gialla", "zona bianca").

3.1.1 Informazioni aggiuntive al dataset

In aggiunta alle variabili estratte direttamente dalle immagini a disposizione, si è pensato di integrare un ulteriore insieme di informazioni che potrebbero risultare correlate al numero di persone e, di conseguenza, rendere l'analisi più completa. Queste variabili sono relative a:

- Temperatura media giornaliera.
- Presenza di precipitazioni nel corso della giornata.

- Orario dell'alba.
- Orario del tramonto.
- Numero di contagi del giorno precedente.

Le temperature sono ricavate a partire da misurazioni giornaliere medie effettuate presso la stazione dell'istituo Cavanis di Venezia. Lo stesso vale per il dato sulla misura di precipitazione, espressa in millimetri, che è in egual modo rilevato presso la stazione sopra citata. Per la variabile delle precipitazioni è stata applicata una trasformazione che la rendesse dicotomica, dove il valore 0 indica l'assenza di precipitazioni lungo tutto il corso della giornata, mentre l'1 suggerisce che i millimetri di pioggia verificatesi sono stati in quantità non nulla. Entrambe queste informazioni, temperatura e precipitazioni, sono reperibili all'interno del sito dell'ARPAV (Agenzia Regionale per la Prevenzione e Protezione Ambientale del Veneto).

Per quanto riguarda gli orari di alba e tramonto si è fatto semplicemente riferimento a dati recepiti da una fonte online. Queste due variabili sono state poi codificate in fattori con sei modalità per l'alba e cinque per il tramonto, al fine di ottenere una distribuzione omogenea delle osservazioni in ciascuna classe.

Infine, il numero dei contagi, come per alba e tramonto, è stato ricavato da un sito internet, che riporta il numero di contagi giornalieri a partire dal 24 febbraio 2020 per tutte le provincie italiane. Questa variabile è stata elaborata per fare in modo che ogni riga del dataset contenesse il dato dei contagi del giorno precedente, che potrebbe avere un effetto sulla variabile risposta del numero di persone.

I dati appena descritti sono stati estratti per il periodo che va dal primo marzo fino a fine agosto, in modo tale da coprire interamente la fascia temporale di interesse. Le informazioni vengono quindi caricate congiuntamente all'interno di un file Excel, sul quale viene aggiunta una colonna che riporta le date comprese nell'intervallo appena identificato. A questo punto, il file così costruito viene caricato all'interno dell'ambiente di lavoro in R e, attraverso l'utilizzo della chiave della data, può essere affiancato al dataset già esistente mantenendo soltanto le osservazioni necessarie. Infine, dato che

esistono più foto scattate lo stesso giorno, può succedere che alcune di queste informazioni vengano ripetute su più righe.

3.2 Analisi preliminare

Prima di procedere con l'analisi vera e propria del fenomeno, ci si vuole dunque soffermare sulla presentazione dei dati a disposizione, descritti nella sezione precedente, enfatizzando alcune delle loro caratteristiche.

L'obiettivo è quello di riuscire a capire se, ed in che modo, i diversi colori di zona hanno influito sul comportamento degli abitanti veneziani. Per fare ciò si è innanzitutto voluto analizzare che relazione sussiste tra queste due variabili, *numero persone* e *colore*. Un primo sguardo alla distribuzione del numero di persone in relazione al colore, suggerisce la presenza di dipendenza tra queste due variabili. Si nota infatti che, Figura 3.1, il numero medio (pallino azzurro) e la mediana (linea nera) crescono al passare da una situazione di pericolo elevata ad una via via più moderata, come, giustamente, ci si poteva immaginare. Inoltre, si evince che per la zona gialla le osservazioni sono più disperse rispetto alle altre modalità e si assiste ad un valore anomalo che è posizionato esternamente ai baffi. La zona rossa è la classe caratterizzata dalla variabilità più ridotta tra le quattro ma anche quella con un'asimmetria a destra più accentuata (e due valori al di fuori delle bande).

Per avere una visione esaustiva dell'andamento della variabile di interesse, nel grafico in Figura 3.2, sono presentate le tre serie del numero di persone, per parte della giornata considerata, suddivise in base al colore di zona. È possibile evidenziare una differenza soprattutto per quanto riguarda la serie della mattina nei confronti di quella serale; nella prima si ha una disposizione di punti che non sembra seguire alcun andamento rivelando un ordine del tutto casuale, nella seconda, invece, l'andamento è più marcato e viene fatta luce sull'effetto del colore di zona sull'affollamento del campo. Infatti, quest'ultima, insieme alla serie pomeridiana, anche se in misura minore, presenta valori ridotti della variabile *numero persone* in corrispondenza dei periodi di zona rossa del Veneto, mentre, quando le misure restrittive si fanno meno pesanti, si nota un leggero aumento.

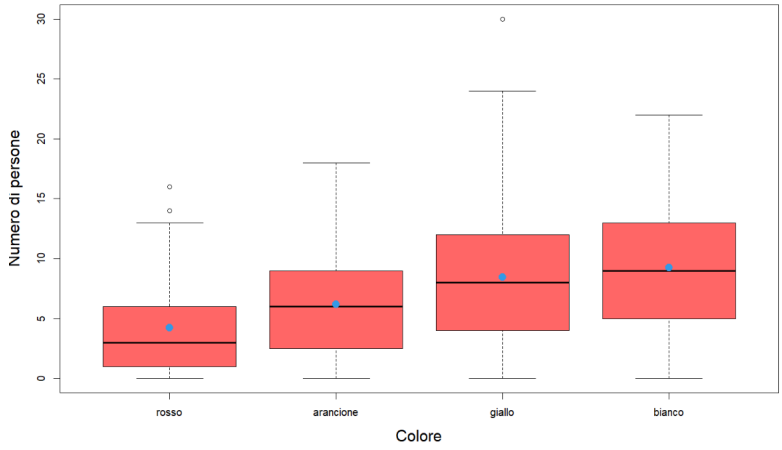

Figura 3.1: Diagramma a scatola e baffi del numero di persone identificate in un'immagine per colore di zona (il pallino blu indica la media).

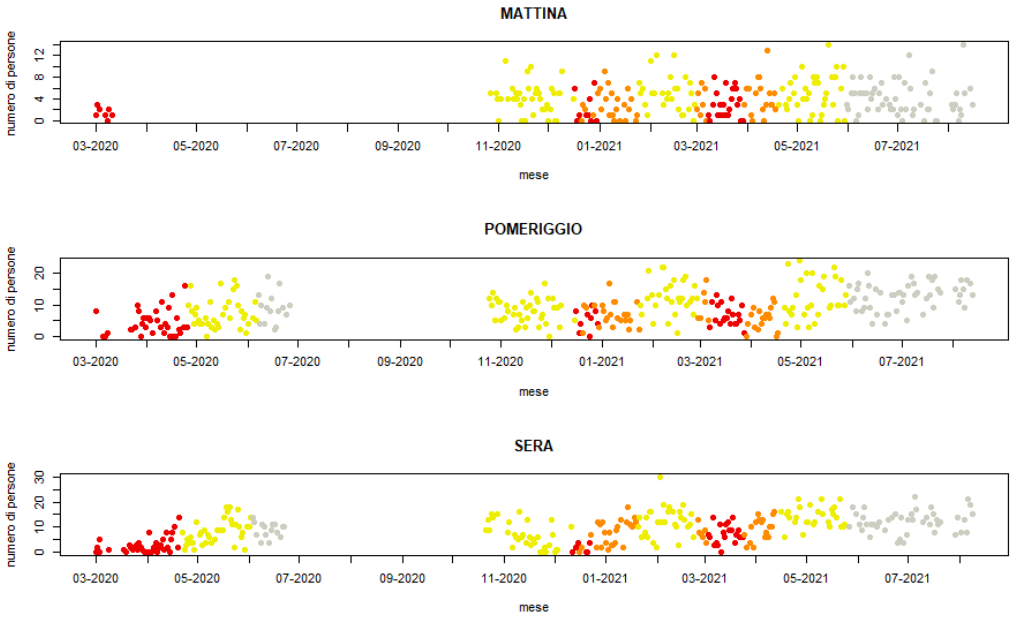

Figura 3.2: Confronto tra le serie del numero di persone in base alla parte della giornata e al colore di zona (i colori dei punti rispecchiano i colori di zona).

Un altro aspetto interessante è quello riguardante il confronto tra il numero di persone medio e il giorno della settimana, sezionando in base al momento della giornata in cui la foto viene scattata. Dal grafico in Figura 3.3 si evince che nei giorni feriali il comportamento della variabile analizzata è abbastanza omogeneo, mentre, nelle giornate di sabato e domenica si osservano valori differenti. Proprio per questa caratteristica appena rilevata si è deciso, ai fini di rendere l'analisi più parsimoniosa possibile, di fondere le modalità *"Monday", "Tuesday", "Wednesday", "Thursday", "Friday"* all'interno di un'unica classe, denominata *"Giorno feriale"*.

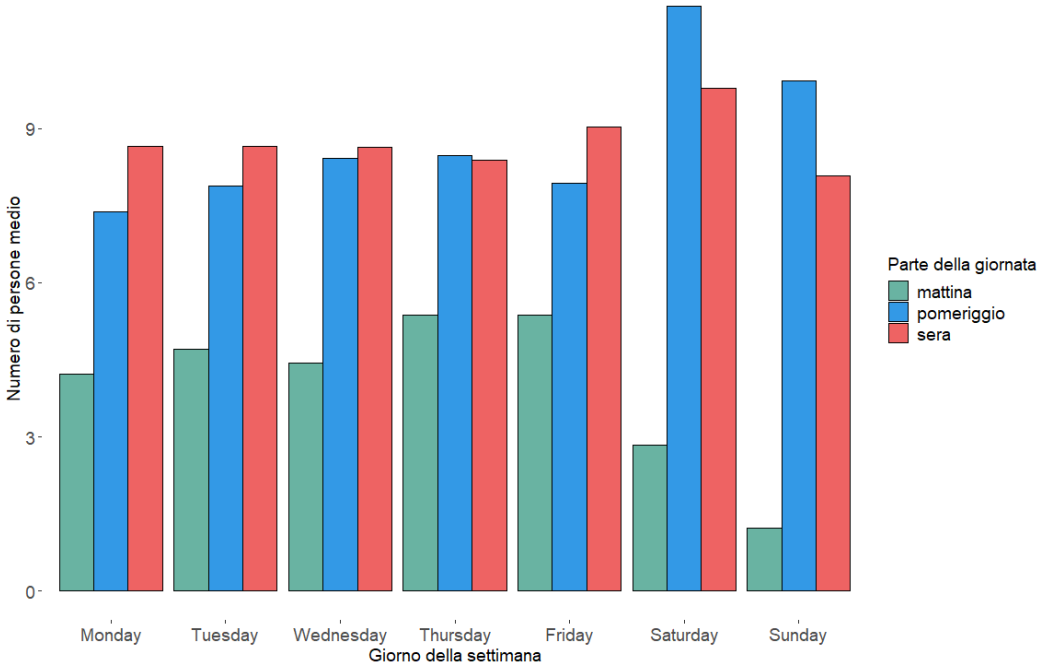

Figura 3.3: Numero di persone medio in relazione al giorno della settimana ed alla fascia giornaliera.

Relativamente alle variabili integrate in un secondo momento può essere interessante confrontare la serie del numero di contagi del giorno precedente con quella del numero di persone presenti nel campo veneziano (Figura 3.4), cercando di estrapolare una possibile dipendenza derivante dall'accortezza assunta dai cittadini in risposta ad un aumento/diminuzione del numero dei contagi nella regione Veneto. Dal grafico delle due serie sovrapposte si può

osservare come, soprattutto nel periodo da novembre 2020 a marzo 2021, le due serie presentino un andamento quasi simmetrico, comportamento che farebbe pensare ad una qualche relazione tra le due variabili, caratterizzata con più attenzione nel paragrafo successivo.

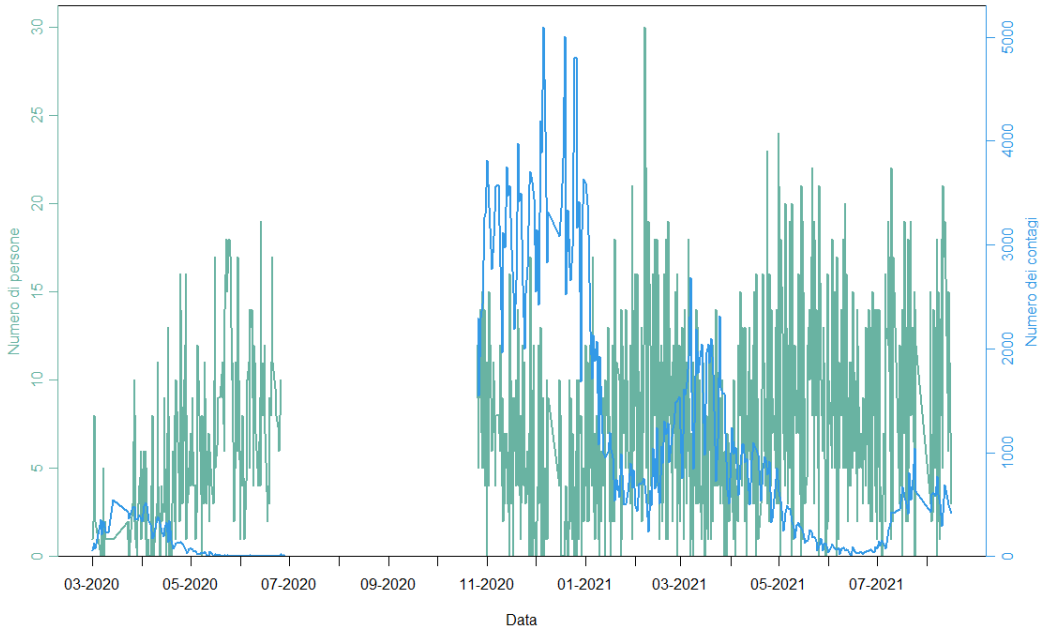

Figura 3.4: Confronto tra le serie del numero di persone e la serie del numero di contagi del giorno precedente.

3.3 Modelli

Lo scopo della modellazione è quindi quello di analizzare la serie storica del numero di persone presenti nel campo si San Felice a Venezia, e studiare la correlazione di questa, con la variabile relativa al colore di zona. Il problema, come descritto nella sezione precedente, è complesso, sia sotto il punto di vista delle diverse variabili che assumono ruoli decisivi nella caratterizzazione della variabile di interesse, sia sotto l'aspetto dell'impossibilità di conoscere il vero processo generatore dei dati, e quindi stimarlo correttamente. Queste premesse suggeriscono l'uso di metodi e tecniche flessibili, che non cerchino

di attribuire alle funzioni una struttura rigida ma che ammettono un certo grado di flessibilità, come accade all'interno della grande categoria dei metodi di stima non parametrici.

Proprio per queste motivazioni, si è deciso di optare per la stima di un modello additivo generalizzato (in breve GAM, da *Generalized Additive Model*), scelto, appunto, per la valorizzazione della relazione di interesse e per i relativi metodi di stima che sono implementati al suo interno.

Il modello additivo può essere considerato come una generalizzazione del modello lineare, in cui la funzione di regressione lineare lascia il posto alla somma degli effetti che le specifiche funzioni di lisciamento apportano alla variabile risposta.

I modelli additivi (Azzalini e Scarpa 2009) appartengono alla classi dei metodi per la stima non parametrica della regressione, che ci consentono di esaminare la relazione tra una variabile risposta ed un certo numero di variabili esplicative, senza, però, imporre una struttura particolare alla funzione che ne esprime la relazione tra le due. Questi modelli, per ovviare all'incoveniente della maledizione della dimensionalità, introducono una forma di "struttura", ovvero un modello sulla forma della funzione di regressione *f(x)*, cercando di mantenere il massimo di flessibilità. Si suppone, quindi, che per *f(x)* valga una rappresentazione del tipo:

$$f(x_1, \dots, x_p) = \alpha + \sum_{j=1}^{p} f_j(x_j) \tag{3.1}$$

dove le funzioni f_1,...,f_p sono funzioni in una variabile dall'andamento sufficientemente regolare e α è una costante.

Per arrivare alla stima delle funzioni relative al modello additivo così istituito, esiste una procedura iterativa che si appoggia ad un metodo di stima non parametrica di funzioni in una variabile per stimare le , il *backfitting*.

L'algoritmo di *backfitting* si compone dei seguenti passaggi:

- Vengono inizializzati

$$\hat{\alpha} \leftarrow \sum_{i=1}^{n} y_i/n, \hat{f}_j \leftarrow 0 \tag{3.2}$$

- Ciclo per $j = 1, 2, ..., p, 1, 2, ..., p, ...$

$$\hat{f}_j \leftarrow S([y_i - \hat{\alpha}_j - \sum_{k \neq j} \hat{f}_j(x_{ik})]_1^n) \tag{3.3}$$

$$\hat{f}_j \leftarrow \hat{f}_j - 1/n \sum_{i=1}^{n} \hat{f}_j(x_{ij}) \tag{3.4}$$

  fino a quando le funzioni $\hat{f}_j$ risultano stabili. S rappresenta un operatore di lisciamento come le spline di regressione, lisciamento o la loess.

Una generalizzazione del modello additivo (3.1) va sotto il nome di modelli additivi generalizzati (GAM). I modelli additivi generalizzati estendono i modelli additivi nello stesso modo in cui i modelli lineari generalizzati (GLM) estendono quelli di regressione lineare, e permettono di modellare gli effetti non lineari facendo uso di funzioni di lisciamento. Il modello additivo generalizzato è infatti definito dalla seguente relazione:

$$g(E\{Y|x_1, ..., x_p\}) = \alpha + \sum_{j=1}^{p} f_j(x_j) \tag{3.5}$$

Per stimare le funzioni relative ad un modello di tipo GAM si utilizza una opportuna combinazione della procedura di *backfitting* con quello dei minimi quadrati pesati iterati, in uso per i GLM.

In conclusione, un GAM riesce a derivare un trend lungo il periodo temporale, identificando e sommando molteplici funzioni, risultanti in un curva che si adatta ai dati nel migliore dei modi. Inoltre, dato che questo tipo di modellazione si basa, appunto, su funzioni piuttosto che su variabili, essi non sono limitati dalle assunzioni di linearità che richiedono ai predittori ed alla variabile risposta di muoversi lungo una linea retta, che invece si verifica nel caso della regressione lineare (Dominici et al. 2002). Infine, l'utilizzo di questo tipo di modelli, offre la possibilità di isolare e studiare i risultati delle funzioni delle singole variabili indipendenti, così da essere in grado di definire l'effetto di ciascuna di esse sulla variabile di output, aspetto di grande rilevanza ai fini del successo di questa analisi (Simpson 2014).

Per la stima del modello additivo generalizzato vengono utilizzate le seguenti variabili:

- *Parte della giornata*, divisa in mattina, pomeriggio e sera.
- *Giorno della settimana*, suddivisa in sabato, domenica e giorno feriale.
- *Temperatura*, variabile quantitativa della temperatura media giornaliera.
- *Contagi* del giorno precedente.
- *Precipitazioni*, variabile dicotomica della presenza o meno di precipitazioni in giornata.
- *Colore* di zona.
- *Tramonto*, variabile categoriale della fascia oraria del tramonto del sole.
- *Alba*, variabile categoriale della fascia oraria dell'alba.

A differenza dei modelli per serie storiche (es. ARMA, ARIMA,...) nella modellazione tramite modello additivo generalizzato la caratterizzazione della struttura di autocorrelazione tra le osservazioni non è specificata nel modello, che ipotizza quindi che i valori in punti temporali consecutivi siano tra di loro indipendenti. Nel caso in esame, si è inizialmente provato a stimare un GAM senza considerare la dipendenza all'interno dei dati, per poi, se necessario, passare ad una struttura più complessa (Laurinec 2017).

Il modello additivo generalizzato viene stimato, in un primo momento, usando una famiglia di tipo gaussiano e come funzione di legame l'identità, per poi passare alla stima con famiglia "Poisson" e funzione di legame logaritmica. Si confronteranno successivamente i risultati ottenuti dalle due specificazioni, in modo tale da evidenziarne eventuali differenze nella stima della relazione di interesse.

Il modello additivo generalizzato viene stimato in R mediante l'utilizzo della libreria *gam*; oltre alle variabili elencate sopra, è stata integrata la componente non parametrica relativa alla funzione del tempo.

In Figura 3.5 e Figura 3.6 sono presentati i risultati ottenuti dall'applicazione del modello con famiglia Gaussiana e funzione di legame identità. Il test ANOVA (Tabella 3.2) sugli effetti parametrici evidenzia la non significatività,

al netto della presenza delle altre variabili e ad un livello del 5%, delle componenti riguardanti il numero dei contagi del giorno precedente e della variabile categoriale degli orari dell'alba. L'ANOVA sugli effetti non parametrici, invece, restituisce la sola significatività della variabile *temperatura*.

Variabile	**Gradi di libertà**	**p-value**
ANOVA per effetti parametrici		
parte della giornata	2	0 ***
giorno della settimana	2	0,0023
temperatura	1	0
contagi	1	0,27
precipitazioni	1	0,0068
colore	3	0
tramonto	4	0
alba	5	0,11
tempo	1	0
ANOVA per effetti non parametrici		
temperatura	3	0,011
contagi	3	0,15
tempo	3	0,20

Tabella 3.2: Test ANOVA sugli effetti parametrici e non parametrici del modello GAM stimato con famiglia Gaussiana e funzione di legame identità.

I risultati più significativi riguardano le variabili *parte della giornata* e *precipitazioni*. Si nota un forte disallineamento sulla variabile risposta, tra le osservazioni appartenenti alla fascia oraria mattutina e quelle relative a foto scattate nella fascia congiunta di pomeriggio e sera; infatti, in quest'ultimo caso, la densità di persone prevista assume valore superiore. Questo ragionamento può essere esteso alla variabile dicotomica della precipitazione giornaliera, che delinea una minore attività nelle giornate piovose, come poteva essere prevedibile.

I grafici relativi agli effetti non parametrici stimati dal GAM, evidenziano un andamento non monotono per la temperatura, mentre si assiste ad una relazione monotona crescente con la variabile relativa al tempo.

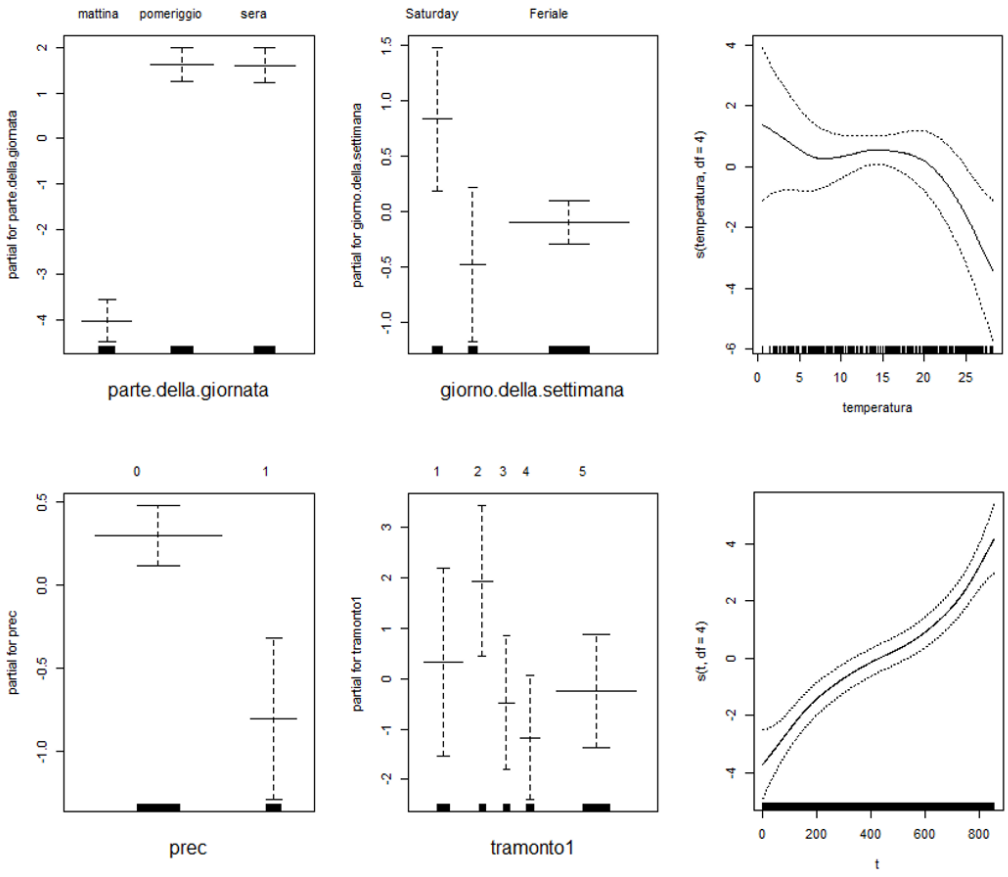

Figura 3.5: Grafici risultanti dall'applicazione del modello GAM sul numero di persone con famiglia "Gaussiana" e funzione di legame "identity", che include anche il colore di zona (si veda Figura 3.6).

Per quanto riguarda, invece, la relazione di interesse tra numero di persone e il colore della zona, il grafico evidenzia principalmente una differenza tra due macrocategorie, quella composta da zona rossa e arancione e quella formata da zona gialla e bianca. Infatti, non sembra esserci una rilevante dissomiglianza tra le componenti interne alle due categorie, diversamente da quanto si poteva ipotizzare dalle analisi preliminari. Inoltre, si evidenzia una minore ampiezza nelle bande di variabilità per la modalità *"zona gialla"*, derivante da una maggior numerosità di osservazioni a disposizione per questa categoria.

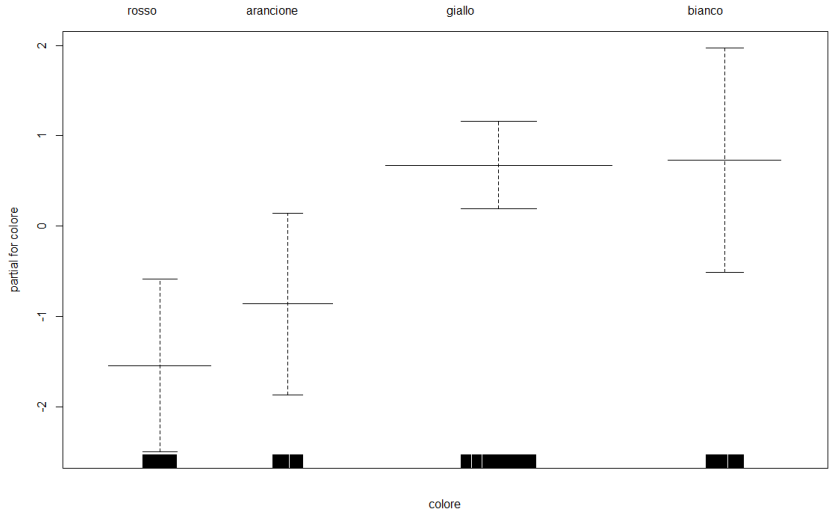

Figura 3.6: Grafico rappresentante la relazione tra numero di persone e colore derivante dall'applicazione del modello GAM con famiglia "Gaussiana" e funzione di legame "identity", che include anche le variabili mostrate in Figura 3.5.

Per verificare il corretto adattamento del modello ai dati è necessario, però, effettuare dei test sulla distribuzione dei residui e assicurarsi, ad esempio, che essi non presentino nessuna struttura di autocorrelazione. Il grafico in Figura 3.7 sembrerebbe indicare l'assenza di una struttura di questo tipo; infatti, sia le autocorrelazioni che le autocorrelazioni parziali rimangono, nella quasi totalità dei valori, all'interno delle bande di confidenza.

In conclusione, dal momento che i residui risultano non autocorrelati, non c'è la necessità di inserire all'interno del modello una componente per tenere

in considerazione un'eventuale struttura di correlazione residuale.

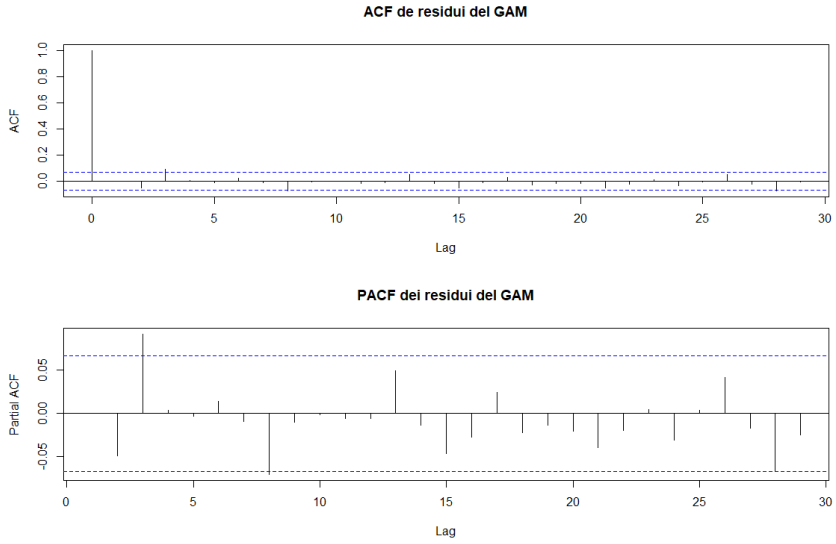

Figura 3.7: ACF e PACF dei residui del modello GAM con famiglia "Gaussiana" e funzione di legame "identity".

Passando al secondo modello, stimato con famiglia "Poisson" e funzione di legame logaritmica, si ottengono dei risultati che non si discostano elevatamente da quelli ottenuti dal modello di cui sopra. In questo caso l'unica componente non significativa, ad un livello del 5%, è quella relativa all'effetto parametrico dell'*"alba"* (Tabella 3.3), in concordanza con quanto ottenuto nella prima analisi.

Anche i grafici in (Figura 3.8 e Figura 3.9), risultanti dalla stima del modello, sono abbastanza in linea con quelli visti in precedenza. L'unica differenze si può riscontrare nel grafico del colore di zona, dove è più netta la distinzione dell'effetto sulla variabile risposta tra le modalità *"zona rossa"* e *"zona arancione"*.

Anche in questo caso, è necessario assicurarsi che i residui non siano autocorrelati al fine di poter considerare la regressione non spuria e quindi ottenere una corretta interpretazione dei risultati. L'assenza di autocorrelazione è visibile dai grafici di ACF e PACF. Anche in questo caso, In conclusione, dato che i residui risultano non autocorrelati, non c'è la necessità di un'ulteriore complicazione nella modellazione.

Variabile	Gradi di libertà	p-value
ANOVA per effetti parametrici		
parte della giornata	2	0
giorno della settimana	2	0,0015
temperatura	1	0
contagi	1	0,0099
precipitazioni	1	0,0013
colore	3	0
tramonto	4	0
alba	5	0,13
tempo	1	0
ANOVA per effetti non parametrici		
temperatura	3	0
contagi	3	0
tempo	3	0

Tabella 3.3: Test ANOVA sugli effetti parametrici e non parametrici del modello GAM stimato con famiglia "Poisson" e funzione di legame logaritmica.

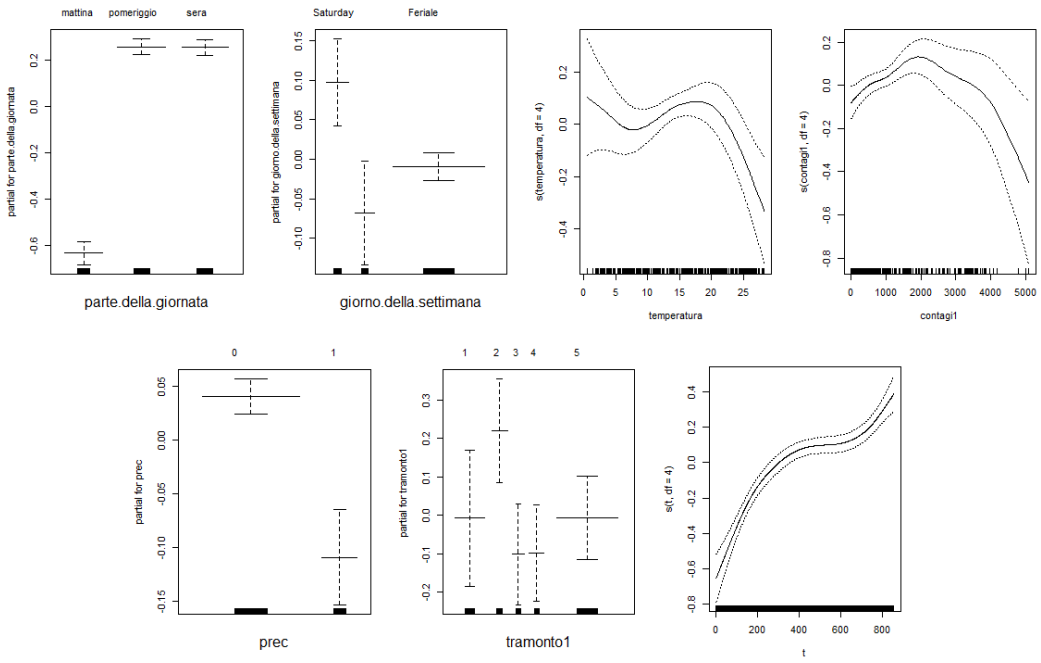

Figura 3.8: Grafici risultanti dall'applicazione del modello GAM sul numero di persone con famiglia "Poisson" e funzione di legame logaritmica.

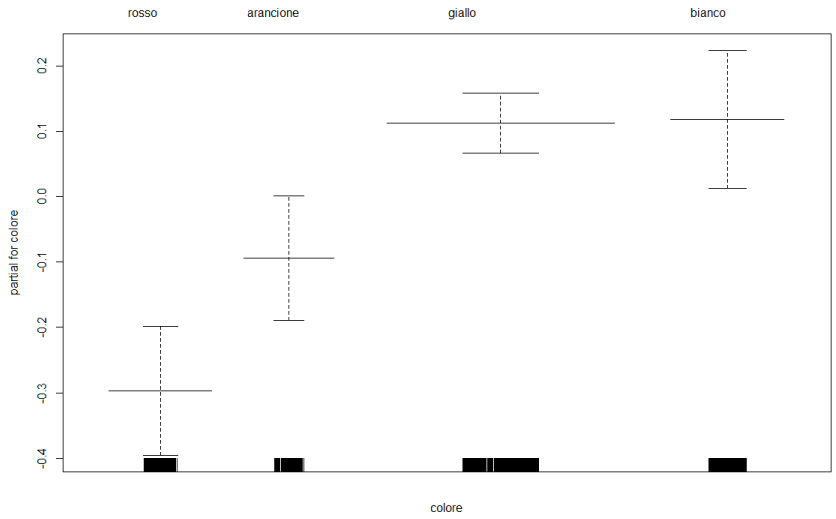

Figura 3.9: Grafico rappresentante la relazione tra numero di persone e colore derivante dall'applicazione del modello GAM con famiglia "Poisson" e funzione di legame logaritmica.

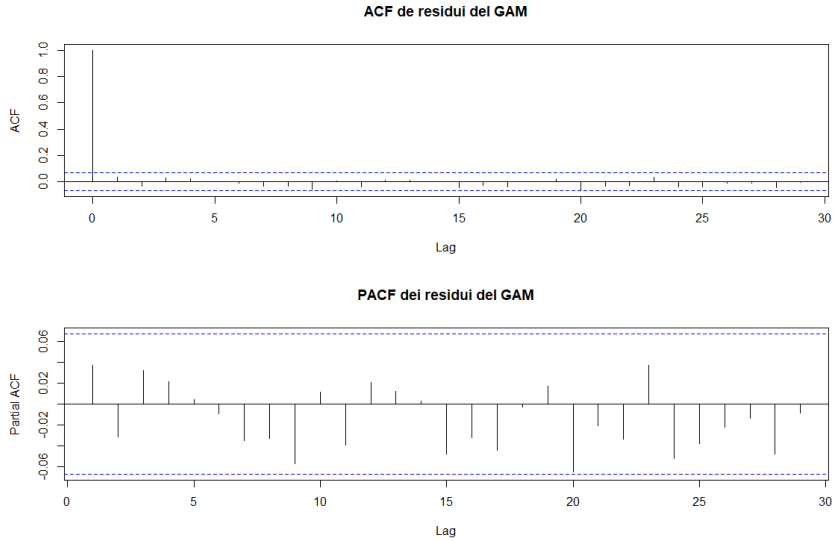

Figura 3.10: ACF e PACF dei residui del modello GAM con famiglia "Poisson" e funzione di legame logaritmica.

In definitiva, il modello con famiglia "Poisson" sembra essere preferibile per due motivi principali:

- Innanzitutto, perché restituisce un valore di AIC minore. Infatti, nella stima con famiglia Gaussiana l'AIC assume valore pari a 5018, mentre con famiglia "Poisson" è uguale a 4857.
- In secondo luogo, per la maggior interpretabilità delle relazioni di dipendenza tra le variabili, in genere caratterizzate da una minor banda di variabilità, soprattutto nella relazione tra numerosità e zona.

Come visto nella prima fase delle analisi preliminari, gli andamenti di variabili, come il giorno della settimana ed il colore di zona, nei confronti del numero di persone subisce importanti variazioni in risposta alla modalità assunta dalla variabile relativa alla fascia oraria considerata. Proprio per sottolineare questa caratteristica si è ritenuto opportuno stimare un GAM su ciascuna delle tre partizioni in cui la variabile *parte della giornata* divide il dataset. Questa metodologia di sviluppo offre la possibilità di studiare i rapporti tra risposta e predittori tenendo in considerazione, a priori, del va-

lore della variabile e quindi dell'interazione di questa con le altre variabili indipendenti.

Le principali differenze, rispetto ai modelli di cui sopra, sono osservabili nelle variabili *giorno della settimana* e *colore* (Figura 3.11).

Per quanto concerne la variabile del giorno della settimana, la difformità si concentra soprattutto a riguardo della modalità del giorno feriale, che, nella fascia mattutina si colloca nella parte superiore del grafico, mentre nella fascia pomeridiana si colloca in quella inferiore.

Riguardo la relazione su cui ci deve soffermare maggiormente, colonna di destra nella Figura 3.11, le differenze si sostanziano sopratutto per le modalità *"mattina"* e *"sera"*. Nel primo caso, si potrebbe ipotizzare che il comportamento delle osservazioni nelle quattro modalità sia abbastanza uniforme rispetto alla variabile di interesse, desunto, inoltre, dalle bande di confidenza sovrapposte una all'altra, per eccezione della coppia (*"zona gialla"*, *"zona rossa"*). Questa caratteristica può trovare spiegazione considerando che la mattina si verificano la maggior parte degli spostamenti casa/lavoro (collegandosi anche a quanto visto per la prima variabile), tra gli unici consentiti in determinate zone a rischio. Nel secondo caso, sono raffigurate sostanzialmente due macrocategorie *"zona rossa"* e (*"zona gialla"*, *"zona arancione"*, *"zona bianca"*).

Per non appesantire troppo la trattazione, si è scelto di mostrare solo i grafici delle variabili che si sono ritenute di maggior interesse nello spiegare il fenomeno. I rimanenti grafici sono riportati in appendice C insieme a quelli relativi all'autocorrelazione dei residui.

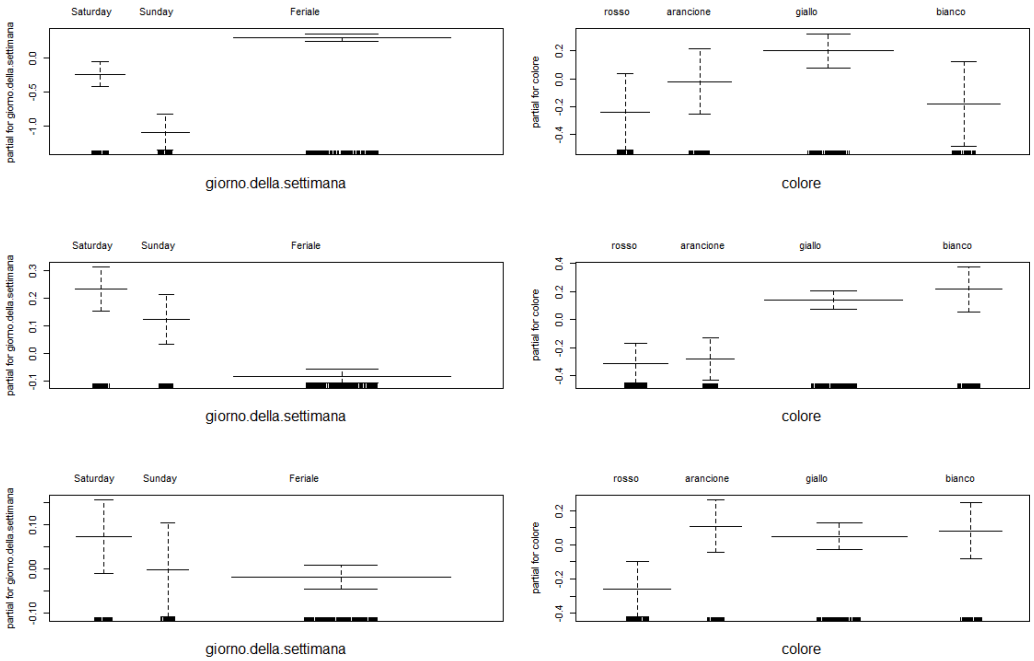

Figura 3.11: Grafici dal modello GAM sulla variabile *numero persone* con dataset su *parte della giornata* = "Mattina" (in alto), *parte della giornata* = "Pomeriggio" (in centro) e *parte della giornata* = "Sera" (in basso).

Capitolo 4

Gruppi di persone: analisi e modelli

Nella seconda fase dell'analisi, l'attenzione è posta sull'identificazione dei gruppi di persone. Anche in questo frangente, l'obiettivo è quello di riuscire a comprendere come si sviluppa, lungo il periodo temporale considerato, la formazione dei gruppi, soffermandosi in particolar modo sulla relazione con il colore di zona.

4.1 Dataset

Il dataset di partenza per il riconoscimento dei gruppi, a differenza di quello utilizzato nel capitolo precedente, si compone di una sequenza di righe per ciascuna foto, per la precisione, un numero di righe pari al numero di persone riconosciute nello scatto. Nello specifico, oltre alle variabili utilizzate per l'analisi del terzo capitolo, si aggiungono:

- xMin, posizione del vertice del *box* inferiore sinistro.
- xMax, posizione del vertice del *box* inferiore destro.
- yMin, posizione del vertice del *box* superiore sinistro.
- yMax, posizione del vertice del *box* superiore destro.

Ogni *box* avrà, quindi, quattro valori, uno per ciascun vertice del rettangolo. Per una più facile comprensione, è riportata in Tabella 4.1 un esempio di

come queste variabili sono valorizzate all'interno del dataset; questi dati fanno riferimento all'immagine riportata in Figura 4.1.

Nome	xMin	xMax	yMin	yMax
IMG_20210415_130128	0,5605	0,5801	0,2985	0,4854
IMG_20210415_130128	0,5395	0,5602	0,2810	0,4783
IMG_20210415_130128	0,6420	0,6573	0,1223	0,2634
IMG_20210415_130128	0,6577	0,6752	0,1212	0,2713

Tabella 4.1: Coordinate dei *box* pre elaborazione per l'immagine in Figura 4.1.

É necessario porre un'attenzione particolare sulla codifica delle coordinate Y dei vertici dei *box*; queste, infatti, sono definiti a partire dalla parte superiore del rettangolo dell'immagine e quindi assumeranno valori più vicini allo zero man mano che le persone si avvicineranno all'estremità alta della foto. Per rendere l'interpretazione di questi valori più intuitiva, come nella solita rappresentazione di un grafico negli assi cartesiani, queste sono state ricodificate nella seguente modalità (a partire dall'informazione sul campo di variazione delle coordinate pari a [0,1]):

$$\begin{aligned} yMin &\leftarrow 1 - yMin \\ yMax &\leftarrow 1 - yMax \end{aligned} \tag{4.1}$$

risultando nella coordinate in Tabella 4.2.

Nome	xMin	xMax	yMin	yMax
IMG_20210415_130128	0,5605	0,5801	0,7015	0,5146
IMG_20210415_130128	0,5395	0,5602	0,7190	0,5217
IMG_20210415_130128	0,6420	0,6573	0,8777	0,7366
IMG_20210415_130128	0,6577	0,6752	0,8788	0,7287

Tabella 4.2: Coordinate dei *box* post elaborazione per l'immagine in Figura 4.1.

In definitiva, le ultime due righe della tabella rimandano alla coppia di persone nella parte superiore dell'immagine, caratterizzate da valori più grandi sulla componente dell'asse Y, mentre le restanti due riguardano gli elementi nella sezione inferiore.

Figura 4.1: Esempio di immagine processata.

4.2 Studio del posizionamento spaziale

Il passo più complicato dell'analisi è quello di riuscire ad identificare quando due o più persone appartengono al medesimo gruppo. I dati che si hanno a disposizione per compiere questo passaggio, sono le coordinate dei vertici del *box* che individua le persone riconosciuto dall'algoritmo *"EfficientDet"*, dal quale vengono restituite in output. Questo passaggio viene affrontato scomponendo il problema in due approfondimenti consecutivi:

- Il primo, riguardante l'esame delle posizioni spaziali delle persone, con lo scopo di tenere in considerazione l'effetto della prospettiva sulla dimensione dei *box* e sulla distanza tra essi.

- Il secondo, relativo allo studio delle distanze interpersonali al fine di identificare i gruppi di persone.

Successivamente, le posizioni spaziali delle persone saranno rappresentate nello spazio euclideo medianti l'utilizzo di punti, le quali coordinate saranno pari a:

$$\begin{aligned} x &\leftarrow \frac{(xMin + xMax)}{2} \\ y &\leftarrow yMin \end{aligned} \tag{4.2}$$

4.2.1 La prospettiva

In questa analisi la prospettiva gioca un ruolo centrale nell'interpretazione delle distanze interpersonali e di conseguenza necessita della giusta considerazione.

L'idea di partenza è quella di riuscire a definire, a partire da elementi chiave all'interno delle foto, quando due coppie di persone, posizionate in punti differenti dell'immagine, presentano la medesima distanza, nella realtà, tra gli individui che la compongono. Per fare maggiore chiarezza ipotizziamo che ci siano due coppie, (A,B) e (C,D), chiamiamo "dR" la distanza reale e "dE" la distanza euclidea tra i punti nella foto, così come specificati precedentemente. Definiamo, quindi, dR(A,B) = dR(C,D) e, partendo da questa uguaglianza e dai valori calcolati di dE(A,B) e dE(C,D), l'obiettivo è quello di riuscire a ricavare una costante di proporzionalità tale da poter ricondurre tutte le distanza interpersonali su una stessa base valida, per poi poter effettuare il confronto.

Per compiere questo procedimento, si è inizialmente individuata un'immagine tale per cui le condizioni di cui sopra fossero verificate, che è la foto in Figura 4.1. La relazione di uguaglianza è confermata grazie all'informazione ottenuta a partire dalla presenza di mattonelle di uguale lunghezza e larghezza lungo la parte centrale del campo. Grazie a ciò si può concludere che le persone componenti le due coppie assumono la stessa distanza l'una dall'altra. Per comodità nell'esposizione, denominiamo la coppia nella parte inferiore, (A,B), e la coppia nell'estremità superiore (C,D).

Il primo passaggio consiste nel misurare la distanza euclidea tra A e B, dE(A,B), e tra C e D, dE(C,D), per poi calcolare la differenze tra le due:

$$dE(A, B) - dE(C, D) \tag{4.3}$$

Il secondo passo corrisponde alla misura della distanza tra le ordinate delle due coppie. Dato che, ciascuna coppia è formata da due punti e quindi due ordinate, si è deciso di utilizzare come ordinata di riferimento la media tra le due. La distanza viene definita, quindi, come:

$$\frac{C(y) - D(y)}{2} - \frac{A(y) + B(y)}{2} \tag{4.4}$$

dove, con *(y) si fa riferimento alla ordinata Y del punto selezionato.

Infine, avendo ricavato il valore relativo alla differenza delle distanze tra gli elementi della coppia e alla distanza tra le altezze medie delle coppie, è possibile ottenere la costante di proporzionalità semplicemente effettuando il rapporto tra queste due componenti:

$$costante\_prop \leftarrow \frac{dE(A,B) - dE(C,D)}{\frac{C(y)-D(y)}{2} \_ \frac{A(y)+B(y)}{2}} \tag{4.5}$$

In questo modo, per riportare tutte le distanze tra punti del dataset sulla stessa base di riferimento, è sufficiente sommare alla distanza di partenza il prodotto tra l'altezza media della coppia e la costante così calcolata, per esempio:

$$dNuova(A,B) \leftarrow dE(A,B) + \left(\frac{A(y)+B(y)}{2} \cdot costante\_prop\right) \tag{4.6}$$

Nella tabella seguente sono riassunti i vari passaggi appena descritti utilizzando i dati reali basati sulla foto in Figura 4.1.

Descrizione	**Formula**	**Risultato**
Distanza tra A e B	$dE(A,B)$	0,02697
Distanza tra C e D	$dE(C,D)$	0,01678
Differenza tra le distanze interpersonali	$dE(A,B) - dE(C,D)$	0,01019
Ordinata media della coppia (A,B) ($h_{A,B}$)	$(A(y) + B(y))/2$	0.7103
Ordinata media della coppia (C,D) ($h_{C,D}$)	$(C(y) + D(y))/2$	0.8782
Differenza tra le ordinate medie delle coppie	$h_{C,D} - h_{A,B}$	0.1680
Costante di proporzionalità	$\frac{dE(A,B) - dE(C,D)}{h_{C,D} - h_{A,B}}$	**0.06065**

Tabella 4.3: Passaggi necessari al calcolo della costante di proporzionalità utilizzando i dati reali.

In conclusione, è possibile utilizzare la costante di proporzionalità per derivare la distanza reale, così come definita nella formula 4.6:

$$\begin{aligned} dNuova(A,B) &\leftarrow 0,02697 + (0.7103 \cdot 0.06065) = 0.07005 \\ dNuova(C,D) &\leftarrow 0,01678 + (0.8782 \cdot 0.06065) = 0.07005 \end{aligned} \tag{4.7}$$

Applicando questa soluzione a ciascuna distanza di interessa è possibile ottenere una serie di valori tra loro confrontabili, depurata della componente prospettica. Facendo riferimento alla foto in Figura 4.1, in Figura 4.2 sono riportati i grafici delle posizioni spaziali delle persone, prima e dopo l'applicazione della trasformazione prospettica. Quello che accade è esattamente quanto visualizzato nella formula 4.7; le distanze, in questo caso, vengono elaborate in modo tale da risultare depurate della componente di prospettiva data dalla fotografia, ed assumono entrambe il valore che avrebbero se gli elementi dell'immagine fossero posizionati esattamente sull'origine dell'asse delle ordinate, in modo tale da essere confrontabili.

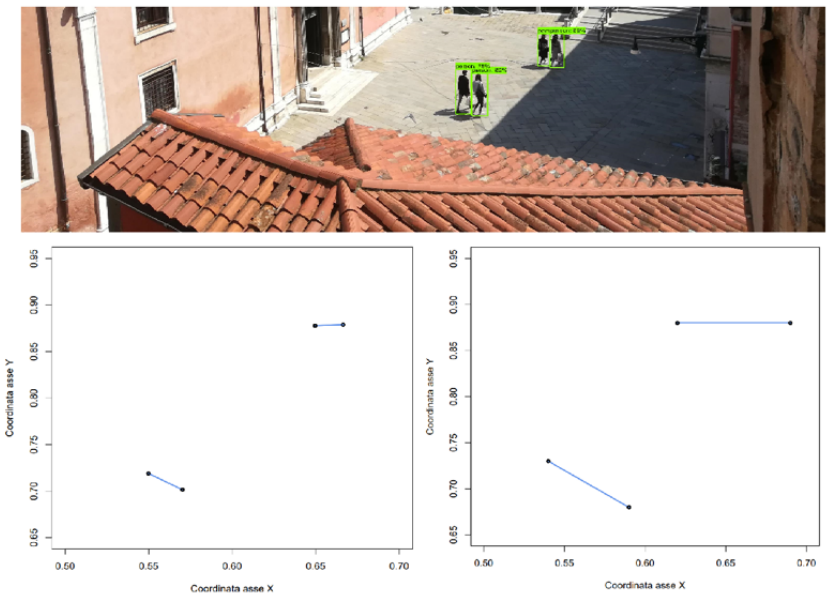

Figura 4.2: Coordinate delle persone in Figura 4.1 prima (sinistra) e dopo (destra) l'applicazione della trasformazione prospettica.

Infine, è necessario aggiungere che, nell'analisi del presente caso di studio, non è stata inserita nel calcolo della costante di proporzionalità, il coefficiente relativo al diverso posizionamento delle coppie lungo l'asse delle ascisse, avendo un'influenza praticamente nulla ai fini dell'identificazione dei gruppi.

4.3 Identificazione dei gruppi

Il passaggio successivo è quello relativo all'elaborazione di una metodologia di aggregazione tra le persone nelle foto. Per raggiungere questo scopo, sono stati eseguiti e poi confrontati, in termini di applicabilità e risultati, tre tipologie di tecniche per il raggruppamento:

- Algoritmo di clustering *"k-means"*, metodo non gerarchico.
- Metodo di raggruppamento gerarchico.
- Analisi di una rete.

4.3.1 *K-means*

L'algoritmo di raggruppamento *k-means* è una tecnica non supervisionata di tipo non gerarchico. *K-means* si basa sulla definizione dei centroidi, ossia punti appartenenti allo spazio che mediano le distanze tra tutti i dati appartenenti al cluster ad esso associato (una sorta di baricentro del cluster).

Il procedimento si sostanzia nei seguenti passaggi:

1. La prima cosa da fare è decidere il numero di classi, pari a K.
2. Si scelgono poi, in modo casuale, K centroidi, non coincidenti tra loro.
3. Si calcola la distanza di ogni punto del dataset rispetto ad ogni centroide.
4. Ogni punto del dataset viene associato al cluster collegato al centroide più vicino.
5. Successivamente, viene ricalcolata la posizione dei centroidi facendo la media delle coordinate di tutti i punti appartenenti al cluster associato.
6. Si itera dal punto 3 in poi, fino a che non ci sarà più nessun cambiamento riguardo la composizione dei gruppi.

Per il caso in esame, ovviamente, non si ha la conoscenza a priori relativa al numero di suddivisioni che avranno le popolazioni delle fotografie. Proprio per questo motivo, anche grazie alla bassa numerosità di osservazioni di cui si dispone in ogni immagine, si decide di iterare l'algoritmo di *k-means* per ciascun valore di K possibile e scegliere, alla conclusione di esso, il K ottimale, tale per cui la somma dei quadrati *entro* i cluster assume valore minimo.

Nel caso pratico, considerando la foto processata in Figura 4.3, viene applicato il metodo delle *k-means*, ottenendo come risultato la suddivisione in gruppi come rappresentata in Figura 4.4.

Figura 4.3: Immagine di riferimento per il confronto tra le metodologie di raggruppamento.

In questo esempio, l'aggregazione che garantisce il valore minimo della statistica di riferimento, corrisponde alla partizione delle osservazioni in sette gruppi, di cui cinque composti da una persona singola e due da una coppia. È possibile notare che l'algoritmo non coglie alla perfezione la reale aggregazione delle persone nel campo, infatti, non inserisce all'interno dello stesso gruppo le due figure posizionate nella parte in alto a destra della foto, mentre aggrega i due elementi nella parte centrale, erroneamente.

Oltre a ciò, questo metodo non offre la possibilità di tenere in considerazione l'effetto dato dalla prospettiva sulle distanze interpersonali.

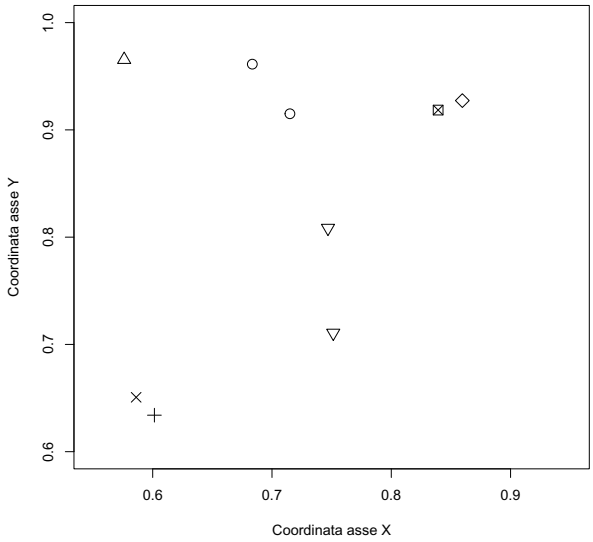

Figura 4.4: Suddivisione in gruppi con algoritmo *k-means*.

4.3.2 Metodo gerarchico

I metodi di clustering gerarchici, rispetto ai precedenti, non fissano a priori un numero di classi predefinito ma si basano su due possibili strutture operative:

- Algoritmo **divisivo**, che individua i gruppi suddividendo il gruppo costituito da tutte le unità del dataset.
- Algoritmo **agglomerativo**, che invece forma i gruppi a partire dall'aggregazione di gruppi composti da una unità ciascuno.

Per lo svolgimento di questa analisi si farà riferimento all'algoritmo agglomerativo. Il metodo gerarchico si articola nella struttura che segue:

1. Si inizia il procedimento calcolando le distanza tra ciascuna coppia di osservazioni, da memorizzare all'interno di una matrice delle distanze.
2. Ogni punto viene collocato nel suo gruppo formato da un unico elemento (algoritmo agglomerativo).

3. Le unità che presentano una distanza tra punti minore vengono fuse all'interno del medesimo gruppo.

4. Successivamente, viene ricalcolata la distanza tra il nuovo cluster ed i vecchi, collocandola in una nuova matrice di distanze, aggiornata.

5. Infine, gli ultimi due passaggi vengono ripetuti fino a che tutte le unità non saranno aggregate in un unico gruppo.

I procedimenti degli algoritmi gerarchici si differenziano unicamente per il diverso criterio che regola la valutazione delle distanze tra i gruppi, ai fini delle aggregazioni in serie, e per la scelta del calcolo delle distanze per la composizione della matrice delle distanze iniziale. In questo caso viene applicato il metodo del legame medio, tale per cui la distanza tra i gruppi è posta pari al valore medio aritmetico di tutte le distanze tra gli elementi. Come misura della distanza viene scelta la distanza euclidea.

La parte più complicata di questa tecnica è quella di riuscire a rendere la sequenza di operazioni elencate automatizzata, senza avere la necessità di scegliere il numero ottimale di partizioni per ognuna delle foto del dataset. Per renderlo possibile si è fatto ricorso all'indice di Dunn. L'indice di Dunn è il rapporto tra la distanza minima tra le osservazioni appartenenti a gruppi differenti e la distanza massima *entro* i cluster. Un indice di Dunn, compreso nell'intervallo $[0, +\infty)$, più elevato implica un clustering migliore. Si presume che un migliore raggruppamento significhi che i cluster siano compatti e ben separati dagli altri cluster.

Il metodo viene implementato sui dati mediante l'utilizzo della libreria R *hclust*. Questa funzione prende in input la matrice delle distanze e restituisce in output un oggetto che descrive l'albero prodotto dal processo di clustering, utilizzato poi per testare l'adattamento di ciascuna partizione elaborata attraverso l'indice di Dunn. Viene scelta la suddivisione che garantisce il valore massimo della statistica.

Il modello così stimato ottiene un risultato piuttosto soddisfacente, riuscendo a cogliere la corretta suddivisione in gruppi per lo scatto in Figura 4.3, a differenza dell'algoritmo *k-means*. È necessario sottolineare, che, come evidenziato precedentemente, in questi approcci di modellazione non è possibile

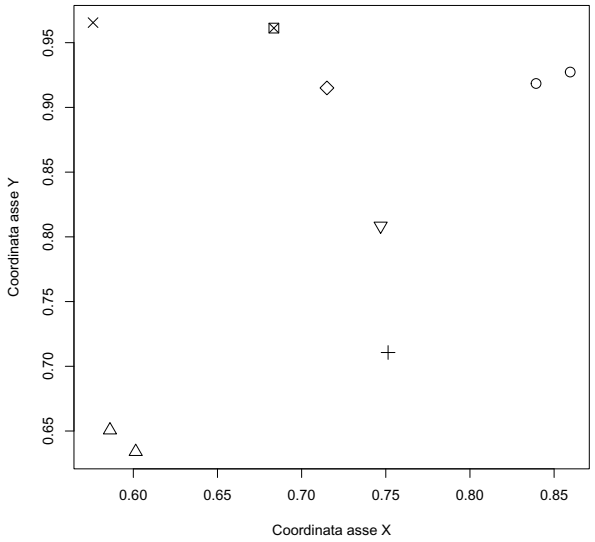

Figura 4.5: Suddivisione in gruppi con algoritmo gerarchico.

tenere conto della componente prospettica, la quale potrebbe causare una distorsione nei risultati e una conseguente errata interpretazione.

4.4 Analisi rete

Sulla base delle problematiche ed i limiti dei metodi precedenti, si è pensato di introdurre una modalità di analisi diversa, basata sulla rappresentazione delle persone come nodi all'interno di una struttura di rete. La differenza con la composizione tipica di una rete è che in questo caso l'unica informazione che si ha a disposizione è quella relativa alle caratteristiche dei nodi (in termini di posizionamento nello spazio euclideo), senza avere alcuna indicazione sulla presenza di archi tra questi. L'obiettivo è riuscire ad utilizzare questa informazione per, in un primo momento, elaborare una strategia per l'identificazione degli archi e poi, utilizzando le due componenti congiunte, costruire una metodologia per il raggruppamento delle unità. La prospettiva del risultato finale corrisponde ad una rete in cui due o più nodi sono colle-

gati da archi, nel solo caso in cui questi appartengano al medesimo gruppo di persone.

4.4.1 Misura delle distanze iniziali

La prima operazione da compiere è il calcolo delle distanze euclidee tra le unità del dataset. Per elevate numerosità delle osservazioni, questa operazione potrebbe diventare computazionalmente onerosa, e, in aggiunta a ciò, per determinate coppie di punti, la misura della distanza non è necessaria, ad esempio, per persone posizionate agli estremi opposti del campo e che quindi, logicamente, non possono far parte dello stesso gruppo. Proprio per questi motivi, si sceglie di filtrare le coppie sulle quali concentrarsi, definendo il *vicino* più *vicino* (libreria R *knearneigh*) per ciascuna persona nella foto analizzata (Figura 4.6), in modo da calcolare le sole distanze di interesse (si ricorda che si fa sempre riferimento alla foto in Figura 4.3). In questo modo il numero di confronti da effettuare si riduce sensibilmente, con conseguente costo computazionale richiesto notevolmente minore.

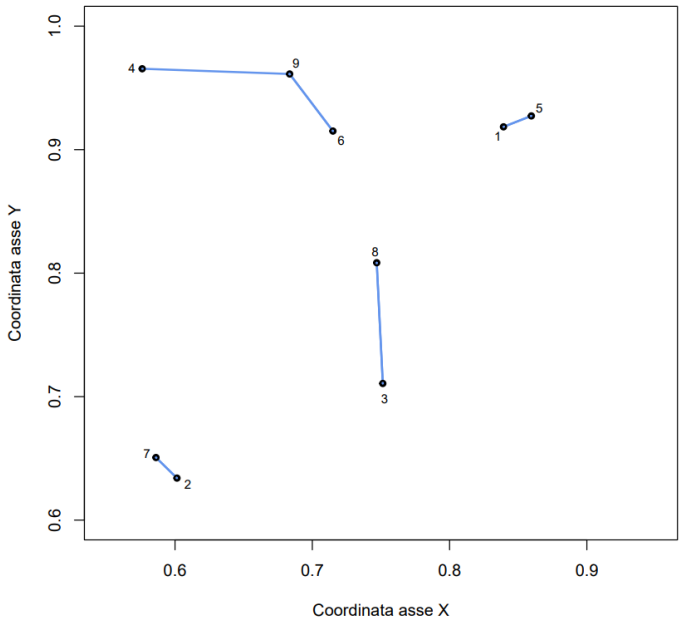

Figura 4.6: Risultato dell'algoritmo del *vicino* più *vicino*.

4.4.2 Elaborazione delle distanze iniziali

Dopo aver misurato le distanze interpersonali, il passaggio successivo si concretizza nel rielaborare questi valori, mediante l'utilizzo della formula 4.6, al fine di riportare tutte le misure su un'unica base confrontabile, depurate della componente prospettica. Per tutte le coppie precedentemente individuate dall'algoritmo del *vicino* più *vicino*, viene quindi calcolata la distanza euclidea e l'ordinata media tra gli elementi di ciascuna di queste. Si compone, in questo modo, un *dataframe* dove, nelle prime due colonne sono riportati gli identificativi delle coppie, nelle successive due le informazioni relative a distanza ed ordinata media ed, infine, nell'ultima colonna viene inserita la nuova distanza misurata per tenere conto della prospettiva dell'immagine (esempio in Tabella 4.4).

Identificativo 1	**Identificativo 2**	**Ordinata**	**Dist. iniziale**	**Dist. elaborata**
1	5	0,93	0,022	0,078
2	7	0,64	0,023	0,062
3	8	0,76	0,098	0,14
4	9	0,96	0,11	0,17
6	9	0,94	0,056	0,11

Tabella 4.4: *Dataframe* costruito per gestione delle distanze elaborate, entro le coppie (riferimento al grafico in Figura 4.6).

4.4.3 Raggruppamento delle unità

A questo punto è possibile definire quali punti siano abbastanza vicini per poter essere definiti appartenenti al medesimo gruppo. Per fare questo si confrontano le nuove distanze elaborate con una soglia minima di *benchmark*, ottenuta da un'analisi qualitativa, al di sotto della quale la relazione viene definita "significativa" ai fine del raggruppamento.

In questo modo si arriva alla definizione di una vera e propria rete, dove i nodi sono le varie persone presenti nella fotografia e gli archi sono rappresentati dalle relazioni tra punti la cui distanza è risultata significativa nel confronto appena effettuato. Il vantaggio di avere a disposizione una strut-

tura rete è la possibilità di applicare modelli per il raggruppamento come il metodo di Louvain.

Il metodo di Louvain (Giordano 2018) per il rilevamento dei gruppi è un algoritmo euristico utilizzato nelle reti, che si basa sulla massimizzazione della modularità. La modularità è una quantità che indica la qualità dell'assegnazione dei nodi a determinate comunità, valutando quanto più densamente sono connessi i nodi all'interno di una comunità rispetto a quanto sarebbero connessi, in media, in una rete casuale opportunamente definita. Il metodo consiste nell'applicazione ripetuta di due passaggi consecutivi:

1. Il primo passo consiste nell'assegnazione dei nodi alle comunità, favorendo la massimizzazione locale della modularità.

2. Il secondo passo riguarda la definizione di una nuova rete, in cui i nodi sono costituiti dalle comunità individuate nella prima fase. Gli archi presenti tra i nodi della nuova rete sono dati dai collegamenti tra i nodi assegnati alle comunità.

Queste due operazioni sono ripetute fino a che non saranno più possibili ulteriori riassegnazioni di comunità che aumenteranno il valore della modularità.

In tutti i casi l'applicazione del metodo di Louvain lascia invariata la suddivisione in gruppi ottenuta dalla modellazione antecedente; questo è dovuto alla caratteristica dei nodi della rete di avere al massimo una connessione. Ad ogni modo, il metodo è stato presentato perché può essere di spunto per eventuali sviluppi futuri, nei quali si decidesse di optare per un numero diverso di *vicini* più *vicini*.

Logicamente, per le immagini in cui il numero di persone è pari o minore di uno, non è necessario applicare quanto detto fino a questo punto. A priori, il procedimento identifica le fotografie di questo genere e assegna valore nullo al numero di gruppi.

4.5 Analisi

Il procedimento analizzato nel precedente paragrafo viene ripetuto per tutte le immagini del dataset, restituendo in ciascuna iterazione il numero

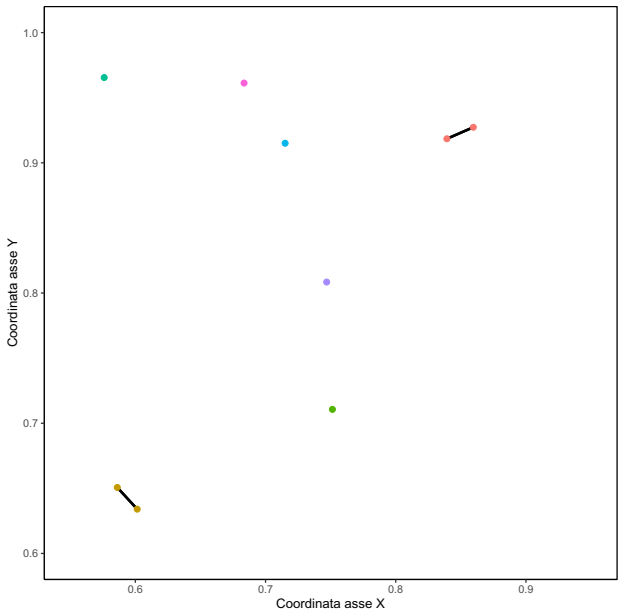

Figura 4.7: Risultato del raggruppamento.

totale di gruppi identificati (per gruppo si intende ciascuna aggregazione di due o più persone). Si ottiene in questo modo un nuovo vettore, di lunghezza pari al numero di fotografie analizzate, contenente in ciascuna cella il numero di gruppi per ognuna di esse (per un maggiore approfondimento delle logiche utilizzate nell'implementazione si veda l'appendice D).

Una prima cosa interessante da poter fare, è studiare la relazione tra questi valori ed il colore di zona. Logicamente, il numero di gruppi sarà strettamente correlato con il numero di persone nell'immagine; si suppone che, in immagini con una maggiore densità di persone sia più probabile che alcune di queste vadano a formare un gruppo, e proprio per questo motivo è utile calcolare il rapporto tra queste due quantità in modo da ottenere una vista sul numero medio di gruppi per persona in ciascun colore di zona (Figura 4.8). Per coerenza nei risultati, non si è tenuto conto di quelle immagini con un numero di persone minore o uguale a uno, dato che queste non ammettevano, a priori, la presenza di gruppi.

Dal grafico si evince un comportamento pressoché simile tra le quattro classi in ascissa, in termini di mediana (linea nera), mentre i valori delle

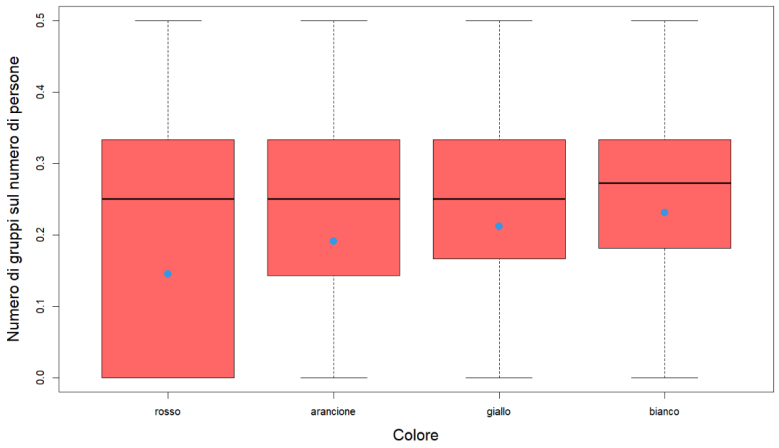

Figura 4.8: Diagramma a scatola e baffi del numero di gruppi sul totale di persone in foto per colore di zona (il pallino blu indica la media).

media (pallino azzurro) riprendono quanto visualizzato nella Figura 3.1 del capitolo 3, con, in questo caso, una accentuazione più ridotta delle differenze tra le quattro modalità. È, comunque, mantenuto l'andamento crescente della variabile di interesse al passare da una zona di pericolo elevata ad una zona a basso rischio.

Inoltre, può essere utile analizzare come varia il numero di persone facente parte di un gruppo, in relazione al numero totale di persone (Figura 4.9). In questo modo, si può evincere la percentuale di queste che, sulla base del colore di zona, si trova a percorrere la strada insieme ad altre persone ed evidenziare le differenze tra le quattro modalità. Come si può vedere nell'immagine, il range di variazione è il medesimo per le quattro modalità della variabile del colore di zona e, come si vedeva anche per lo scorso grafico, la media ha un andamento crescente al passare dalla zona rossa a quella bianca.

Per quanto riguarda lo studio della relazione di interesse tra numero di gruppi e colore di zona, si sceglie di seguire il *fil rouge* della modellazione dello scorso capitolo, mediante l'utilizzo di un modello Gam. Come visto precedentemente, l'obiettivo è quello di analizzare la serie storica dei valori riuscendo a valorizzare la componente di dipendenza tra le due variabili coinvolte.

Per la stima del modello additivo generalizzato vengono incluse le medesime variabili utilizzate nella prima analisi: *giorno della settimana*, *tem-*

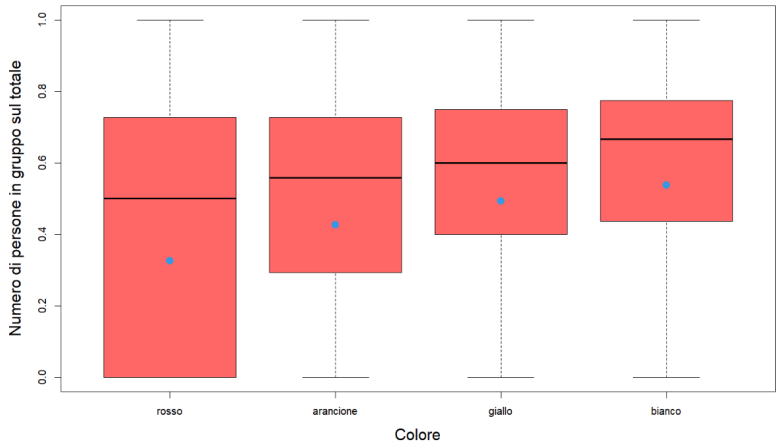

Figura 4.9: Diagramma a scatola e baffi del numero di persone appartenenti ad un gruppo sul totale di persone in foto per colore di zona (il pallino blu indica la media).

peratura, *contagi*, *precipitazioni*, *colore di zona*, *tramonto*, *alba* e la variabile relativa al tempo. Anche in questo caso la modellazione viene svolta su tre sezioni complementari del dataset, una per la mattina, una per il pomeriggio e una per la sera e viene scelta la famiglia Poisson con funzione di legame logaritmica in linea con le scelte fatte nel precedente capitolo.

I risultati dei modelli, in questo caso, sono particolarmente discordanti l'uno dall'altro, sia in termini di significatività delle variabili coinvolte, sia per quanto concerne i risultati ottenuti riguardo gli effetti sulla variabile risposta dovuta alla variazione del colore di zona.

Il modello stimato sulla fascia mattutina restituisce la significatività del solo parametro legato alla variabile del giorno della settimana, indicando quindi l'assenza di un effetto significativo della variabile *colore*, diversamente da quanto ottenuto per il numero di persone.

Per gli altri due modelli, se ci si concentra sulla relazione di interesse, (Figura 4.9), senza appesantire ulteriormente la trattazione, è possibile notare, per le osservazioni serali, un comportamento pressoché omogeneo del numero di gruppi tra le quattro modalità della variabile del colore mentre per quelle con parte della giornata pari a pomeriggio ci si ritrova con un grafico similare a quello in Figura 3.11, dove le differenze si sostanziano essenzialmente tra

i due macrogruppi (*"zona rossa"*, *"zona arancione"*) e (*"zona gialla"*, *"zona bianca"*)

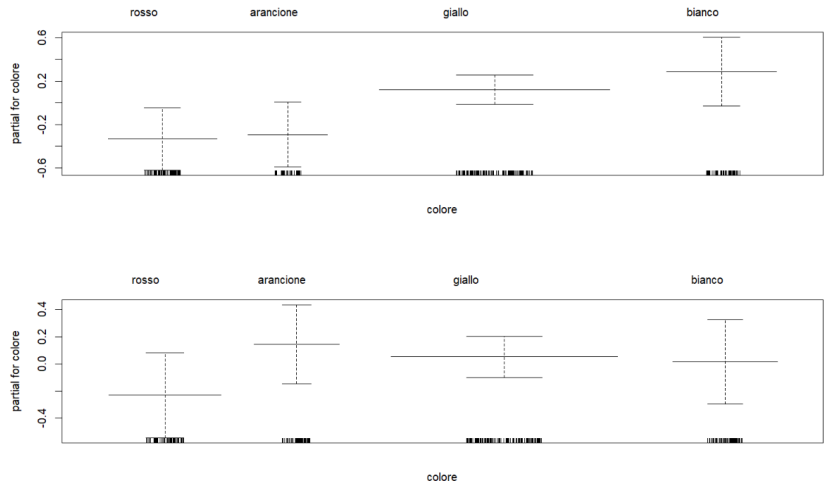

Figura 4.10: Grafici dal modello GAM con variabile riposta *numero gruppi* sulla variabile *colore zona* con dataset su *parte della giornata* = "Pomeriggio" (in alto) e *parte della giornata* = "Sera" (in basso).

Per tutti e tre i modelli l'analisi dei residui dimostra che non è presente una struttura di autocorrelazione e quindi non è necessario complicare ulteriormente la modellazione proposta. Le tabelle per la significatività dei parametri e i grafici delle autocorrelazioni e autocorrelazioni parziali sono riportati all'interno della sezione dell'appendice C.2.

Capitolo 5

Conclusioni

L'elaborato si è posto l'obiettivo di studiare se e che tipo di relazione fosse presente tra le variabili di interesse, numero di persone e numero di gruppi, ed il colore della zona di rischio del Veneto, lungo un sottoinsieme del periodo di convivenza con il Coronavirus. A tal proposito, in una prima fase del lavoro, sono state elaborate, mediante tecniche per il rilevamento di oggetti, una grande quantità di fotografie scattate in più giorni e fasce orarie diverse, al fine di estrarre informazioni utili per la codifica delle variabili risposta.

Alla luce di quanto appena descritto, l'analisi statistica per lo studio della correlazione tra le variabili ha avuto luogo in due sezioni distinte. In una prima fase si è studiato l'andamento della serie storiche del numero di persone, in relazione al colore di zona e ad una serie di variabili esplicative utili per la definizione dei diversi scenari e condizioni di contorno. Dalle analisi esplorative è emerse una forte eterogeneità tra i quattro colori e tra le categorie della variabile riguardante la fascia oraria dello scatto, sopratutto tra la mattina ed il resto della giornata. Proprio per questo motivo, si è ritenuto opportuno, in fase di stima, adattare i modelli separatamente per ciascuna fascia oraria. Dato che lo scopo dell'elaborato è quello di valorizzare la relazione tra le variabili, offrendo un'interpretazione di questa, si è scelto di procedere con un modello che garantisse una modellazione di questo tipo. Dalla stima dei modelli additivi è stato, infatti, possibile valutare l'impatto di ciascun predittore sulla variabile indipendente. In particolare, si evince che il colore di zona ha un comportamento pressoché omogeneo tra le quattro modalità nella fascia

mattutina, mentre presenta differenze più significative nelle altre due fasce rimanenti, nelle quali è possibile rilevare come l'aggravarsi della situazione di rischio comporti un decremento nel numero di persone presenti nel campo. Infine, è stata integrata una rappresentazione dei grafici di autocorrelazione e autocorrelazione parziale dei residui del modello, al fine di confermare il suo corretto adattamento ai dati.

La seconda fase di analisi è suddivisa in due *step* consecutivi: l'iniziale studio per il riconoscimento dei gruppi e la successiva modellazione dei risultati ottenuti. Dopo aver sviluppato una regola per la considerazione dell'effetto attribuito alla componente prospettica, ciascuna immagine è stata trattata come una rete di individui in modo da poter derivare le aggregazioni in base alle distanze interpersonali ricalcolate e confrontate. In questo caso i nodi delle reti erano rappresentati dalle coordinate spaziali da ciascuna persone riconosciuta nelle foto, mentre la presenza degli archi veniva elaborata in un secondo momento, mediante il confronto delle nuove distanze calcolate con una soglia prefissata. Per la fase di stima si è ritenuto opportuno riprendere la stessa modellazione proposta per la prima parte di analisi, e quindi procedere stimando un GAM sulle tre porzioni di dataset elaborate come esposta poco prima. Riguardo la relazione tra numero di gruppi e colore di zona, nel modello stimato sulle righe con *parte della giornata* pari a "Mattina" non si è ottenuto un effetto significativo di quest'ultima sulla variabile risposta, mentre la fascia serale si evince un comportamento pressoché omogeneo del numero di gruppi tra le quattro modalità del colore. Infine, per il pomeriggio la rappresentazione della relazione di interesse presenta una distribuzione analoga a quanto visto per l'analisi del numero di persone, infatti si individuano due macrogruppi con caratteristiche differenti: il primo formato da (*"zona rossa"*, *"zona arancione"*) ed il secondo da (*"zona gialla"*, *"zona bianca"*). Per la precisione, nel secondo insieme sembrerebbe essere più probabile l'instaurazione di gruppi tra le persone nel Campo veneziano.

In conclusione, per rispondere alla domanda di ricerca iniziale, si può dire che gli abitanti di Venezia abbiano rispettato le misure restrittive instaurate lungo il periodo di analisi, soprattutto nei casi in cui queste fossero diventate più pesanti (e.g. zona rossa). Infatti, il numero di persone ed il numero di gruppi è sempre superiore nei casi di zona gialla e zona bianca, senza molte

differenze tra le due, soprattutto nella fascia pomeridiana. Il punto di attenzione è riguardo il numero di persone e di gruppi riscontrati nella fascia mattutina, che, diversamente dalle altre, evidenzia un effetto non determinante del diverso colore di zona. Questo può essere dovuto al fatto che la maggior parte degli spostamenti in questa fascia siano dovuti a motivi lavorativi oppure di necessità, e che quindi potevano garantire di un maggior livello di tolleranza.

Possibili sviluppi futuri possono riguardare la gestione più accurata delle immagini caratterizzate da una visibilità molto scarsa, che in alcuni casi hanno causato delle imperfezioni nei risultati dell'algoritmo di riconoscimento di oggetti. Inoltre, potrebbe essere interessante integrare la serie attuale del numero di persone e del numero di gruppi con i nuovi dati relativi agli ultimi mesi e verificare se si è assistito ad un cambiamento nei comportamenti con il prolungarsi dello stato di emergenza.

Bibliografia

Azzalini, Adelchi e Bruno Scarpa (2009). *Analisi dei dati e data mining*. Springer Science & Business Media.

Dominici, Francesca et al. (2002). «On the use of generalized additive models in time-series studies of air pollution and health». In: *American journal of epidemiology* 156.3, pp. 193–203.

Ghosh, Anirudha et al. (2020). «Fundamental concepts of convolutional neural network». In: *Recent Trends and Advances in Artificial Intelligence and Internet of Things*. Springer, pp. 519–567.

Giordano, Monica (2018). «Analisi di comunità in una rete bipartita pazienti-diagnosi di ricoveri ospedalieri in Regione Lombardia». In.

Gupta, Dishashree (2017). «Architecture of convolutional neural networks (cnns) demystified». In: *Analytics Vidhya*.

Karn, Ujjwal (2016). «An intuitive explanation of convolutional neural networks». In: *The data science blog*.

Laurinec, Peter (2017). «Doing magic and analyzing seasonal time series with GAM (Generalized Additive Model) in R». In: *Time series data mining in R.[Google Scholar]*.

Saha, Sumit (2018). «A comprehensive guide to convolutional neural networks—the ELI5 way». In: *Towards data science* 15.

Simpson, Gavin (2014). «Modelling seasonal data with GAMs». In: *From the Bottom of the Heap*.

Tan, Mingxing e Quoc Le (2019). «Efficientnet: Rethinking model scaling for convolutional neural networks». In: *International conference on machine learning*. PMLR, pp. 6105–6114.

Tan, Mingxing, Ruoming Pang e Quoc V Le (2020). «Efficientdet: Scalable and efficient object detection». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790.

Tsang, Sik-Ho (2019). «Review: SSD—single shot detector (object detection)». In: *medium. com. https://towardsdatascience. com/review-ssd-single-shotdetector-object-detection-851a94607d11 (accessed June 25, 2021)*.

Wu, Jianxin (2017). «Introduction to convolutional neural networks». In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23, p. 495.

Appendice A

Codice Python utilizzato per la fase di *pre-processing*

Codice A.1: Caricamento modelli e creazione metodo per convertire le immagini in array.

```
def load_image_into_numpy_array(path):
        image = None
        if(path.startswith('http')):
        response = urlopen(path)
        image_data = response.read()
        image_data = BytesIO(image_data)
        image = Image.open(image_data)
        else:
        image_data = tf.io.gfile.GFile(path, 'rb').read()
        image = Image.open(BytesIO(image_data))

        (im_width, im_height) = image.size
        return np.array(image.getdata()).reshape(
        (1, im_height, im_width, 3)).astype(np.uint8)

ALL_MODELS = {
        'CenterNet HourGlass104 1024x1024' : 'https://tfhub.dev/tensorflow
            /centernet/hourglass_1024x1024/1',
```

```
        'CenterNet HourGlass104 Keypoints 1024x1024' : 'https://tfhub.dev/
            tensorflow/centernet/hourglass_1024x1024_kpts/1',
        'CenterNet Resnet50 V2 512x512' : 'https://tfhub.dev/tensorflow/
            centernet/resnet50v2_512x512/1',
        'CenterNet Resnet50 V2 Keypoints 512x512' : 'https://tfhub.dev/
            tensorflow/centernet/resnet50v2_512x512_kpts/1',
        'EfficientDet D7 1536x1536' : 'https://tfhub.dev/tensorflow/
            efficientdet/d7/1',
        'SSD MobileNet V2 FPNLite 640x640' : 'https://tfhub.dev/tensorflow
            /ssd_mobilenet_v2/fpnlite_640x640/1',
        'SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)' : 'https://tfhub.
            dev/tensorflow/retinanet/resnet152_v1_fpn_1024x1024/1',
        'Faster R-CNN ResNet152 V1 1024x1024' : 'https://tfhub.dev/
            tensorflow/faster_rcnn/resnet152_v1_1024x1024/1',
        'Faster R-CNN Inception ResNet V2 1024x1024' : 'https://tfhub.dev/
            tensorflow/faster_rcnn/inception_resnet_v2_1024x1024/1',
        'Mask R-CNN Inception ResNet V2 1024x1024' : 'https://tfhub.dev/
            tensorflow/mask_rcnn/inception_resnet_v2_1024x1024/1'
}

model_display_name = 'EfficientDet D7 1536x1536'
model_handle = ALL_MODELS[model_display_name]
hub_model = hub.load(model_handle)
```

Codice A.2: Creazione raccolta di immagini processate e tagliate e formazione dataset con informazione su *score*, coordinate spaziali e classe dell'oggetto rilevato. Inoltre vengono filtrati i *box* con classe = 1 (persona) e *score* > 0.25.

```
directory = 'C:/Tesi/ImmaginiTagliate'

dfImgProcessed_list = []
for filename in os.listdir(directory):
        nome = filename
        path = os.path.join('C:/Tesi/ImmaginiTagliate/', filename)
        im = Image.open(path)
        left = 0
```

```
        top = im.size[1]/6
        right = im.size[0]
        bottom = im.size[1]
        im1 = im.crop((left, top, right, bottom))
        image_resized = im1.resize((3072, 1024), Image.ANTIALIAS)
        name = os.path.join('C:/Tesi/ImmaginiTagliate1/', filename)
        im1 = image_resized.save(name)
        image_np = load_image_into_numpy_array(name)
        results = hub_model(image_np)
        score = results["detection_scores"]
        score = np.array(score)
        score = np.reshape(score, -1)
        coord = results["detection_boxes"]
        coord = np.array(coord)
        coord = np.asmatrix(coord)
        classe = results["detection_classes"]
        classe = np.array(classe)
        classe = np.reshape(classe, -1)
        groupedTable = np.column_stack((np.repeat(nome, score.shape),
            score, classe, coord))
        df_temp = pd.DataFrame(groupedTable)
        df_temp.columns = ['nome', 'score', 'classe', 'ymin', 'xmin', '
            ymax', 'xmax']
        df_temp["score"] = df_temp.score.astype(float)
        df_temp["classe"] = df_temp.classe.astype(float)
        df_temp = df_temp.loc[(df_temp['classe'] == 1.0) & (df_temp['score
            '] >= 0.25)]
        dfImgProcessed_list.append(df_temp)

dfImgProcessed = pd.concat(dfImgProcessed_list)
dfImgProcessed.nome = [s.replace(".jpg", "") for s in dfImgProcessed.nome]
```

Codice A.3: Creazione del dataset con variabili rilevanti su giorno, mese, anno, giorno della settimana, ora e parte della giornata a partire dall'informazione del nome della fotografia.

```
directory = 'C:/Tesi/ImmaginiTagliate1'
```

```
dfImgInfo_list = []
for filename in os.listdir(directory):
        nome = filename
        giorno = re.findall("(\d{2})", filename)[3]
        mese = re.findall("(\d{2})", filename)[2]
        anno = re.findall("(\d{4})", filename)[0]
        giorno_settimana = datetime.date(int(anno), int(mese), int(giorno)
            ).strftime('%A')
        ora = re.findall("(\d{2})", filename)[4]
        if (ora == '06' or ora == '07' or ora == '08'):
                parte_giorno = 'mattina'
        elif (ora == '11' or ora == '12' or ora == '13' or ora == '14' or
            ora == '15' or ora == '16'):
                parte_giorno = 'pomeriggio'
        else:
                parte_giorno = 'sera'
        ds = [nome, giorno, mese, anno, giorno_settimana, ora, parte_
            giorno]
        df_temp = pd.DataFrame(ds)
        df_temp = df_temp.transpose()
        df_temp.columns = ['nome', 'giorno', 'mese', 'anno', 'giorno della
             settimana', 'ora', 'parte della giornata']
        dfImgInfo_list.append(df_temp)

dfImgInfo = pd.concat(dfImgInfo_list)
dfImgInfo.nome = [s.replace(".jpg", "") for s in dfImgInfo.nome]

num = np.zeros(dfImgInfo.shape[0])
j = 0
for i in dfImgInfo.nome:
        if (i in dfImgProcessed.values):
                num[j] = dfImgProcessed['nome'].value_counts()[i]
                j += 1
        else:
                j += 1
```

Codice A.4: Creazione dei due dataset finali: il primo (*"dfNumeroPersone"*) per studiare il numero di persone in ciascuna foto ed il secondo (*"dfNumeroGruppi"*) per analizzare il numero dei gruppi nel tempo

```
dfNumeroPersone = np.column_stack((df_immagini, num))
dfNumeroPersone = pd.DataFrame(df_immagini_fin)
dfNumeroPersone.columns = ['nome', 'giorno', 'mese', 'anno', 'giorno della
    settimana', 'ora', 'parte della giornata', 'numero persone']

dfNumeroGruppi = pd.merge(df_immagini, df_proc, how='right', on = 'nome')
```

Appendice B

Raccolta di fotografie elaborate

B.1 Foto elaborate correttamente (Figura B.1 - B.5)

Di seguito sono presentati alcuni esempi di fotografie in cui, nonostante problemi scaturiti da una scarsa visibilità della zona interessata, il modello *EfficientDet* riesce a cogliere perfettamente i soggetti presenti nel campo.

Figura B.1: Fotografia scattata in condizioni di nebbia.

Figura B.2: Fotografia di bassa qualità scattata al tramonto.

Figura B.3: Fotografia scattata all'alba.

Figura B.4: Fotografia del campo affollato con presenza di bambini.

Figura B.5: Fotografia scattata in condizioni di buio elevato.

B.2 Foto elaborate con imperfezioni (Figura B.6 - B.9)

Attraverso un controllo manuale approfondito sulle immagini elaborate, è stato possibile identificare quelle fotografie in cui la modellazione proposta non è riuscita a raggiungere il massimo dell'accuratezza nell'operazione di *object detection*. Queste sono riportate nella presente sezione.

Nelle prime due immagini (Figura B.6 e Figura B.7) l'algoritmo non riesce a cogliere perfettamente tutte le persone nel Campo. Nella prima foto confonde una figura umana con quella di un cane, classificando, di conseguenza nella categoria sbagliata, mentre nella seconda non identifica la coppia di persone vicino al ponte.

Infine nelle Figure B.8 e B.9 ci sono altre due imperfezioni, relative, nel primo caso ad un cane identificato in modo errato, e nel secondo al mancato riconoscimento di una persona nella parte superiore dell'immagine.

Questi esempi servono a mostrare che, anche quando l'algoritmo non si adatta perfettamente all'immagine, l'errore che può generare nella classificazione dei soggetti è molto ridotto.

Figura B.6: Fotografia mossa in condizioni di buio elevato, in cui il modello confonde una persona con un cane.

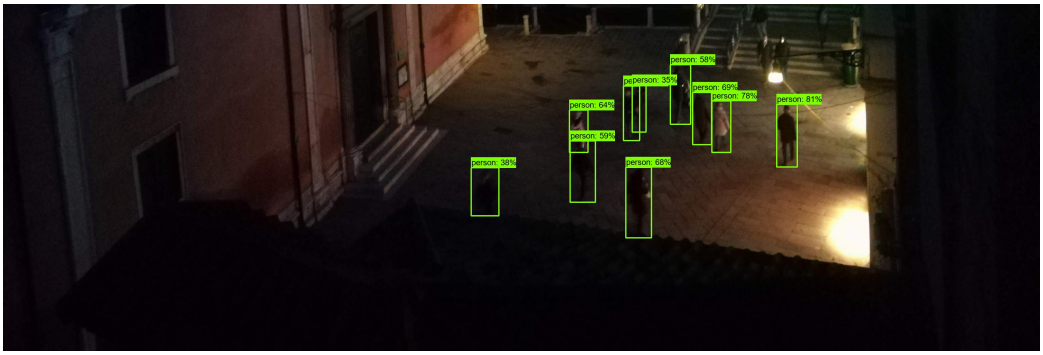

Figura B.7: Fotografia scattata di notte con mancato riconoscimento di una coppia di persone.

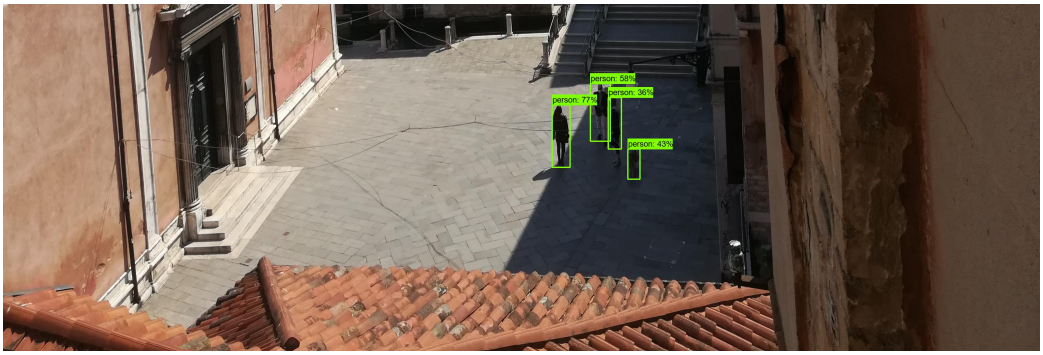

Figura B.8: Immagine in cui l'algoritmo classifica un cane come persona.

Figura B.9: Fotografia con un'alta densità di persone all'interno del Campo, in cui il modello non coglie alla perfezione tutte le persone.

Appendice C

Grafici supplementari

C.1 Analisi per il numero di persone

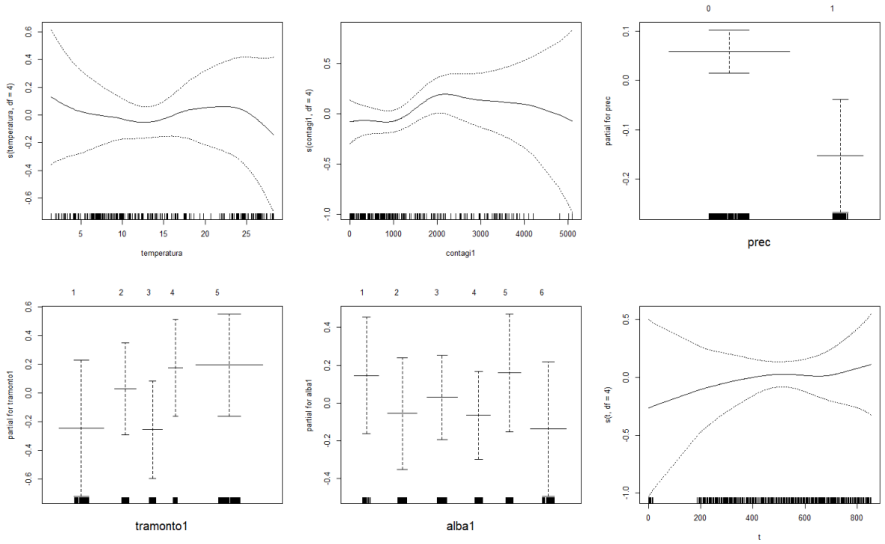

Figura C.1: Grafici modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica su sezione dataset con variabile *parte della giornata* uguale a "Mattina".

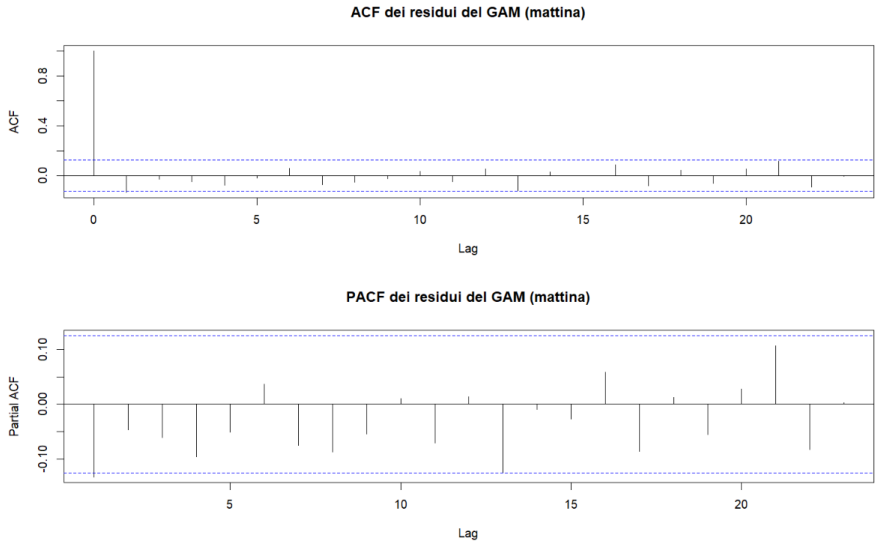

Figura C.2: Grafici di ACF e PACF dei residui del modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica su sezione dataset con variabile *parte della giornata* uguale a "Mattina".

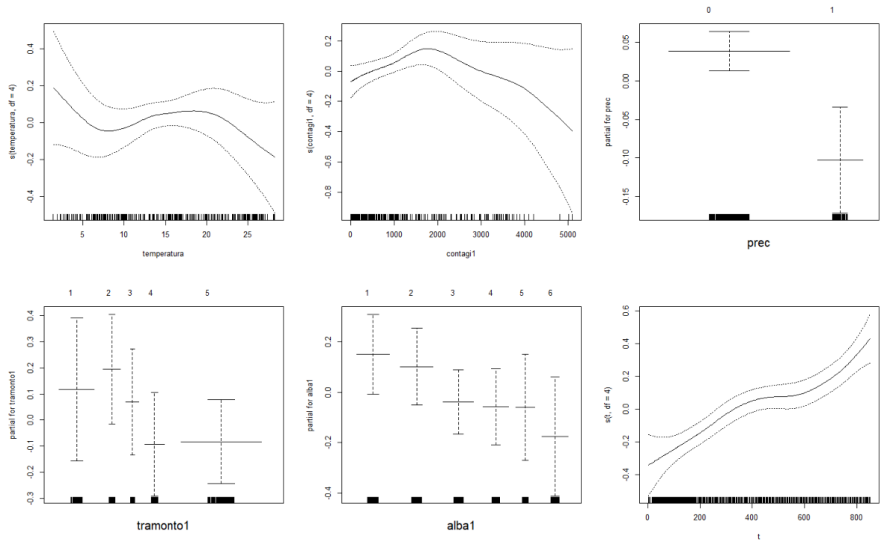

Figura C.3: Grafici modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica su sezione dataset con variabile *parte della giornata* uguale a "Pomeriggio".

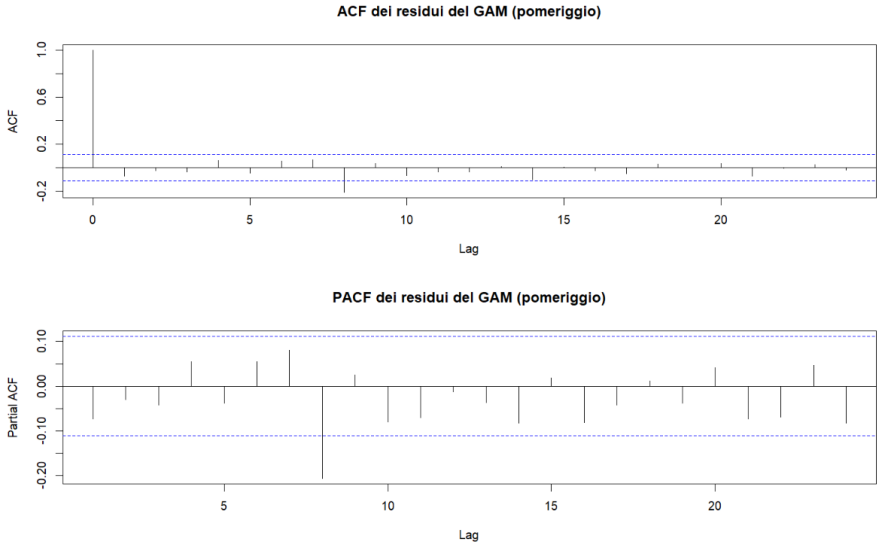

Figura C.4: Grafici di ACF e PACF dei residui del modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica su sezione dataset con variabile *parte della giornata* uguale a "Pomeriggio".

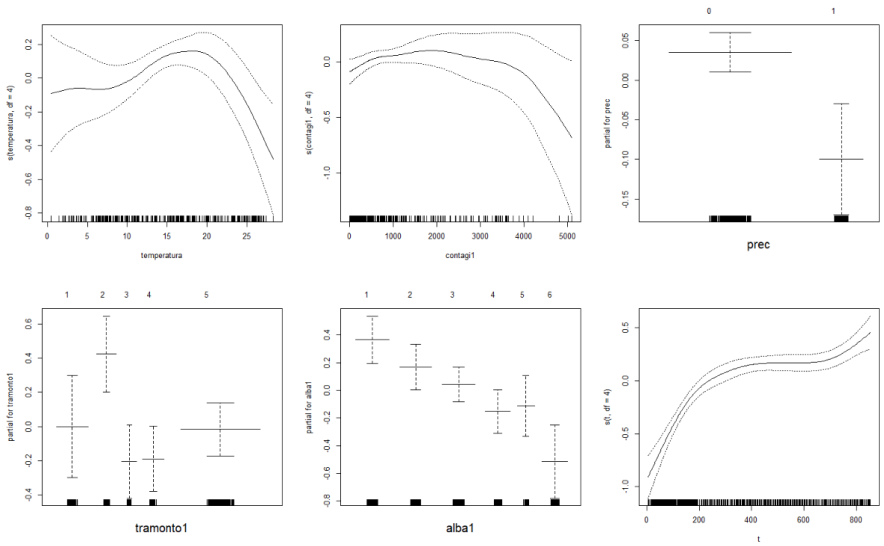

Figura C.5: Grafici modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica su sezione dataset con variabile *parte della giornata* uguale a "Sera".

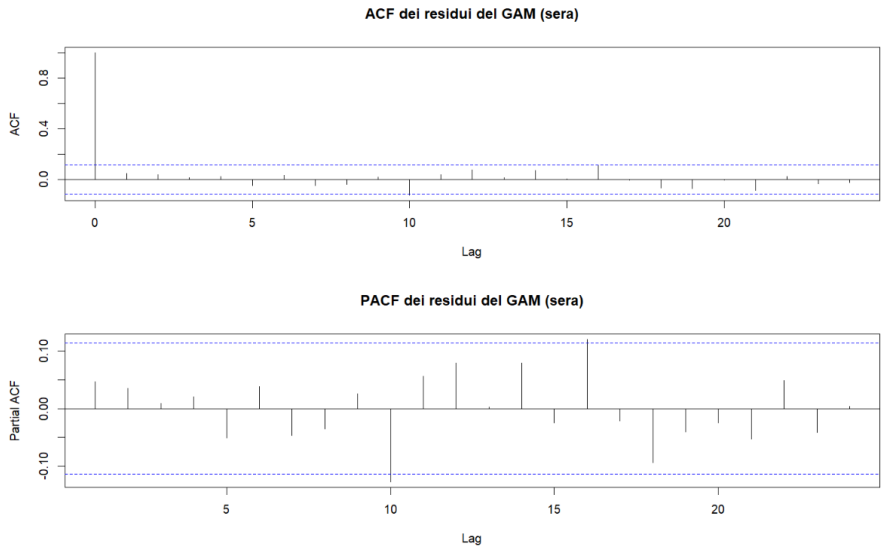

Figura C.6: Grafici di ACF e PACF dei residui del modello additivo sul numero di persone stimato con famiglia "Poisson" e funzione di legame logaritmica per la sezione del dataset con variabile *parte della giornata* uguale a "Sera".

C.2 Analisi per il numero di gruppi

Variabile	**Gradi di libertà**	**p-value**
ANOVA per effetti parametrici		
giorno della settimana	2	0,0012 **
temperatura	1	0,33
contagi	1	0,37
precipitazioni	1	0,30
colore	3	0,071 .
tramonto	4	0,70
alba	5	0,48
tempo	1	0,62
ANOVA per effetti non parametrici		
temperatura	3	0,47
contagi	3	0,47
tempo	3	0,93

Tabella C.1: Test ANOVA sugli effetti parametrici e non parametrici del modello GAM sul numero di gruppi stimato con famiglia "Poisson" e funzione di legame logaritmica, per la sezione del dataset con variabile *parte della giornata* uguale a "Mattina".

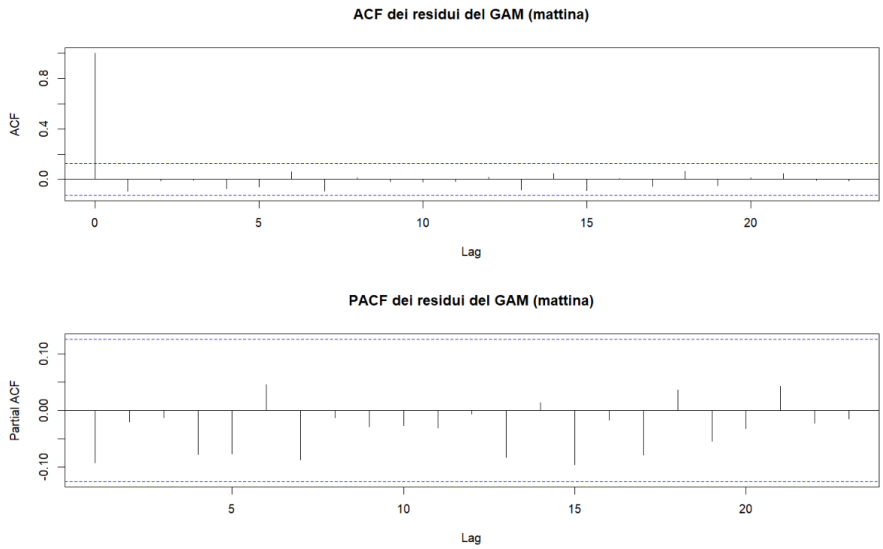

Figura C.7: Grafici di ACF e PACF dei residui del modello additivo sul numero di gruppi stimato con famiglia "Poisson" e funzione di legame logaritmica per la sezione del dataset con variabile *parte della giornata* uguale a "Mattina".

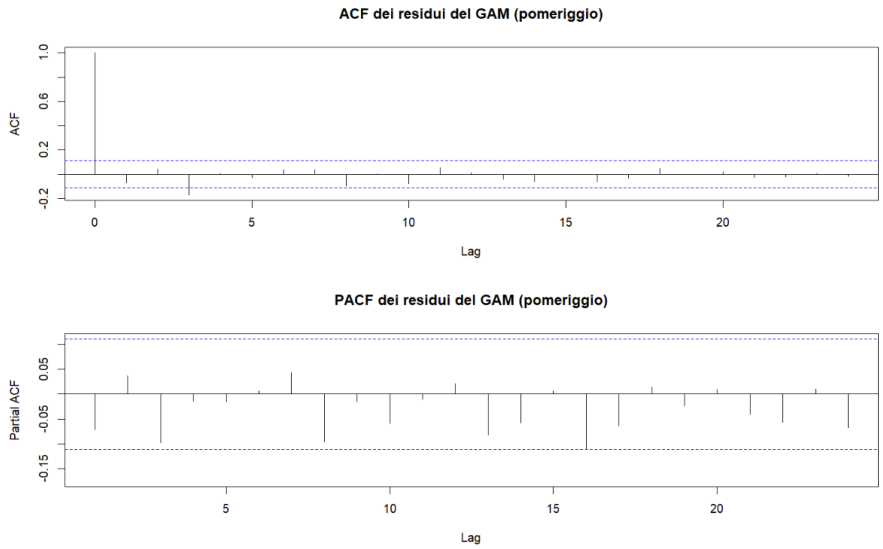

Figura C.8: Grafici di ACF e PACF dei residui del modello additivo sul numero di gruppi per la sezione del dataset con variabile *parte della giornata* uguale a "Pomeriggio".

Variabile	**Gradi di libertà**	**p-value**
ANOVA per effetti parametrici		
giorno della settimana	2	0 ***
temperatura	1	0 ***
contagi	1	0,25
precipitazioni	1	0,26
colore	3	0 ***
tramonto	4	0,16
alba	5	0,53
tempo	1	0 ***
ANOVA per effetti non parametrici		
temperatura	3	0,064 .
contagi	3	0,29
tempo	3	0,40

Tabella C.2: Test ANOVA sugli effetti parametrici e non parametrici del modello GAM sul numero di gruppi stimato con famiglia "Poisson" e funzione di legame logaritmica, per la sezione del dataset con variabile *parte della giornata* uguale a "Pomeriggio".

Variabile	**Gradi di libertà**	**p-value**
ANOVA per effetti parametrici		
giorno della settimana	2	0,0014 **
temperatura	1	0 ***
contagi	1	0,0026 **
precipitazioni	1	0,090 .
colore	3	0 ***
tramonto	4	0 ***
alba	5	0,34
tempo	1	0 ***
ANOVA per effetti non parametrici		
temperatura	3	0,021 *
contagi	3	0,27
tempo	3	0 ***

Tabella C.3: Test ANOVA sugli effetti parametrici e non parametrici del modello GAM sul numero di gruppi stimato con famiglia "Poisson" e funzione di legame logaritmica, per la sezione del dataset con variabile *parte della giornata* uguale a "Sera".

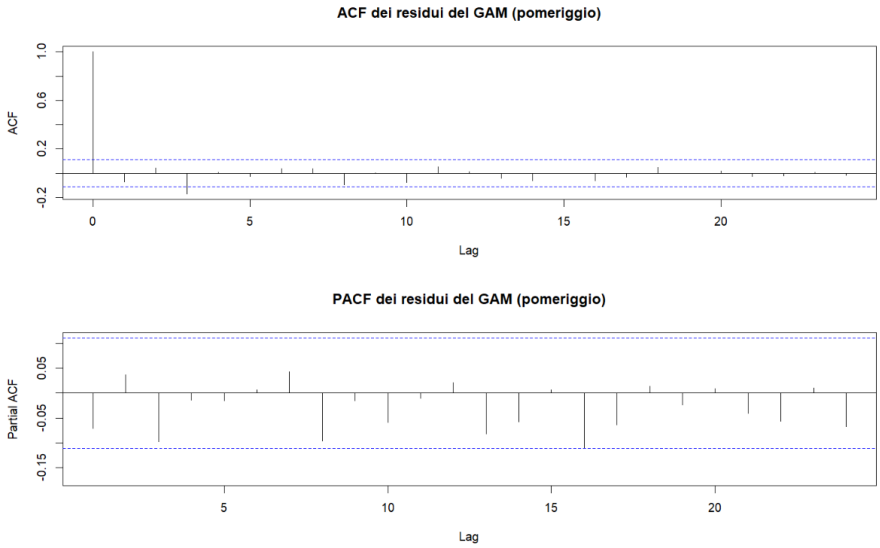

Figura C.9: Grafici di ACF e PACF dei residui del modello additivo sul numero di gruppi per la sezione del dataset con variabile *parte della giornata* uguale a "Sera".

Appendice D

Codice R utilizzato per la fase di *processing*

Codice D.1: Calcolo della costante di proporzionalità

```
x <- NULL
y <- NULL
for (i in rownames(dfNumeroGruppi[which(dfNumeroGruppi$nome == "IMG_
    20210415_130128"),])){
        x = c(x, (dfNumeroGruppi[i, 'xmin'] + dfNumeroGruppi[i, 'xmax'])/
            2)
        y = c(y, dfNumeroGruppi[i, 'ymin'])
}
meuse <- data.frame(
        x = x,
        y = y
)
coordinates(meuse) <- c("x", "y")
coords <- coordinates(cbind(meuse$x, meuse$y))

distanzaAB <- euclidean(coords[1,], coords[2,])
distanzaCD <- euclidean(coords[3,], coords[4,])
diffDistanzeInter <- euclidean(coords[1,], coords[2,]) - euclidean(coords
    [3,], coords[4,])
```

```
altezzaMediaAB <- mean(c(coords[1,2], coords[2,2]))
altezzaMediaCD <- mean(c(coords[3,2], coords[4,2]))
diffAltezze <- mean(c(coords[3,2], coords[4,2])) - mean(c(coords[1,2],
    coords[2,2]))
costanteProp <- diffDistanzaInter/diffAltezze

distanzaNuovaAB <- distanzaAB + (AltezzaMediaAB * costanteProp)
distanzaNuovaCD <- distanzaCD + (AltezzaMediaCD * costanteProp)
```

Codice D.2: Composizione rete e calcolo del numero di gruppi per immagine

```
p <- 1
numero_gruppi <- rep(0, length(unique(dfNumeroGruppi$data)))
for (i in unique(dfNumeroGruppi$data)){
        numero_persone <- dim(dfNumeroGruppi[which(datajoin$data == i),])
            [1]
        if (numero_persone > 1){
                x <- (dfNumeroGruppi[which(dfNumeroGruppi$data == i), '
                    xmin'] +
                        dfNumeroGruppi[which(dfNumeroGruppi$data == i), '
                            xmax'])/2
                y <- dfNumeroGruppi[which(dfNumeroGruppi$data == i), 'ymin
                    ']
                meuse <- data.frame(
                        x = x,
                        y = y
                )
                coordinates(meuse) <- c("x", "y")
                coords <- coordinates(cbind(meuse$x, meuse$y))
                id <- row.names(as.data.frame(coords))
                neigh <- knn2nb(knearneigh(coords, k=1, longlat = T), row.
                    names = id)
                nodoPartenza <- rep(0, length(neigh))
                nodoArrivo <- rep(0, length(neigh))
                dist_vecchia <- rep(0, length(neigh))
                dist <- rep(0, length(neigh))
                media <- rep(0, length(neigh))
```

```
k = 1
for (j in 1:length(neigh)){
        media[k] <- mean(c(coords[j,2], coords[neigh[[j
            ]][1],2]))
        nodoPartenza[k] <- j
        nodoArrivo[k] <- neigh[[j]][1]
        dist_vecchia[k] <- euclidean(coords[j,], coords[
            neigh[[j]][1],])
        dist_nuova[k] <- dist_vecchia[k] + (media[k] *
            costanteProp)
        k = k + 1
}
df <- data.frame(
        nodoPartenza = nodoPartenza,
        nodoArrivo = nodoArrivo,
        distanzaVecchia = dist_vecchia,
        distanzaNuova = dist_nuova
)
dfGruppi <- df[which(df$distanzanuova < 0.1),]
if (dim(dfGruppi)[1] != 0){
        edge_list <- tibble(from = dfGruppi$nodoPartenza,
            to = dfGruppi$nodoArrivo)
        edge_list <- edge_list %>%
                mutate(X1 = coords[from, 1]) %>%
                mutate(Y1 = coords[from, 2]) %>%
                mutate(X2 = coords[to, 1]) %>%
                mutate(Y2 = coords[to, 2])
        edge_list <- edge_list %>%
                mutate(edgeID = c(1:n()))
        dt <- as.data.table(edge_list)
        dt1 <- dt[, .(from, to, edgeID, X = X1, Y = Y1)]
        dt2 <- dt[, .(from, to, edgeID, X = X2, Y = Y2)]
        dt1[, seq := 1L ]
        dt2[, seq := 2L ]
        dt <- rbindlist(list(dt1, dt2), use.names = TRUE)
        setorder(dt, edgeID, seq)
```

```
                        sf <- sfheaders::sf_linestring(
                                obj = dt, x = "X", y = "Y", linestring_id
                                    = "edgeID")
                        edge_list <- as_tibble(sf)
                        edge_list <- edge_list %>%
                                mutate(from = dfGruppi$nodoPartenza) %>%
                                mutate(to = dfGruppi$nodoArrivo)
                        node_list <- tibble(nodeID = 1:length(neigh)) %>%
                                mutate(X = coords[,1]) %>%
                                mutate(Y = coords[,2])
                        node_list <- node_list %>%
                                st_as_sf(coords = c('X', 'Y'))
                        net <- tbl_graph(nodes = node_list, edges = edge_
                            list, directed = FALSE)
                        net.louvain <- net %>%
                                activate("nodes") %>%
                                mutate(louvain = as.factor(group_louvain()
                                    ))
                        gruppi_louvain <- net.louvain %>%
                                activate(nodes) %>%
                                as_tibble()
                        numero_gruppi[p] <- sum(table(gruppi_louvain$
                            louvain) > 1)
                        x <- table(gruppi_louvain$louvain) > 1
                        p <- p + 1
                }
                else{
                        numero_gruppi[p] <- 0
                        p <- p + 1
                }
        }
        else {
                numero_gruppi[p] <- 0
                p <- p + 1
        }
}
```