

La raccolta differenziata, un problema di classificazione

Modelli previsionali e applicazioni

Candidato:
Giacomo Saccaggi

Relatore:
Ch.mo Prof. Nicola Lunardon

Correlatore:
Ch.mo Prof. Matteo Borrotti

Facoltà di
Scienze Statistiche ed Economiche



Dipartimento di Statistica

Università Bicocca

Sede di Milano

Anno Accademico 2018/2019

Alla mia famiglia.

Abstract

Il sistema industriale odierno sta migrando sempre più verso materie prime riciclate. In questo contesto il riciclo e la raccolta differenziata ricoprono un'importanza cruciale. L'obiettivo di questo lavoro vuole essere quello di affrontare il problema della raccolta differenziata con un approccio scientifico e proporre alcune idee e soluzioni di come il Data Science possa essere d'aiuto nel riuscire a migliorare lo sfruttamento delle risorse. Le domande della ricerca, quindi, sono due: “Come si può analizzare la raccolta differenziata come un problema di classificazione statistica?” e “Quali applicazioni potrebbero essere implementate con l'utilizzo dei modelli previsionali?”. Per rispondere a tali domande si è deciso di reperire i dati per poter classificare i rifiuti utilizzando tre strategie: Web Scraping, programmazione di un Chatbot di Telegram e infine progettazione e costruzione con tecnologie Arduino di un cestino che potesse registrare i dati dei rifiuti. Una volta ultimata la fase di raccolta dei dati si è deciso di utilizzare algoritmi di Deep Learning e Machine Learning per classificare nel modo migliore i rifiuti, successivamente questi modelli sono stati combinati tramite meccanismi di Ensemble Learning così da trovare un previsore migliore, con il quale si sono costruite due applicazioni: un Chatbot che dalle foto riconoscesse la tipologia di rifiuto e un cestino di raccolta differenziata automatica.

Indice

Introduzione	1
1 Raccolta dei dati	5
1.1 Struttura del lavoro	5
1.2 Come differenziare	6
1.3 Web Scraping	7
1.3.1 Funzionamento	7
1.3.2 Implementazione	8
1.4 Bot Telegram	12
1.4.1 Funzionamento	12
1.4.2 Implementazione	12
1.5 Il cestino	14
1.5.1 Meccanica	14
1.5.2 Robotica	18
1.5.3 Softwaristica	26
2 Modellizzazione e addestramento	29
2.1 Descrizione tipologie di dati	29
2.2 Dati Strutturati	35
2.2.1 Gradient boosting algorithm (GBM)	35
2.2.2 Multi-Layer Perceptron (MLP)	39
2.2.3 Risultati	45
2.3 Dati non strutturati	49
2.3.1 Convolutional Neural Network (CNN)	49
2.3.2 Residual Neural Network (ResNet)	54

2.3.3	Risultati	60
3	Performance e previsioni	62
3.1	Confronto modelli	62
3.2	Ensemble Learning	64
3.2.1	Modelli Stacking	64
3.2.2	Risultati	66
3.3	Esempi pratici di applicazioni	70
3.3.1	Bot di Telegram	70
3.3.2	Cestino	70
Conclusioni		71
A	Script di programmazione	72
A.1	Reperimento dei dati	72
A.1.1	Web Scraping (R)	72
A.1.2	Bot Telegram per salvare i dati (Python)	76
A.1.3	Arduino (C++)	77
A.1.4	Software per salvare i dati (VB.NET)	81
A.2	Addestramento dei modelli	95
A.2.1	Gradient boosting algorithm (Python)	95
A.2.2	Multi-Layer Perceptron (Python)	96
A.2.3	Residual Neural Network (Python)	96
A.3	Applicazioni	99
A.3.1	Modello di Ensemble Lening (Python)	99
A.3.2	Software per differenziare (VB.NET)	99
A.3.3	Bot Telegram per differenziare (Python)	99

Introduzione

L'economia italiana è oggi la più *performante* d'Europa per circolarità di materia, produttività delle risorse e capacità di riciclo. A dimostrarlo sono i numeri del rapporto “*L'Economia Circolare in Italia – la filiera del riciclo asse portante di un'economia senza rifiuti*” [1]. Il documento, pubblicato nel gennaio del 2019 e curato dall'esperto ambientale Duccio Bianchi di Ambiente Italia, rappresenta il primo vero bilancio sulla “circolarità” nazionale, settore che vale oggi 88 miliardi di fatturato e 22 miliardi di valore aggiunto, ovvero l'1.5 % del PIL.

In questo libro emerge come l'Italia sia attualmente capofila in Europa nei tre indice che l'autore definisce fondamentali per valutare un'economia circolare: **tasso di produttività nell'uso delle risorse** (l'ammontare di euro di PIL prodotte per ogni kg di risorse consumate), **tasso di circolarità della materia nell'economia** (la quantità di materie seconde impiegate sul totale dei consumi di materia) e infine il **tasso di riciclo dei rifiuti** (il volume di rifiuti, urbani e non urbani, inclusi l'import ed export, destinati al riciclo internamente). Le *performance* nazionali risultano non solo superiori alla media UE, ma anche alle prestazioni dei principali stati come Germania, Spagna, Regno Unito e Francia.

Il riciclo è un processo di conversione che trasforma i rifiuti in nuovi materiali, oggetti o sostanze del tutto differenti dai rifiuti d'origine. Questo processo porta quattro vantaggi:

- Conservazione delle risorse: i materiali ricavati vengono convertiti in nuovi prodotti, riducendo la necessità di estrarre materie prime dalla Terra, attraverso l'estrazione e la silvicoltura. Il riciclaggio aiuta a conservare importanti materie prime e protegge gli habitat naturali;

inoltre, può essere una grande possibilità di evitare importazioni per territori come quello italiano scarso di risorse naturali.

- Risparmio di energia: l'uso di materiali riciclati nel processo di produzione consuma molta meno energia di quella necessaria per la produzione di nuovi prodotti; inoltre, si ottiene un ulteriore risparmio energetico perché è necessaria più energia per estrarre, raffinare, trasportare e elaborare materie prime pronte per l'industria rispetto al riciclo.
- Protezione dell'ambiente: il riciclaggio riduce la necessità di estrazione, raffinazione e lavorazione di materie prime che creano un notevole inquinamento dell'aria e dell'acqua; inoltre, poiché il riciclaggio consente di risparmiare energia, riduce anche le emissioni di gas serra, il che aiuta a contrastare i cambiamenti climatici.
- Riduzione dell'accumulo di rifiuti: i materiali riciclabili vengono ri elaborati in nuovi prodotti e, di conseguenza, la quantità di rifiuti inviati alle discariche si riduce.

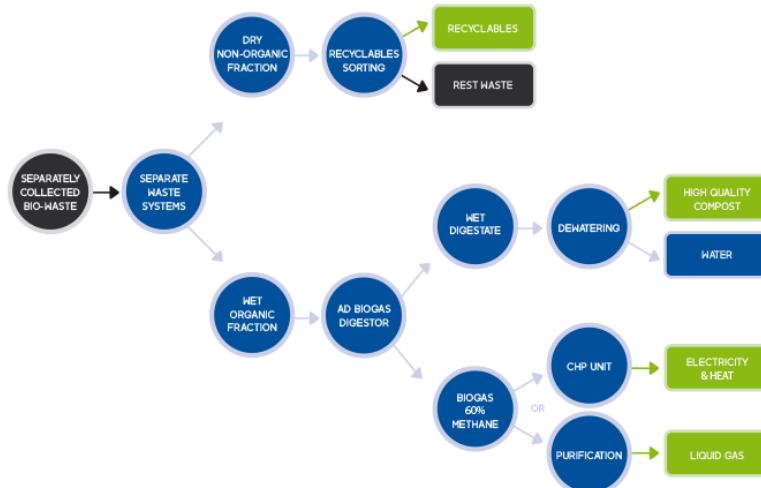


Figura 1: Diversi possibili riutilizzi rifiuti riciclati. (Fonte: separate waste systems).

In quest'ambito subentra il concetto di economia circolare ossia un sistema economico volto ad eliminare gli sprechi e l'uso continuo delle risorse. I sistemi circolari impiegano il riutilizzo, la condivisione, la riparazione, il rinnovo, la rigenerazione e il riciclaggio per creare un sistema a circuito chiuso, riducendo al minimo l'uso di *input* di risorse e la creazione di rifiuti, inquinamento ed emissioni di carbonio. L'economia circolare mira a mantenere i prodotti, le attrezzature e le infrastrutture in uso più a lungo rispetto invece alla classica economia lineare, che si basa sull'utilizzo di risorse sempre nuove e l'eliminazione di quelle vecchie.



Figura 2: Economia circolare. (Fonte: Ecosport economia circolare).

Tra le tante eccellenze italiane nell'ambito del riciclo una delle più importanti riguarda il sistema del recupero e del riciclo del legno che, come viene evidenziato nell'articolo del giornale Il Sole 24 Ore da Giovanna Mancini [2], in poco più di 20 anni ha creato una nuova economia che ha prodotto risultati

importanti sia in termini ambientali, sia per la capacità di creare sviluppo e occupazione. L'impatto economico sulla produzione nazionale delle attività della filiera del recupero del legno *post consumo* è stimabile in circa 1.4 miliardi di euro, mentre il contributo sull'occupazione è di quasi seimila posti di lavoro complessivamente sostenuti in Italia.

In un sistema industriale che sta migrando verso materie prime riciclate, la raccolta differenziata, ossia il processo mediante il quale i rifiuti vengono separati in diversi elementi, ricopre un'importanza cruciale, in quanto più essa viene effettuata con cura e controllo e più il prodotto finale sarà “puro”. Questa procedura è essenziale affinché le aziende che si occupano di riciclo possano operare nel settore.

L'obiettivo di questo lavoro vuole essere quello di affrontare il problema della raccolta differenziata da un punto di vista scientifico e proporre alcune idee e soluzioni di come il *Data Science* possa essere d'aiuto nel riuscire a migliorare lo sfruttamento delle risorse. La dinamica della raccolta differenziata si declina in un problema di classificazione; durante il lavoro si andranno a sfruttare sia dati strutturati, che dati non strutturati, utilizzando algoritmi di *Deep Learning* per riuscire a classificare in modo accurato i differenti tipi di rifiuti.

Le domande della ricerca, quindi, sono due: “Come si può analizzare la raccolta differenziata come un problema di classificazione statistica?” e “Quali applicazioni potrebbero essere implementate con l'utilizzo dei modelli previsionali?”.

Capitolo 1

Raccolta dei dati

1.1 Struttura del lavoro

Il quesito di questo progetto ha implicato uno studio e delle conoscenze trasversali ai problemi di classificazione statistica, in quanto si è dovuto fare un’analisi “completa”: dalla costruzione del *data set* fino all’implementazione di applicazioni.

Per affrontare questo problema si è deciso di strutturare il lavoro in quattro fasi: reperimento dei dati, elaborazione, previsione e, infine, scelta del modello. Per il carattere applicativo del progetto durante tutto il lavoro si approfondiranno la teoria dei modelli e degli strumenti usati andando ad evidenziare alcune implementazioni ritenute utili ai fini della comprensione, per le altre si riportano gli script nell’Appendice.

Nella prima fase, si è deciso di reperire i dati per poter classificare i rifiuti utilizzando tre strategie: Web Scraping, programmazione di un *Chat Bot di Telegram* e costruzione di un cestino che potesse registrare i dati dei rifiuti. Ognuna di queste strategie è stata scelta per ragioni differenti.

Il *Web Scraping* è stata scelta come strategia iniziale perché il *World Wide Web* costituisce una fonte quasi infinita di dati (in questo caso di immagini) e non usarlo sarebbe stata una scelta che avrebbe portato all’esclusione di risorse determinanti.

Il *Bot di Telegram* è stato scelto come strumento per avere immagini più precise riguardo alle macrocategorie più comuni di rifiuti, quali: carta,

plastica e vetro.

Infine, si è deciso di costruire un cestino così da poter raccogliere dei dati selezionati reputati importanti nel classificare gli oggetti. Come tecnologia di implementazione in questa prima fase si è deciso di utilizzare la piattaforma hardware Arduino per la sua semplicità e versatilità. Per la realizzazione di questa parte del progetto, è stato necessario costruire un software che fosse in grado di registrare e salvare i dati inviati dall'Arduino così da automatizzare e sveltire il più possibile il processo di costruzione del dataset.

La fase successiva costituisce il fulcro di questo progetto, in quanto analizzando i dati raccolti precedentemente si è cercato di comprendere come, sfruttando algoritmi di *Deep Learning* e *Machine Learning*, si riuscisse a classificare nel modo migliore i rifiuti.

Nell'ultima fase si sono combinati tramite meccanismi di *Ensemble Learning* i vari classificatori, così da trovare un previsore migliore, con il quale si sono costruite due applicazioni per fare la raccolta differenziata.

1.2 Come differenziare

Il regolamento della raccolta differenziata può differire da zona a zona. Come regole di riferimento si è deciso di utilizzare il regolamento vigente nella città metropolitana di Milano. Il capoluogo lombardo, infatti, è diventato il Comune che ha raggiunto la quota record del 54% di raccolta differenziata, situandosi in cima alla classifica italiana e al secondo posto in Europa dopo Vienna. Un risultato importante e un fenomeno che ha catturato anche l'attenzione di New York, il cui Assessorato all'ambiente ha deciso di studiare il *modus operandi* di Milano per adattarlo alla metropoli americana.

Nei prossimi capitoli si parlerà genericamente di plastica, carta o altro rifiuto. In questa sezione si descrive quali materiali o oggetti rientrano nelle rispettive categorie.

Innanzitutto va ricordato come esistano cinque differenti tipi di rifiuti: rifiuti generici, plastica e metallo, vetro, carta e cartone e rifiuti organici, definiti come segue:

- Indifferenziato: tutti i rifiuti generici come i piatti rotti, la ceramica in generale, la carta sporca e oleata, cd, DVD, videocassette e musicas-

sette, piccoli accendini, filtri dell'aspirapolvere, pannolini e assorbenti e mozziconi di sigaretta.

- Carta: vi finiscono tutti gli oggetti di carta o cartone quali giornali, riviste, libri e quaderni privati delle parti di plastica, adesive e in metallo, ma anche i contenitori in Tetra Pak (quelli usati per il latte, i succhi di frutta e così via), le scatole in cartone e i cartoni della pizza senza avanzi.
- Plastica: le plastiche e i metalli, inclusivi di bottiglie, flaconi e sacchetti di plastica, le vaschette per gli alimenti anche in polistirolo, tutte le scatolette e i barattoli per alimenti una volta sciacquate, lattine per bevande, tubetti di plastica (del dentifricio, per esempio), fogli di alluminio, pellicole per imballaggio, nonché oggetti in metallo come pentole, posate, caffettiere, tappi, capsule, chiavi e lucchetti.
- Vetro: tutti i materiali in vetro come bottiglie e bicchieri, vasi, caraffe e barattoli. Da non inserirvi invece lampadine, specchi e gli oggetti in cristallo.
- Umido: tutti i rifiuti di natura organica in tutte le loro parti, inclusivi di frutta, frutta secca, verdura, carne e pesce, pane, riso, pasta, scarti di cucina, avanzi, fondi di caffè, filtri di tè e tisane, fiori, semi e foglie, e alimenti avariati.

Per una questione igienica intrinseca nella categoria dei rifiuti organici non è stato possibile in alcuni modelli e applicazioni prenderli in analisi, in quei casi, pertanto, si è deciso di considerarli rifiuti generici.

1.3 Web Scraping

1.3.1 Funzionamento

Il Web scraping [3] (chiamato anche web harvesting, o web data extraction) è un insieme di tecniche utilizzate per l'estrazione di dati dai siti Web. È una forma di copia, in cui i dati specifici vengono raccolti e copiati accedendo al

World Wide Web direttamente utilizzando il protocollo Hypertext Transfer Protocol o tramite un browser Web. In genere il termine si riferisce a processi automatizzati implementati utilizzando un web crawler, un bot o un server per il successivo recupero o analisi in un database locale centrale o foglio di calcolo.

Tra le tante possibili soluzioni si è deciso di utilizzare la tecnica di Web Scraping chiamata HTML Parsing ossia una deserializzazione delle pagine HTML. Questo processo riceve il codice HTML non elaborato, lo interpreta e genera dal codice una struttura ad albero DOM (Document Object Model).

Questa tecnica è molto versatile in quanto gran parte dei siti Web hanno grandi raccolte di pagine generate dinamicamente da un'origine strutturata sottostante come un database. I dati della stessa categoria sono in genere codificati in pagine simili da uno script o un modello comune. Nel data mining, un programma che rileva tali modelli in una particolare fonte di informazioni, ne estrae il contenuto e lo traduce in un modulo relazionale, viene chiamato wrapper. Gli algoritmi di generazione dei wrapper presuppongono che le pagine di input di un sistema di induzione di wrapper siano conformi a un modello comune e che possano essere facilmente identificate in termini di schema comune URL.

1.3.2 Implementazione

Per fare Web Scraping si è deciso di utilizzare il linguaggio R, questo perchè oltre ai pacchetti appositi che aiutano nell'operazione di creazione dello wrapper in R sono presenti delle ottime librerie per fare text mining, la più famosa “stringr”, che aiutano a gestire operazioni di HTML parsing più complicate.

La libreria utilizzata in questa fase è “rvest” (creata da Hadley Wickham [aut, cre], RStudio [cph]) la quale gestisce e migliora alcune funzioni contenute nel pacchetto “xml2” (creata da Hadley Wickham [aut], Jim Hester [aut, cre], Jeroen Ooms [aut], RStudio [cph], R Foundation [ctb]), queste due librerie si basano su un pacchetto creato per il linguaggio C++ da Daniel Veillard (versione definitiva pubblicata alla fine del 2012) di nome “libxml2”. Le funzioni che si sono utilizzate di queste librerie sono tre: *read_html()* del pacchetto “xml2”; *xml_attrs()* e *html_nodes()* del pacchetto “rvest”.

La prima funzione `read_html()` serve per deserializzare le pagine HTML e trasformarle in una lista strutturata come un albero DOM.

Script di programmazione 1.1: Deserializzazione delle pagine HTML

```
install.packages("xml2")
library("xml2")
page<-read_html("###---link---###")
```

Le due funzioni del pacchetto “`rvest`”, invece, aiutano ad individuare i diversi elementi del documento. In questo caso si utilizzano congiuntamente per trovare il link di origine delle immagini per poi scaricarle.

Script di programmazione 1.2: Attributi nella struttura ad albero DOM

```
install.packages("rvest")
library("rvest")
t<-1 # img t-esima presente nella pagina
link_img<-xmlAttrs(html_nodes(page, xpath =
  "//img")[[t]][["src"]])
```

Infine una volta trovati i link delle immagini presenti nelle varie pagine si salvano tramite la funzione `download.file()`

Script di programmazione 1.3: Scaricare immagini

```
download.file(link_img, destfile="PATH", method='curl')
```

Per la ricerca dei siti dove scaricare le immagini si è sfruttato il funzionamento alla base delle ricerche sui più comuni motori di ricerca per immagini: Google, Bing e Pixabay.

La logica che sta alla base delle ricerche online permette di trovare tutte le immagini collegate inserendo una o più parole chiave. Quindi si è deciso di creare delle liste di parole per ogni categorie rifiuto e di trovare le immagini collegate per le differenti Keywords.

In termini generali si è creato un *array* con una serie di parole chiave, queste sono state combinate all'interno di un ciclo *for* creando, attraverso le funzioni `gsub()` e `paste0()`, un URL che potesse comunicare con il motore di

ricerca.

Le parole chiavi scelte per l'analisi:

Categoria	Parole chiave
Plastica	“Plastica”, “bottiglie di plastica”, “sacchetti di plastica”, “tappi di plastica”, “giochi di plastica”, “utensili di plastica”, “pacchi di plastica”, “posate di plastica”, “bicchieri di plastica”, “vaschette di plastica”, “tubetti dentifricio”, “pellicola cucina”, “sedie di plastica”, “secchi di plastica”, “vaschette plastica”, “contenitore uova di plastica”
Carta	“Carta”, “carta giornali”, “fogli di carta”, “contenitore uova di carta”, “tovaglioli di carta”, “pacchi di cartone”, “cartone pizza”, “posate di carta”, “bicchieri di carta”, “libri di carta”, “quaderni di carta”, “sacchetti di carta”, “tovaglie di carta”, “scatole di cartone”, “riviste di carta”, “carta appallottolata”
Tetra Pak	“Tetra Pak latte”, “Tetra Pak succhi”, “contenitori in tetra pak”, “tetra pak succo brico”
Polistirolo	“contenitori in polistirolo”, “polistirolo”, “vaschette polistirolo”
Vetro	“Vetro”, “Bottiglie di vetro”, “tazzine di vetro”, “vetro rotto”, “vetro bottiglie vino”, “vetro bottiglie birra”, “vetro bottiglie alcolici”, “vetro bottiglie bevante”, “caraffe di vetro”, “calici di vetro”, “vasi di vetro”, “bicchieri di vetro”, “bicchieri di vetro rotti”, “lastre di vetro”, “contenitori di vetro”
Indifferenziato	“ceramica”, “carta sporca e oleata”, “cd”, “dvd”, “videocassette”, “musicassette”, “accendini”, “pannolini”, “assorbenti”, “mozziconi di sigarette”, “lampadine”, “cristallo”, “specchi”, “vasi ceramica”, “filtri dell’aspirapolvere”
Legno	“legno”, “legname”, “giochi di legno”, “utensili di legno”, “assi di legno”, “posate di legno”, “ciotole di legno”, “mestoli di legno”, “sedie di legno”, “tavoli di legno”, “piatti di legno”, “scaffali di legno”, “mobili di legno”, “tagliere di legno”

Metalli	“ferro”, “metallo”, “pentole”, “mestoli di ferro”, “lastre metallo”, “chiavi di ferro”, “utensili di ferro”, “ferro battuto”, “padelle”, “posate di metallo”, “forchette di metallo”, “coltelli di metallo”, “cucchiai di metallo”, “caffettiere”, “Lucchetti di ferro”
Umido	“scarti cibo”, “umido cibo”, “organico cibo”, “scarti pesce”, “scarti carne”, “fondi di caffè”, “frutta secca”, “avanzi cibo”, “filtri te e tisane”, “pane”, “fiori e foglie”, “alimenti avariati”, “scarti di cucina”, “avanzi pasta”, “avanzi di riso”
Latta e alluminio	“lattine per bevande”, “fogli di alluminio”, “lattine di coca-cola”, “lattine di pepsi”, “lattine redbull”, “lattine di sprite”, “lattine di 7up”, “lattine di fanta”, “lattine di birra”, “lattine di pelati”, “lattine di cereali”, “lattine di legumi”, “alluminio per cucina”, “teglie di alluminio”, “lattine”

La funzione per scaricare le immagini:

Script di programmazione 1.4: Web Scraping

```
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
 ,gsub(" ","+",parola_chiave[i]),"+jpg&source=lnms&tbo=isch")))
 for (t in 1:length(html_nodes(page,xpath = '//img')))) {
   txt<-c(txt,xml_attrs(html_nodes(page, xpath =
   "//img")[[t]])[["src"]])
 })
 try({page<-read_html(paste0("https://www.bing.com/images/search?q="
 ,gsub(" ","+",parola_chiave[i]),"+jpg&FORM=HDRSC2")))
 for (t in 1:length(html_nodes(page,xpath = '//img')))) {
   txt<-c(txt,xml_attrs(html_nodes(page, xpath =
   "//img")[[t]])[["src"]])
 })
 try({page<-read_html(paste0("https://pixabay.com/it/images/search/"
 ,gsub(" ", "%20",parola_chiave[i])))
 for (t in 1:length(html_nodes(page,xpath = '//img')))) {
   txt<-c(txt,xml_attrs(html_nodes(page, xpath =
   "//img")[[t]])[["src"]])
 }})
```

Così facendo si è stati in grado di trovare i links di tutte le immagini presenti nel sito. Nella funzione sopra descritta si sono spesso utilizzati dei *try()* per evitare gli errori 404, molto comuni quando si automatizza il parsing HTML. Funzione al completo descritta nell'Appendice A.1.1 .

1.4 Bot Telegram

1.4.1 Funzionamento

Telegram è una applicazione di messaggistica gratuita creata e ideata a Berlino da Pavel Durov e Nikolai Durov. La prima particolarità di Telegram è quella di basarsi su un protocollo di comunicazione completamente open source, sviluppato per ridurre al minimo la quantità di byte inviati per ogni messaggio. Questa soluzione permette di avere una maggiore velocità anche in condizioni di scarsa recezione. La seconda qualità di Telegram riguarda l'elevato livello di sicurezza, infatti, grazie all'utilizzo di algoritmi di criptazione interni alle chat aumenta notevolmente il livello di privacy.

Secondo la definizione classica data al termine Bot (abbreviazione di “robot”), è una chat con la quale è possibile inviare input ad un server tramite comandi precedentemente programmati e ricevere output direttamente nella chat e/o sul server. I chatbot stanno diventando sempre più importanti in un mondo nel quale si cerca di automatizzare il più possibile ogni tipo di processo, anche le grandi aziende si stanno spingendo verso quella direzione si pensi ad esempio a “TOBI” chatbot di Vodafone con il quale è possibile chattare sull'app o sul sito della società di telecomunicazioni per aiutare il cliente a risolvere eventuali problemi.

1.4.2 Implementazione

Il Bot implementato in questa sezione ha come funzione quella di ricevere immagini per categoria di rifiuto, salvarle e restituire a colui che ha inviato l'immagine un messaggio di ringraziamento. Nel programmare tale Bot si è utilizzato il pacchetto di Python “Telepot” grazie al quale si riesce ad inviare e ricevere (GET e POST request nel linguaggio PHP) richieste al server direttamente dalla chat. In particolare, questa libreria permette di identificare

la tipologia di messaggio che il soggetto invia sulla chat tramite la funzione “content_type” e istruirlo a compiere determinate azioni nel caso vengano inviate determinate tipologie di messaggio. Il bot in questione doveva salvare l’immagine con la funzione “bot.download_file(file, path)” e restituire un messaggio di ringraziamento attraverso la funzione “bot.sendMessage(chat_id, message)”.

Script di programmazione 1.5: Scaricare foto da chatbot di Telegram

```
if content_type == 'photo':
    bot.download_file(msg['photo'][1]['file_id'], 'dati/image'+
                      str(datetime.now())[0:19] +'.png')
    bot.sendMessage(chat_id, 'Thanks for send me the photo! See you
soon!')
```

Si è deciso di creare 3 bot, visualizzati in Figura 1.1, per le tre categorie principali di rifiuti: Plastica, Carta e Vetro.

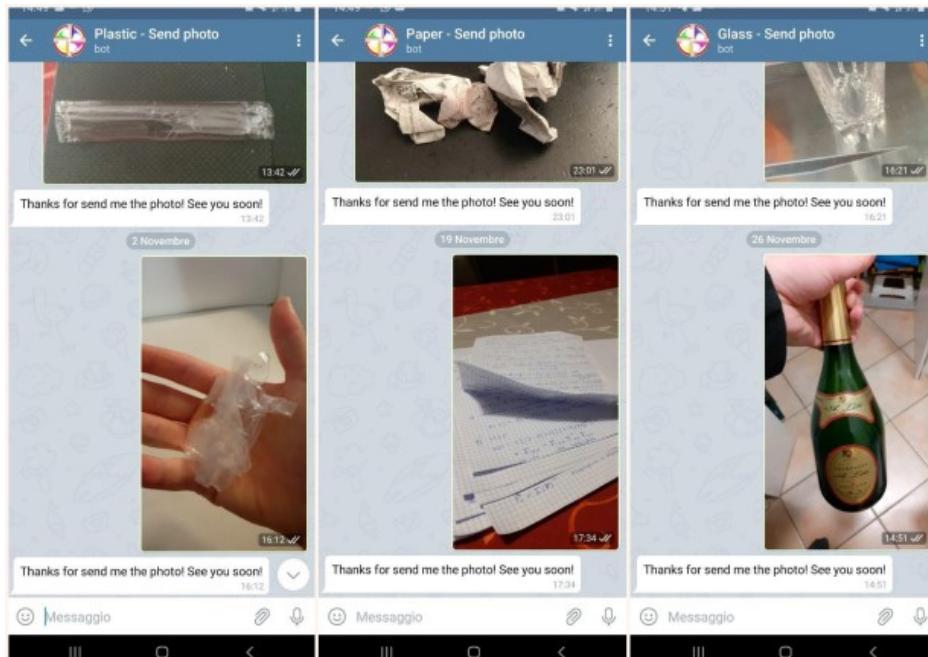


Figura 1.1: Funzionamento Bot di Telegram

Per il funzionamento del Bot si è deciso di utilizzare un server collegato ad

una rete locale piuttosto che un VPS (Virtual Private Server) per due ragioni: da un lato, non avendo la necessità di una grande capacità computazionale, risultava più conveniente in termini economici e dall’altro erano più semplici eventuali modifiche in corso d’opera.

1.5 Il cestino

Come ultima fonte per reperire i dati si è deciso di costruire un cestino così da poter raccogliere anche dati di tipo strutturato sui rifiuti. La scelta di questa terza modalità di raccolta di informazioni da avere un dataset molto preciso circa le diverse tipologie di rifiuto. Nel costruire il cestino si è dovuto progettare un lavoro di tipo “meccanico” per far sì che il cestino funzionasse dal punto di vista strutturale; di tipo “robotico” in quanto questo cestino doveva essere in grado di muoversi e inviare i dati ricevuti dai sensori al computer; e, infine, di tipo “softwaristico” in quanto si è dovuto programmare un software che fosse in grado di ricevere e salvare i dati inviati dal cestino in fase iniziale e in una fase successiva riconoscere l’oggetto attraverso i dati ricevuti e riuscire a comandare il cestino per porre il rifiuto nell’apposito comparto.

1.5.1 Meccanica

Nel costruire il cestino per semplicità si è pensato di dividere la struttura in due unità: la struttura inferiore (o struttura reggente) che ha come scopo quello di contenere i comparti delle categorie di rifiuto e di reggere l’assetto; la struttura superiore (o il “cervello”) ossia una telaio mobile sul quale vengono posti tutti i componenti elettronici che hanno come scopo quello di interpretare i messaggi inviati dal cestino, interfacciarsi con l’utente e fare la raccolta differenziata ruotando sopra la struttura reggente.

La struttura inferiore, Figura 1.2, è composta da una base circolare di dimensioni 60x60x5 cm da cui si ergono 4 pilastri di dimensione 1x1x20 cm che sorreggono un cerchio di raggio 60 cm e di dimensione della linea 5 cm e spessore 1 cm su cui vengono poste delle ruote che permettono alla struttura superiore di muoversi in fluidità.

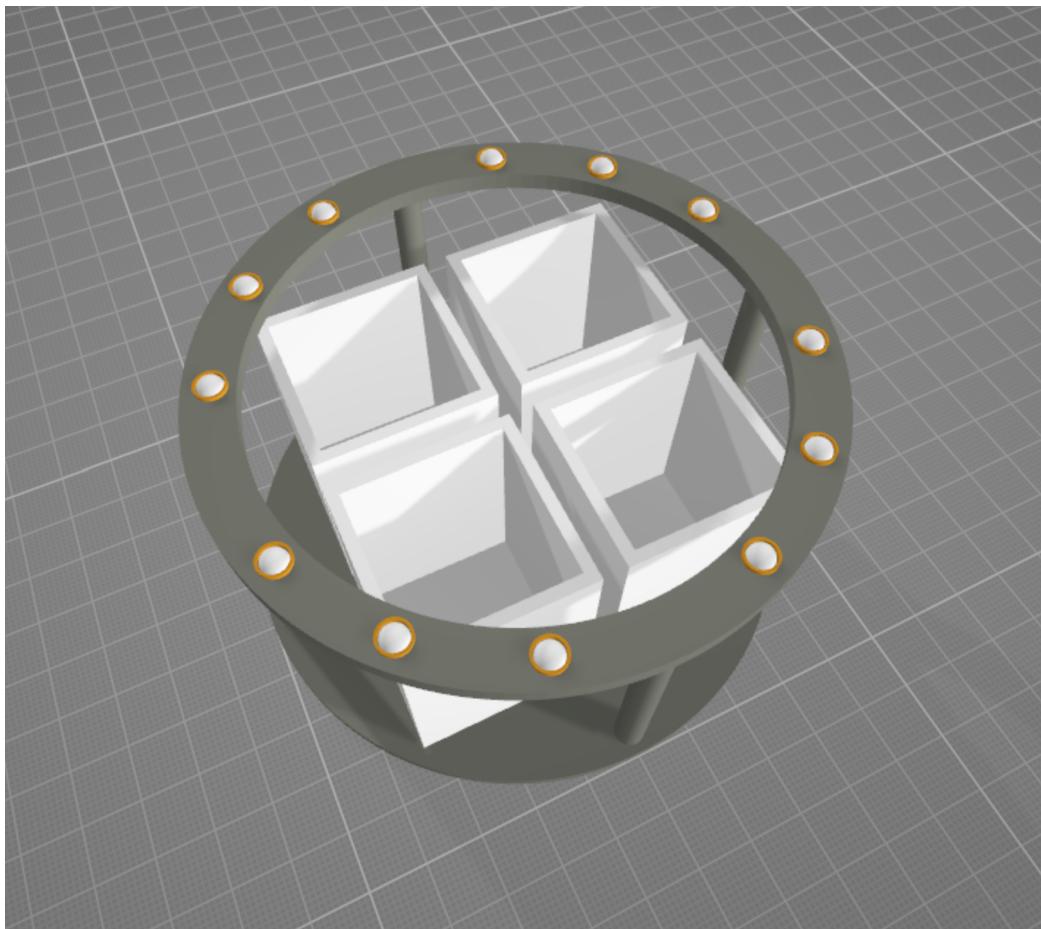


Figura 1.2: Modello 3D della struttura inferiore.

In fase di costruzione, Figura 1.3, si è deciso di diminuire il numero di ruote scorrevoli poste sopra in quanto alcune si reputavano superflue, al posto di 12 si è preferito collocarne 8. La base della struttura è stata costruita in legno, il resto della struttura in metallo.

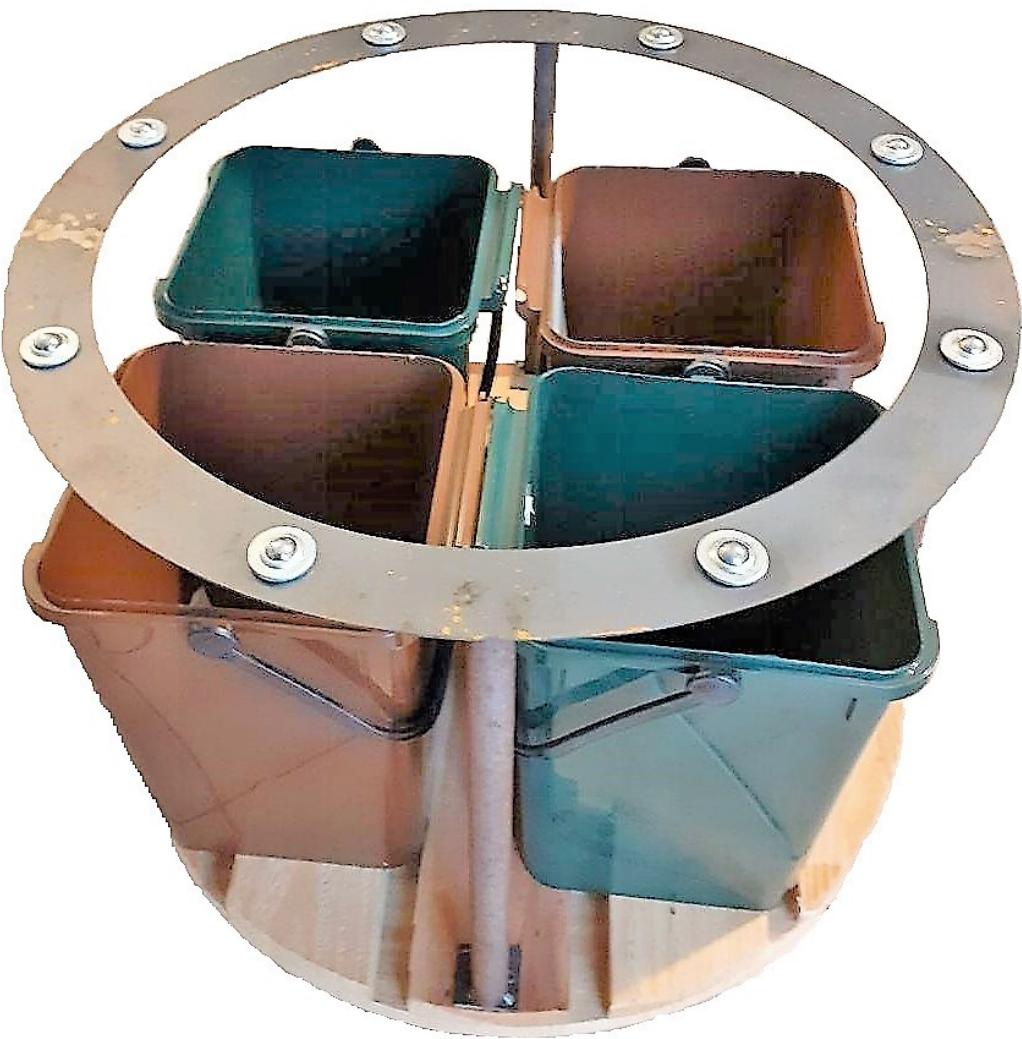


Figura 1.3: Foto della struttura inferiore

La struttura superiore costituisce la base sopra la quale vengono poste tutte le componenti di robotica, essa è costituita da una base circolare 60x60x1 cm sopra alla quale è agganciato un cestino 10x15x25 cm con all'interno una base mobile delle dimensioni interne del cestino che permette di far cadere i rifiuti nell'apposito comparto. Sotto la base sono posizionate su, un triangolo equilatero inscritto nella circonferenza posta sugli estremi della base,

tre ruote, due di queste girano a vuoto per centrare la base sulla struttura inferiore mentre una è robotizzata e consente all’impianto di ruotare.

Come si può osservare in Figura 1.4 e in Figura 1.5 la base sopra è divisa in tre sezioni: la parte con le componenti di robotica che inviano e ricevono segnali, la parte dove è posto un mini pc Beelink AP34 (con processore Intel Celeron N3450 e 4 GB di memoria RAM DDR3 SDRAM) sul quale viene installato il software in grado di ricevere e inviare input e output al sistema e infine una terza sezione dove è posizionato un monitor 3,5 pollici touch screen che permette all’utente di interfacciarsi con il sistema attraverso il programma.

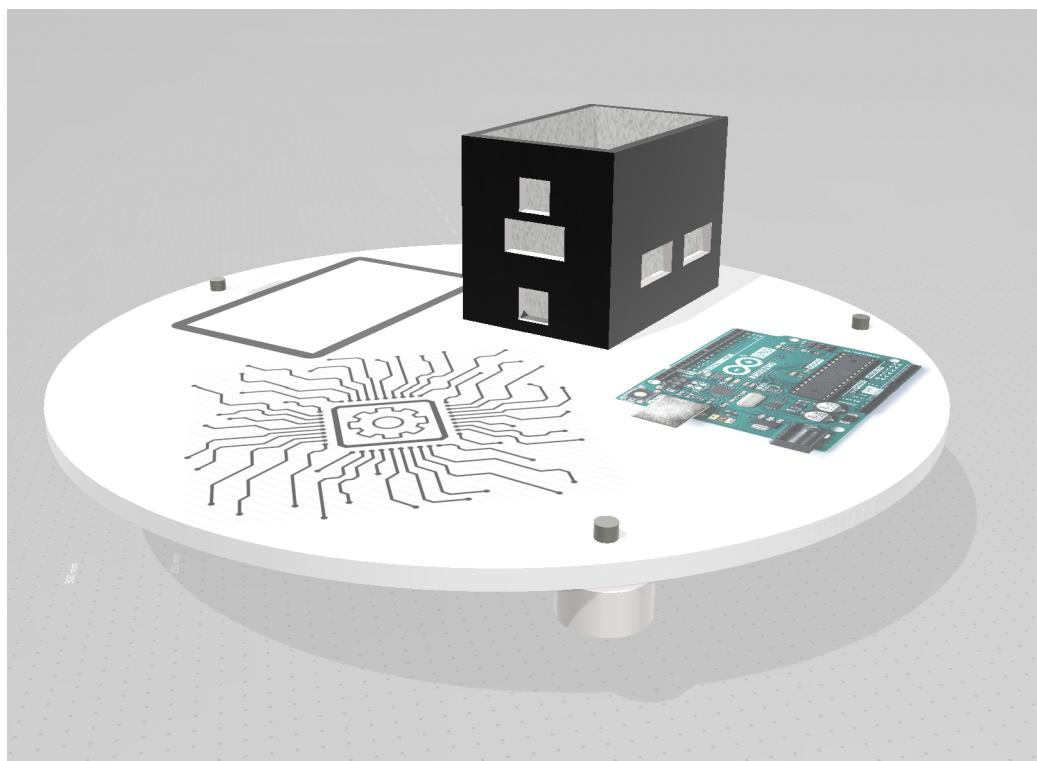


Figura 1.4: Modello 3D della struttura superiore

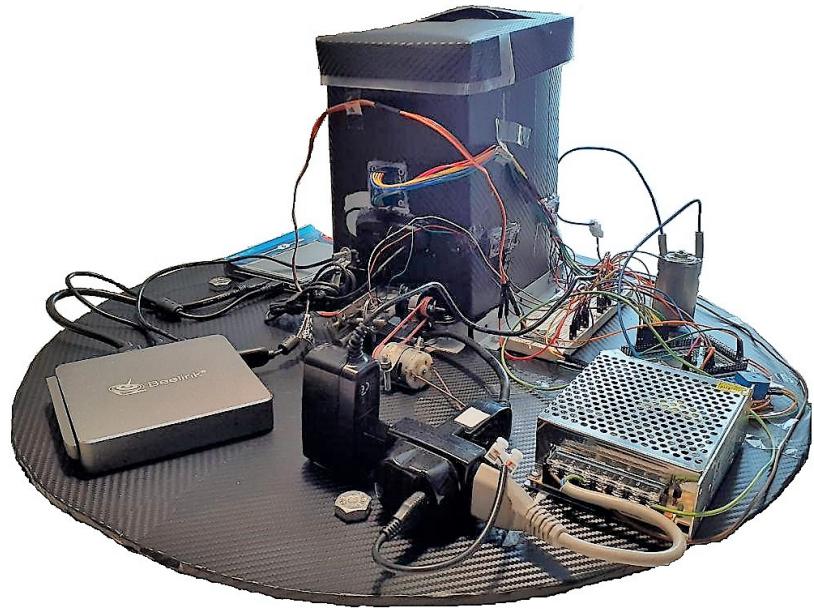


Figura 1.5: Foto della struttura superiore

1.5.2 Robotica

Arduino unisce due mondi: quello hardware [4] [5], rappresentato dalla scheda e dai componenti ad essa collegabili e quello software, rappresentato dal programma scritto e caricato all'interno della componente fisica. La scheda che si è deciso di utilizzare è la Arduino Mega 2560; come si può osservare in Figura 1.6, essa è composta da un microcontrollore ATmega2560 [6], di 54 pin digitali e 16 analogici, 4 porte seriali UART, un cristallo oscillatore a 16 MHz, una porta USB e un jack di alimentazione, un header ICSP e un pulsante di reset. Ha tre tipi di memoria: Flash, SRAM ed EPROM. La scheda lavora ad una tensione nominale di 5V e sopporta una corrente massima di 40 mA.

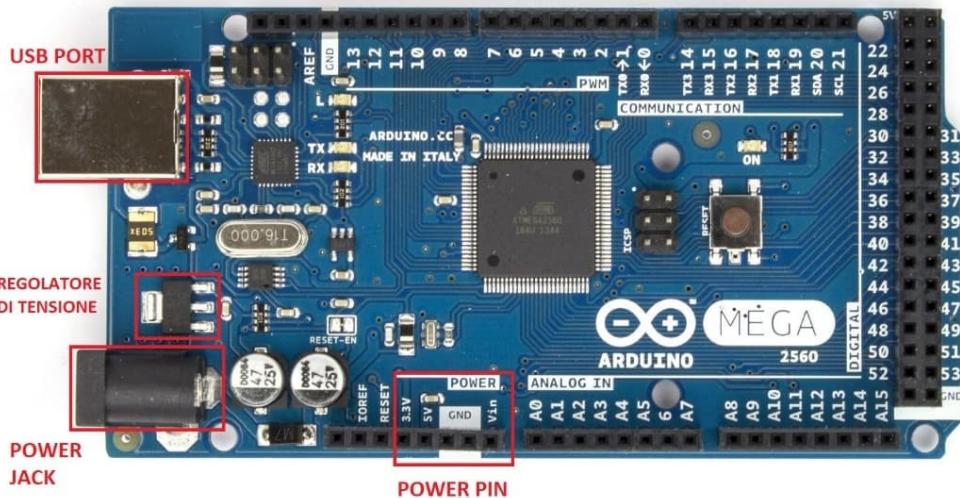


Figura 1.6: Scheda Arduino Mega 2560.

I componenti collegabili ad Arduino possono essere classificati in quattro macrocategorie [7]:

- I sensori: componenti elettronici in grado di percepire e misurare le caratteristiche fisiche dell’ambiente circostante e quindi, ad esempio, luminosità, temperatura, umidità, suono, movimento, campo magnetico ed elettromagnetico.
- Gli attuatori: componenti in grado di modificare le caratteristiche fisiche dell’ambiente circostante e quindi essenzialmente, sorgenti di luce, calore, umidità, suono, movimento, campo magnetico ed elettromagnetico.

- I componenti complessi: circuiti, dotati di microprocessore e componenti, in grado di fornire un servizio e quindi da fungere contemporaneamente, da sensori ed attuatori, come, ad esempio, i sistemi di gestione delle connessioni bluetooth, che possono ricevere ed inviare informazioni ad arduino, oppure la stazione di lettura/scrittura di secure digital.
- I componenti di supporto: aiutano l'operatività di sensori ed attuatori. Tra i componenti di supporto più comuni ci sono le resistenze, i condensatori, i fusibili ed altri ancora come la breadboard, gli shield, i pulsanti ed i cavi di collegamento.

In questa sezione non si andranno a descrivere tutti gli elementi inseriti all'interno del cestino, ma si andranno ad approfondire i quattro componenti usati per ricavare le informazioni dai rifiuti inseriti nel cestino.

Una fotoresistenza, Figura 1.7, è una resistenza la cui impedenza, ossia la cui capacità di far circolare elettricità, varia al variare della luce che la colpisce. All'aumentare della luce diminuisce la resistenza, e viceversa. Tipicamente è un sensore di tipo analogico. Per utilizzarlo si collega una gamba ad una porta analogica e, in parallelo, ad una resistenza da 10k ohm collegata a terra mentre si collega l'altra gamba all'alimentazione da 5 volt. La porta analogica restituisce un valore da 0 a 1023 che varia al variare della luce che colpisce la fotoresistenza. Più la luce è forte, più il valore si avvicina a 1023



Figura 1.7: Fotoresistenza

Script di programmazione 1.6: Fotoresistenza

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(A2, INPUT);  
    pinMode(A3, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    luce1=analogRead(A2);  
    luce2=analogRead(A3);  
  
    Serial.print(luce1);  
    Serial.print(",");  
    Serial.print(luce2);  
    Serial.println(";");
}
```

Il modulo ad ultrasuoni HC-SR04, Figura 1.8, viene normalmente utilizzato per rilevare eventuali ostacoli e misurarne la distanza (da 2 a 400 cm). Il modulo opera usando la medesima tecnica di rilevamento utilizzata, in natura, dai pipistrelli, è formato da un generatore di ultrasuoni, da un ricevitore e da un circuito di controllo. Il modulo si avvia quando riceve un impulso di almeno 10 microsecondi attraverso il “trig pin”, ossia la porta di attivazione. A questo punto lancia una serie di otto onde sonore da 40 kHz e si mette in attesa di un segnale di ritorno. Appena lo riceve attiva la porta di uscita (echo pin) e la mantiene attiva per un tempo proporzionale al tempo intercorso tra l’invio del segnale acustico ed il suo ritorno Conoscendo la velocità del suono e sapendo che il “viaggio” dell’onda sonora è il doppio della distanza tra il modulo e l’ostacolo (l’onda è andata dal generatore all’ostacolo e da qui’è tornata al sensore) la distanza è derivata dalla seguente formula:

$$distanza = tempo \frac{340}{2},$$

dove: distanza sono i metri tra il modulo HC-SR04 e l’ostacolo, tempo sono i secondi di attivazione della porta di uscita , 340 è la velocità del suono in metri al secondo.

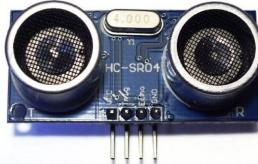


Figura 1.8: Il modulo ad ultrasuoni HC-SR04

Script di programmazione 1.7: HC-SR04

```
#include <NewPing.h>

NewPing sonar1(10, 9, 200);
NewPing sonar2(12, 11, 200);

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
}

void loop() {
    // put your main code here, to run repeatedly:

    unsigned int distanza1 = sonar1.ping();
    distanza1=distanza1 / US_ROUNDTRIP_CM;

    unsigned int distanza2 = sonar2.ping();
    distanza2=distanza2 / US_ROUNDTRIP_CM;

    Serial.print(distanza1);
    Serial.print(",");
    Serial.print(distanza1);
    Serial.println(";");
}
```

I sensori di peso, Figura 1.9, sfruttano la variazione di resistenza elettrica che alcuni materiali manifestano quando sono sottoposti a compressione o a trazione. Un sensore di peso ha la forma di una barra di metallo caratterizzata da due grandi fori aventi lo scopo di facilitarne la flessione nel momento in cui, su uno dei due estremi, viene esercitata una forza. Gli elementi che

permettono il funzionamento di questa tecnologia sono gli estensimetri che, opportunamente posizionati sulla barra forata forniscono indicazioni sufficienti a dimensionare la sollecitazione. L'estensimetro elettrico a resistenza è costituito da una griglia di sottilissimo filo metallico rigidamente applicata su di un supporto di materiale plastico. L'estensimetro viene utilizzato incollandolo sulla superficie del corpo di cui si vogliono misurare le deformazioni. Il filo segue le deformazioni della superficie a cui è incollato, allungandosi ed accorciandosi insieme ad essa; queste variazioni dimensionali causano una variazione della resistenza elettrica del filo. Misurando tali variazioni, si può risalire all'entità della deformazione che le ha causate. La variazione di resistenza, interpretata da un apposito driver **la scheda HX711**, consente ad Arduino di formulare precise indicazioni sulla sollecitazione cui la barra è sottoposta. Esistono diversi sensori di peso che hanno portate sensibilmente differenti, in questo progetto è stato utilizzato un sensore la cui portata massima è 20kg. Con questo componente utilizzeremo una libreria che restituisce il peso in Ounce quindi per trasformarlo in grammi occorre moltiplicare il valore restituito per il cambio ossia 28.3495231.

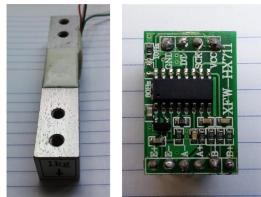


Figura 1.9: A sinistra sensore del peso a destra scheda HX711

Script di programmazione 1.8: HX711

```
#include "HX711.h"
// HX711.DOUT - pin #A4
// HX711.PD_SCK - pin #A5
HX711 scale(A4, A5);

void setup() {
    // put your setup code here, to run once:
    // set HX711
    scale.set_scale(2280.f);
```

```

    scale.tare();
    Serial.begin(9600);
}

void loop() {
    // put your main code here, to run repeatedly:
    Serial.print(scale.get_units(10)*28.3495231, 1);
}

```

Il sensore TCS3200, Figura 1.10, è un componente utilizzato allo scopo di individuare quali colori sono presenti nel suo campo visivo. Si tratta di un sensore composto da una matrice di 64 elementi fotosensibili di cui 16 con un filtro rosso 16 con un filtro verde e 16 con un filtro blu, più altri 16 non filtrati. Questo sensore all'uscita produce una frequenza che è funzione della luce che lo colpisce. E' possibile selezionare l'uscita legandola alla misura di uno dei 4 gruppi di sensori descritti. Arduino seleziona il colore da misurare e legge la frequenza in uscita dal modulo TCS230. Si è usato una funzione per leggere i colori chiamata “readColor()”.



Figura 1.10: ColorSensor TCS3200

Script di programmazione 1.9: TCS3200

```

//Read-Color Function
int readColor() {
    //Rosso
    digitalWrite(s2, LOW);
    digitalWrite(s3, LOW);

    //Lettura della frequenza degli output
    frequency = pulseIn(out, LOW);
    int R = frequency;

```

```

//scrivo sul monitor seriale i diversi valori
Serial.print(frequency); //frequenza del colore rosso
Serial.print(",");
//Verde della frequenza degli output
digitalWrite(s2, HIGH);
digitalWrite(s3, HIGH);

//lettura
frequency = pulseIn(out, LOW);

//scrivo sul monitor seriale i diversi valori
Serial.print(frequency); //frequenza del colore rosso
Serial.print(",");
//Blu
digitalWrite(s2, LOW);
digitalWrite(s3, HIGH);

//lettura della frequenza degli output
frequency = pulseIn(out, LOW);
int B = frequency;

//scrivo sul monitor seriale i diversi valori
Serial.print(frequency); //frequenza del colore rosso
Serial.print(",");
if(R<260 & R>230 & G<860 & G>800){
    color = 1; // Rosso
}
if(G<420 & G>370 & B<350 & B>305){
    color = 2; // Blu
}
if(R<450 & R>420 & G<420 & G>390){
    color = 3; // Verde
}
return color;
}

```

Lo schema di collegamento del cestino finale è visualizzabile in Figura 1.11, in aggiunta a questo sistema è collegato un motore DC 12 V controllato con un pin digitale e collegato ad un Power Supply 12V 10mA installato per fare ruotare la struttura superiore

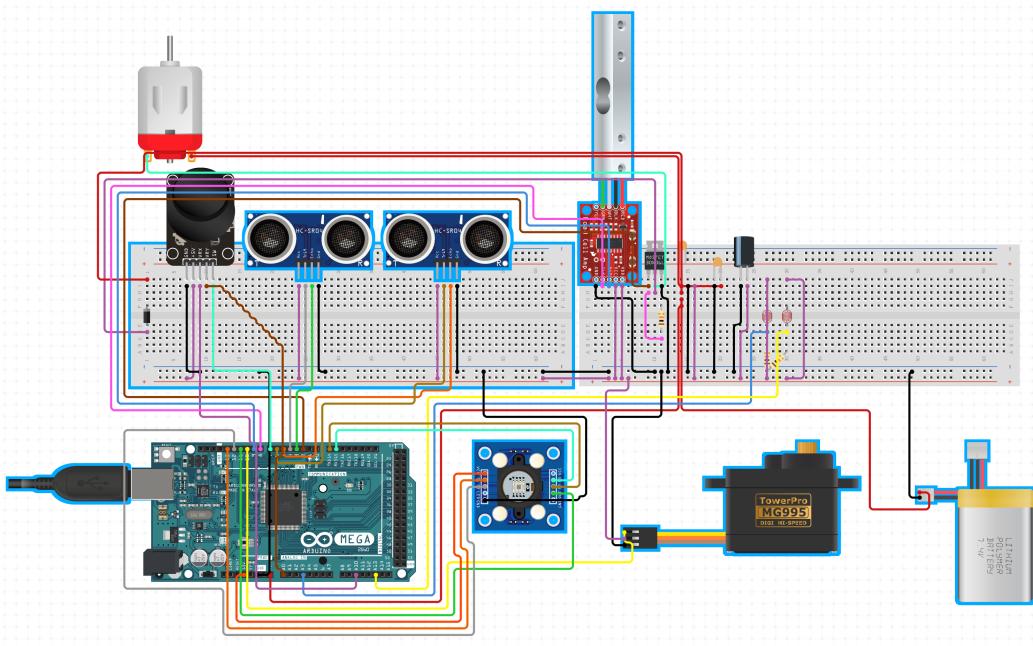


Figura 1.11: Schema di collegamento.

1.5.3 Softwaristica

In questo capitolo non si andranno ad analizzare i codici nello specifico in quanto molto lunghi e poco utili ai fini della tesi lasciando gli script commentati nell'Appendice A sezione [A.1.4] e [A.3.1], ma si spiegheranno le funzioni che il software deve svolgere.

Il programma si è deciso di chiamarlo “Recycling bin” ed è strutturato in 3 pagine. La prima pagina denominata “Form1:Home”, grazie a due pulsanti posizionati nella parte sinistra della schermate, serve per indirizzare l’utente verso la funzione di addestramento e reperimento dei dati o la sezione di riconoscimento e analisi dei dati, che si vedrà in modo approfondito nel capitolo

3, si è deciso inoltre di aggiungere, nella parte di destra, delle istruzioni così da rendere più chiaro l'utilizzo del software e lo scopo del progetto.

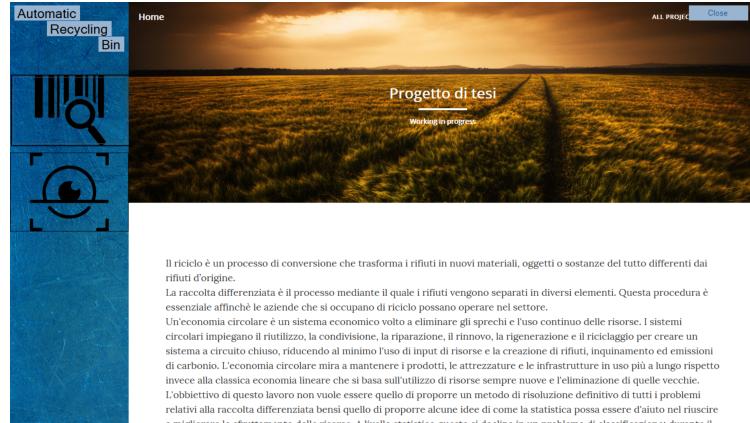


Figura 1.12: Form1: Home.

La seconda pagina chiamata “Form2:Addestramento”, Figura 1.13, serve per registrare i dati relativi ai rifiuti. Per inizializzare il programma occorre prima collegarlo alla telecamera del cestino una Logitech C270 Webcam HD scegliendo le specifiche grafiche utilizzando l'apposita finestra per la selezione della camera, Figura 1.14, e poi selezionare la porta COM relativa all'Arduino, operando antecedentemente una scansione delle porte così da poter leggere ed inviare i dati.

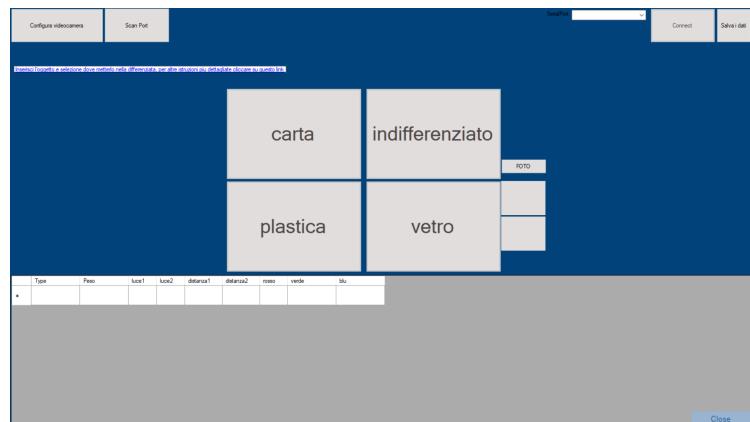


Figura 1.13: Form2: Addestramento.

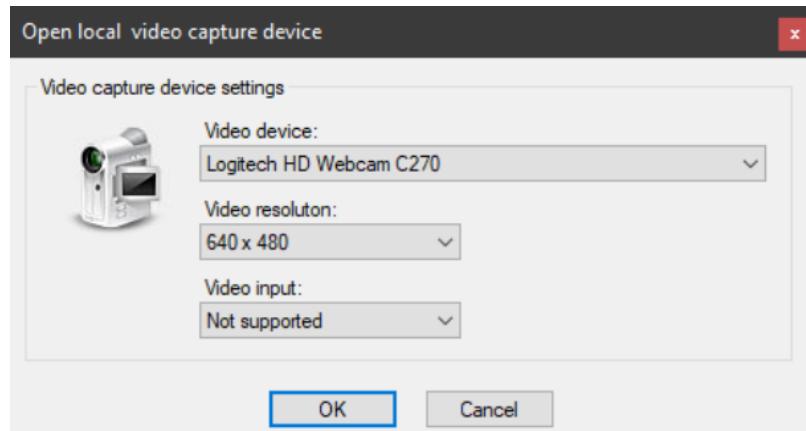


Figura 1.14: Form di inizializzazione Webcam.

Affinché il software possa leggere i dati è importante inizializzarlo collegandolo ai sensori del cestino con la procedura vista in precedenza. Una volta che il software è collegato occorre inserire un rifiuto nel cestino, cliccare tramite la pagina in Figura 1.12 la categoria di rifiuto che si è inserito e il software sarà in grado di leggere i dati che vengono scritti dall'Arduino, programmato in precedenza, sulla porta seriale, e successivamente di scattare una foto al rifiuto che verrà salvata nella directory denominandola con la categoria del rifiuto, giorno e ora di registrazione.

Capitolo 2

Modellizzazione e addestramento

2.1 Descrizione tipologie di dati

I dati raccolti in questo progetto sono sia strutturati che non strutturati, prima di analizzare singolarmente ogni componente del set occorre definire cosa si intende con le due tipologie.

I dati strutturati sono spesso considerati dati tradizionali, costituiti principalmente da file di testo che includono informazioni molto ben organizzate; questi sono conformi a un formato tabulare con relazione tra le diverse righe e colonne (il formato tabulare non deve essere per forza in due dimensioni, ma può essere anche in più di 2 dimensioni e si parla, in questi casi, di dati semi-strutturati). Esempi comuni di dati strutturati sono i file di Excel o i database SQL.

I dati non strutturati sono informazioni che non dispongono di un modello di dati predefinito questo però non implica che al loro interno non ci sia una organizzazione bensì che essa non è formalmente definita, ciò si traduce in irregolarità e ambiguità che rendono difficile la comprensione utilizzando programmi tradizionali rispetto ai dati memorizzati in database strutturati. Esempi comuni di dati non strutturati includono elementi multimediali (file audio, video, immagini) testi (libri, riviste, social media, email, messaggi).

In questo progetto come dati strutturati si sono raccolte le informazioni relative ai quattro sensori spiegati nel capitolo precedente. Grazie alle due fotoresistenze si è stati in grado di raccogliere le informazioni relative

alla quantità di luce che attraversa l'oggetto. I sensori sono stati posizionati in due punti contrapposti tali da ricevere la luce da angoli differenti. I due moduli ad ultrasuoni sono stati posizionati in modo tale da ricevere e trasmettere il segnale in tutta l'area oggetto dell'indagine per tentare di misurare le differenti modalità con cui cadono i rifiuti nel cestino. Considerando che questi due sensori misurano un'onda presa in un determinato momento è stato necessario calcolarne la media di più rilevazioni per ogni rifiuto in entrambi i moduli. Il sensore HX711 restituisce come informazione il peso del rifiuto. Il sensore TCS3200, infine, è un componente utilizzato allo scopo di individuare quali colori sono presenti nel suo campo visivo e restituisce tre variabili che indicano la quantità registrata di colore "rosso", "blu" e "verde". Di seguito, in Tabella 2.1, si è svolta un'analisi descrittiva calcolando la Media e l'intervallo di confidenza (CI) sfruttando il teorema centrale del limite, di ogni variabile per ogni categoria di rifiuto:

Tabella 2.1: Tabella statistiche descrittive Automatic Recycling Bin.

Categoria	Plastica	Carta	Indifferenziata	Vetro
Numero	500	500	500	500
Peso - media	50.71	23.02	45.29	127.15
Peso - CI	56 - 46	26 - 20	58 - 33	139 - 116
luce1 - media	83.86	87.66	91.44	86.4
luce1 - CI	85 - 82	89 - 86	93 - 90	87 - 85
luce2 - media	88.58	88.6	92.27	89.88
luce2 - CI	91 - 87	90 - 87	93 - 91	91 - 89
ultrasuoni1 - media	8.92	9.43	10.02	6.2
ultrasuoni1 - CI	10 - 8	10 - 9	11 - 9	7 - 6
ultrasuoni2 - media	20	19.93	20.07	19.94
ultrasuoni2 - CI	20 - 20	20 - 20	20 - 20	20 - 20
rosso - media	2676.2	3434.24	4142.35	3456.68
rosso - CI	2847 - 2505	3595 - 3274	4287 - 3998	3555 - 3359
verde - media	2337.96	3064.44	3600.75	2961.74
verde - CI	2477 - 2199	3212 - 2917	3721 - 3480	3044 - 2879
blu - media	1756.63	2339.18	2703	2221.66
blu - CI	1849 - 1664	2450 - 2229	2793 - 2613	2284 - 2160

Per rendere ancora più chiare le dinamiche tra le variabili che intervengono nello studio e le differenti categorie di rifiuto si è deciso di fare un'analisi di tipo esplorativa andando a visualizzare, in Figura 2.1, correlazione, istogramma e scatter plot con trend calcolato con metodo LOESS ed, in Figura 2.2, uno grafico di dispersione associando ad ogni classidi rifiuto un colore differente.

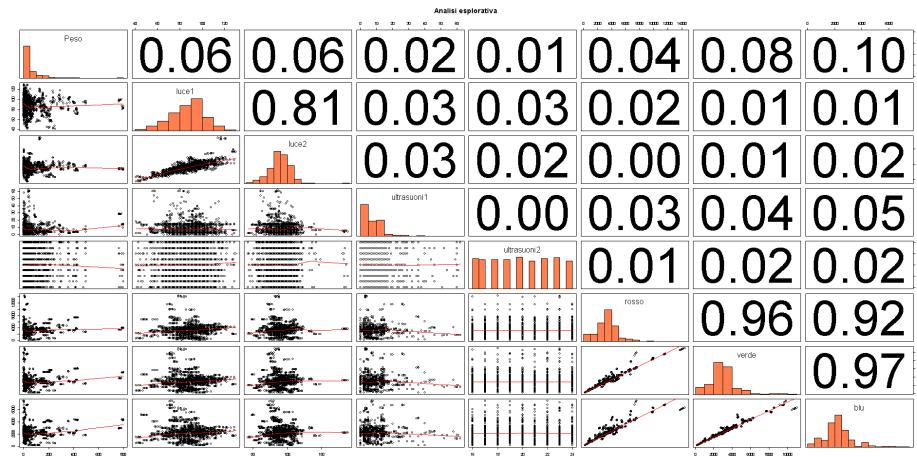


Figura 2.1: Analisi Esplorativa: correlazione, istogramma e scatterplot con trend.

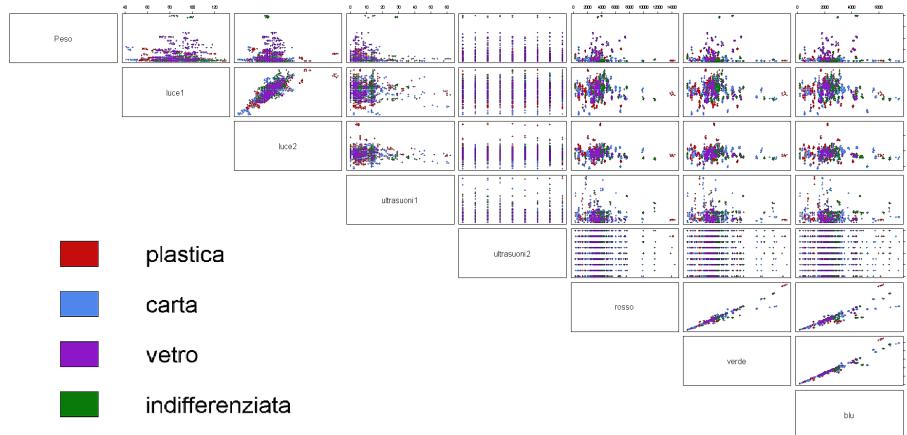


Figura 2.2: Analisi Esplorativa: Scatterplot per classi.

Per quanto riguarda i dati non strutturati si sono raccolte le immagini relative alle differenti tipologie di rifiuti con i tre metodi esposti in precedenza.

Con la prima metodologia, il Web Scraping, si sono raccolte 41042 immagini divise in 129 categorie che per semplicità si sono raggruppate in 10, Tabella 2.2. Sul Chatbot di Telegram sono state inviate 1348 foto relative alle tre categorie principali di rifiuti, Tabella 2.3. Infine, attraverso il cestino, è stato possibile raccogliere, oltre ai dati strutturati spiegati in precedenza anche 2000 immagini, Tabella 2.4.

Immagini raccolte per le differenti tipologie di raccolta:

Tabella 2.2: Immagini Web Scraping.

Numero	Descrizione	Quantità
1	Plastica	5082
2	Carta	5086
3	Tetra Pak	1282
4	Polistirolo	940
5	Vetro	4811
6	Indifferenziato	4705
7	Legno	4799
8	Metalli	4806
9	Umido	4751
10	Latta	4780

Tabella 2.3: Immagini Chatbot di Telegram.

Numero	Descrizione	Quantità
1	Plastica	464
2	Carta	493
5	Vetro	391

Tabella 2.4: Immagini Automatic Recycling Bin.

Numero	Descrizione	Quantità
1	Plastica	500
2	Carta	500
5	Vetro	500
5	Indifferenziata	500

Le immagini digitali monocromatiche sono rappresentate come matrici bidimensionali (2D) le quali rappresentano una misura opportuna di una o più caratteristiche (luminosità, colore, ecc.) di una data scena. Ogni elemento della matrice corrisponde ad un singolo pixel nell’immagine visualizzata. Ad esempio, un’immagine con risoluzione 200 x 200 pixel, come quella visualizzata in Figura 2.3, sarà una matrice composta da 200 righe e 200 colonne.

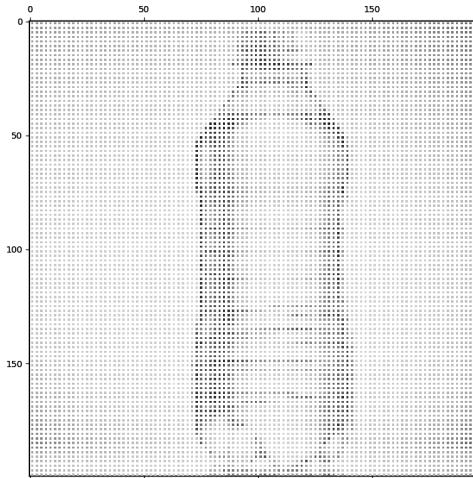


Figura 2.3: Immagine divisa in matrice

Le immagini presenti in questo progetto sono RGB e richiedono una matrice tridimensionale, in cui il primo piano della terza dimensione rappresenta le intensità dei pixel rossi, il secondo piano rappresenta le intensità dei pixel verdi e il terzo piano rappresenta le intensità dei pixel blu. Questa conven-

zione, specificata in Figura 2.4, rende l'utilizzo di immagini in formato di file grafico simile all'utilizzo di qualsiasi altro tipo di dati della matrice.

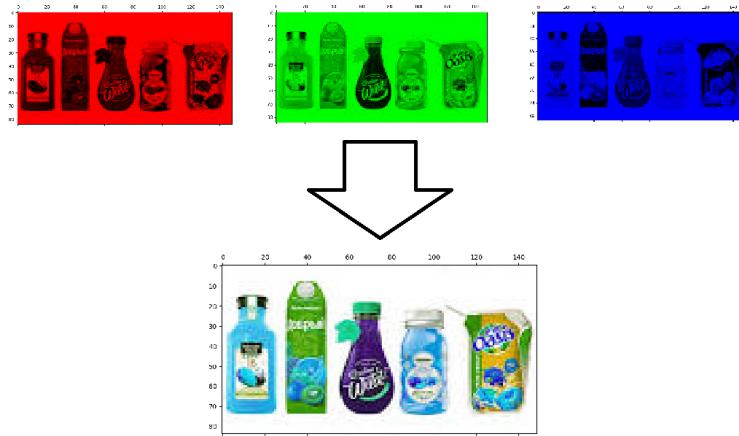


Figura 2.4: Immagine RGB

In questa parte del lavoro si utilizzano tecniche di validazione interna come cross validation e divisione in test e train. Si è deciso, poi, di creare un ulteriore validation set che contenesse sia immagini che dati strutturati relativi al rifiuto per estendere l'analisi al confronto di modelli. Di seguito, in Tabella 2.5 e in Figura 2.5, si è svolta un'analisi descrittiva ed esplorativa sui dati:

Tabella 2.5: Validation set.

Numero	Descrizione	Quantità
1	Plastica	60
2	Carta	60
5	Vetro	60
5	Indifferenziata	60

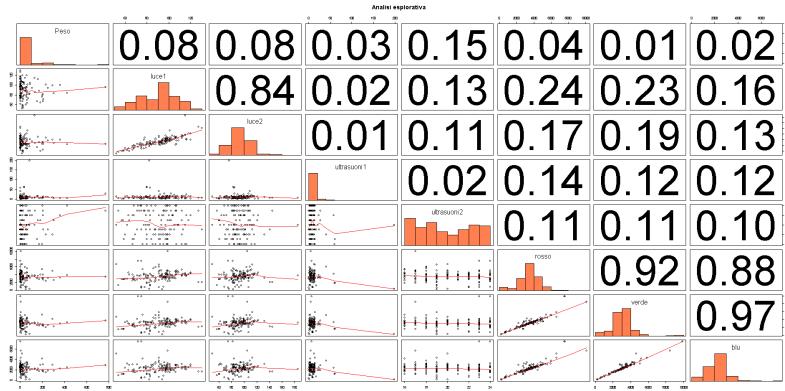


Figura 2.5: Analisi Esplorativa: correlazione, istogramma e scatterplot con trend calcolato con losess

2.2 Dati Strutturati

2.2.1 Gradient boosting algorithm (GBM)

Il Boosting fu ideato da Michael Kearns e Leslie Gabriel Valiant nel 1988 e combina componenti di costruzione semplici chiamati weak learner per creare un classificatore più forte.

Anche se, in genere, il Boosting è implementato utilizzando dei modelli ad albero decisionale, viene definito un meta-algoritmo in quanto lascia ampio margine di manovra per quanto riguarda la natura dei modelli da utilizzare; infatti, come weak learner potrebbe potenzialmente essere utilizzato qualsiasi modello previsivo.

L'albero decisionale [8] (o Decision Tree) è una tecnica di classificazione o di regressione strutturata ad albero.

Gli algoritmi di Decision Tree comunemente implementati sono: Chi-squared Automatic Interaction Detection (CHAID), Classification and Regression Trees (CART), C4.5 e C5.0. Tra questi, l'albero convenzionalmente utilizzato per operazioni di boosting è il CART per le sua struttura semplice.

L'idea fondamentale alla base di questi algoritmi consiste nel cercare di verificare per gli attributi se è possibile distinguere delle dinamiche ricorrenti nel dataset attraverso misure di diversità. Il concetto di diversità indica quanto è disomogenea una popolazione e quindi la classe prevista per un'unità è la classe prevalente nella regione.

In letteratura, le funzioni di impurità comunemente usate per individuare la purezza di una regione R_k sono: indice di Gini; errore di scorretta classificazione; entropia.

Assumendo come misura di diversità l'indice di Gini ($G()$) e che la variabile di risposta abbia due livelli $Y = \{\text{green}, \text{blue}\}$, una divisione S_2 che divide lo spazio in S_{2a} e S_{2b} è da considerarsi migliore rispetto ad una divisione S_1 se $\Delta G(S_2) > \Delta G(S_1)$.

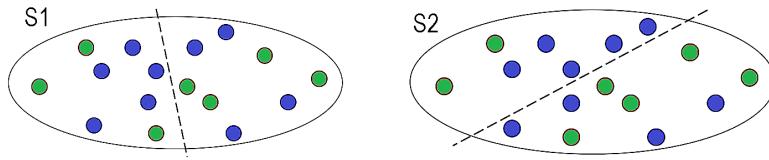


Figura 2.6: Esempio albero di classificazione

L'indice di Gini, è un indice che misura il grado di eterogeneità (omogeneità) in una distribuzione statistica di dati suddivisa in k categorie ciascuna delle quali ha frequenza relativa pari a f_i . La formula per il calcolo dell'indice di Gini assoluto è:

$$G = 1 - \sum_{i=1}^k f_i^2.$$

Maggiore è tale indice più i dati saranno distribuiti in maniera eterogenea mentre più è prossimo allo 0 più i dati tenderanno a distribuirsi in maniera non equa tra le k modalità.

Dall'esempio specificato in Figura 2.6 si ricava il seguente indice di Gini:

$$G_T = 1 - \left\{ \left(\frac{7}{16} \right)^2 + \left(\frac{9}{16} \right)^2 \right\} = 0.49$$

Per S_1 :

$$G(S_1) = \frac{8}{16} \left\{ 1 - \left[\left(\frac{5}{8} \right)^2 + \left(\frac{3}{8} \right)^2 \right] \right\} + \frac{8}{16} \left\{ 1 - \left[\left(\frac{4}{8} \right)^2 + \left(\frac{4}{8} \right)^2 \right] \right\} = 0.48$$

Per S_2 :

$$G(S_2) = \frac{7}{16} \left\{ 1 - \left[\left(\frac{5}{7} \right)^2 + \left(\frac{2}{7} \right)^2 \right] \right\} + \frac{9}{16} \left\{ 1 - \left[\left(\frac{4}{9} \right)^2 + \left(\frac{5}{9} \right)^2 \right] \right\} = 0.45$$

Quindi:

$$\Delta G(S_1) = 0.49 - 0.48 = 0.01$$

$$\Delta G(S_2) = 0.49 - 0.45 = 0.04$$

$\Delta G(S_2) > \Delta G(S_1) \Rightarrow$ Quindi si sceglierà S2 come partizione.

Un albero partiziona ricorsivamente i dati del training set usando di volta in volta l'attributo che crea il massimo scarto nell'indice di diversità finché tutti i dati di una partizione apparterranno ad una classe (gruppo omogeneo). La procedura di partizionamento si arresta quando non si identifica alcuna segmentazione che possa ridurre in maniera significativa la diversità di un dato nodo.

Il Gradient Boosting [9] è una implementazione dell'algoritmo di Boosting, dal quale eredita la logica di costruzione di uno Strong Learner. I modelli che vengono utilizzati sono, generalmente, alberi decisionali e il suo scopo è quello di minimizzare una generica funzione di costo.

Questo algoritmo inizia ad addestrare un weak learner sul training data e successivamente questo viene nuovamente adattato dando un peso (importanza) maggiore alle osservazioni classificate in modo errato grazie ad un parametro di restringimento. Questo processo viene ripetuto finché non viene raggiunta una regola di arresto.

Il GBM costruisce con gli alberi di classificazione dei modelli adattando ripetutamente questi ai residui, infatti in questo algoritmo ogni albero cerca di correggere gli errori commessi dall'insieme di alberi precedentemente addestrati.

1. Si inizializza un albero che non esegue divisioni e che per ogni classe restituisce una probabilità associata pari a zero $\hat{f}^0(x) = 0$, si decide un $B \gg 0$ e il parametro di restringimento (shrinkage parameter) $\lambda > 0$ in modo arbitrario;
2. Si reiterano i seguenti punti per $b = 1, 2, \dots, B$ volte:

- (a) Si calcola il gradiente negativo della funzione di perdita (pointwise negative gradient of the loss function) per ogni punti stimati (fitted value)

$$r_i = -\frac{\partial L(y_i, f_i)}{\partial f_i} \Big|_{f_i=\hat{f}^{b-1}(x_i)}, \quad i = 1, \dots, n$$

- (b) Si approssimano valori attraverso la stima di un albero g con una profondità d :

$$(x_1, r_1), \dots, (x_n, r_n) \rightarrow \hat{g}(x)$$

- (c) Si aggiornano i dati del primo step e si reitera il procedimento

$$\hat{f}^b(x) = \hat{f}^{b-1}(x) + \lambda \hat{g}(x)$$

3. L'output del modello è: $\hat{f}^b(x), b = 1, \dots, B$

Come tutti gli algoritmi di Machine Learning anche il Gradient Boosting è soggetto ad un fenomeno conosciuto come overfitting. L'overfitting è la tendenza che ha un algoritmo a “memorizzare” gli esempi del training set, diventando sempre più accurato nel predirne i valori, ma rendendo il previsore incapace di generalizzare al di fuori del dataset. Risulta chiaro che questo fenomeno sia da evitare dato che lo scopo finale di un modello è proprio quello di fare previsioni partendo da valori di input che siano diversi da quelli di training.

Parametri di tuning per il boosting:

- Il numero di alberi (B): il numero di estimatori è fondamentale in quanto indica il numero di alberi da costruire. Un maggior numero di alberi aiuta ad imparare meglio dai dati, d'altra parte, può comportare oltre a tempi computazionali più elevati anche il problema dell'overfitting.
- Il parametro di restrinzione (shrinkage parameter o learning rate) λ : questo parametro è positivo, prossimo allo zero e controlla il tasso con cui l'algoritmo apprende. La scelta di questo dipende dal problema, ma come linea guida si può generalizzare che ad un piccolo λ dovranno corrispondere un numero più elevato di alberi B per avere una buona performance.

- Il numero di split (max depth) d : questo parametro controlla la complessità della struttura di ogni singolo albero. Spesso $d = 1$ risulta sufficientemente accurato creando degli alberi con una sola divisione (chiamati stump), ma potrebbe essere necessario aumentare questo parametro per aumentare le performance del modello.

2.2.2 Multi-Layer Perceptron (MLP)

La rete neurale è un sistema di calcolo che nasce dall'idea di simulare artificialmente il comportamento del cervello umano, per sfruttarne le sue caratteristiche di:

- Complessità;
- Non-linearietà;
- Sistema a processamento parallelo (Parallel processing system).

Si può infatti definire, in maniera precisa, una rete neurale come un processore parallelo composto di singole unità di calcolo, dette neuroni, che possiede una naturale predisposizione a memorizzare le conoscenze sperimentalmente acquisite ed a renderle disponibili per l'uso. La sua somiglianza con il cervello umano si riferisce a:

- La conoscenza che è acquisita dalla rete mediante un processo di apprendimento.
- Le connessioni neuronali, ossia i pesi sinaptici, che sono utilizzati per memorizzare le informazioni acquisite.

L'uso delle reti, quindi, consente la possibilità di generalizzare la conoscenza acquisita durante la fase di apprendimento.

Il cervello umano ha come sua unità di processamento di informazioni il neurone. Il neurone artificiale è costituito da un'unità che riceve in ingresso un valore numerico che consiste in una somma ponderata da dei pesi w di diversi segnali, l'unità di processamento elabora gli input attivandosi oppure rimanendo inattivo a seconda che venga superata (o meno) la soglia di attivazione.

Più specificatamente, il neurone artificiale usa il dato in ingresso come argomento per una funzione, detta appunto di attivazione $s()$ e che restituisce

in uscita i valori 0, 1 oppure un valore compreso in $[0, 1]$. Pertanto, il neurone sarà caratterizzato da una funzione e dalla soglia di attivazione. Quest'ultima viene solitamente introdotta mediante un ingresso costante, uguale ad 1. Alla soglia viene aggiunto un coefficiente, che si indica con il nome di bias, il cui effetto è quello di controllare la traslazione della soglia di attivazione rispetto all'origine dei segnali. Formalmente, il bias ha un ruolo non diverso da quello dei pesi che funzionano da regolatori dell'intensità del segnale emesso (o ricevuto).

È d'uso adottare diverse funzioni di attivazione, a seconda del ruolo che il neurone e la rete neurale sono destinati a svolgere. Le più comuni funzioni di attivazione sono:

- Sigmoide

$$s(x) = \frac{1}{1 + e^{-x}}$$



Figura 2.7: Sigmoide

- Gradino

$$s(x) = \begin{cases} 0, & \text{se } x \leq a \\ 1, & \text{se } x > a \end{cases}$$



Figura 2.8: Gradino

- Tangente iperbolica

$$s(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

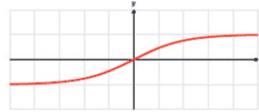


Figura 2.9: Tangente iperbolica

- Rampa

$$s(x) = \begin{cases} -1, & \text{se } x \leq -a \\ x, & \text{se } -a < x \leq a \\ 1, & \text{se } x > a \end{cases}$$

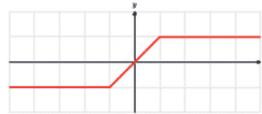


Figura 2.10: Rampa

Si ipotizza una rete neurale semplice, strutturata come segue:

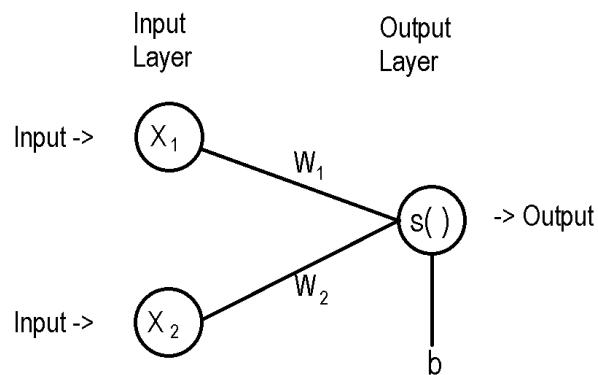


Figura 2.11: Funzionamento neurone artificiale

Per addestrare questa semplice rete in Figura 2.11 per un problema di classificazione binaria, ipotizzando che le due variabili di input siano X_1 e

X_2 e che l'output si una variabile dicotomica Y , occorre adottare due processi chiamati Back-propagation e Foward-propagation:

1. Scelgo un Learning Rate (lr) e un numero di replicazioni L (epoch);
2. Si inizializzano i valori w_1 , w_2 e $b = 1$ in modo arbitrario;
3. Si selezionano casualmente $x_1 \in X_1$ e $x_2 \in X_2$ (per semplicità esplicativa si ipotizza che nella rete passi un dato alla volta: batch size = 1; ma lo stesso ragionamento può essere esteso ad un vettore di informazioni);
4. Foward-propagation:
 - (a) Si procede alla stima di Y (o):

$$\eta = x_1 w_1 + x_2 w_2 + b$$

$$o = s(\eta)$$

- (b) Si calcola la funzione di perdita, solitamente si utilizza l'errore quadratico di previsione:

$$MSE = (y - o)^2 = (y - s(x_1 w_1 + x_2 w_2 + b))^2$$

5. Back-propagation:

- (a) Calcolare le derivate parziali del costo rispetto a w_1 , w_2 e b

* Derivata parziale della funzione di costo rispetto a w_1 :

$$\begin{aligned} \frac{\partial MSE}{\partial w_1} &= \frac{\partial MSE}{\partial o} \frac{\partial s(x_1 w_1 + x_2 w_2 + b)}{\partial (x_1 w_1 + x_2 w_2 + b)} \frac{\partial (x_1 w_1 + x_2 w_2 + b)}{\partial w_1} = \\ &= 2(y - o) \frac{\partial s(\eta)}{\partial \eta} x_1 \end{aligned}$$

* Derivata parziale della funzione di costo rispetto a w_2 :

$$\begin{aligned} \frac{\partial MSE}{\partial w_2} &= \frac{\partial MSE}{\partial o} \frac{\partial s(x_1 w_1 + x_2 w_2 + b)}{\partial (x_1 w_1 + x_2 w_2 + b)} \frac{\partial (x_1 w_1 + x_2 w_2 + b)}{\partial w_2} = \\ &= 2(y - o) \frac{\partial s(\eta)}{\partial \eta} x_2 \end{aligned}$$

* Derivata parziale della funzione di costo rispetto a b :

$$\begin{aligned}\frac{\partial MSE}{\partial b} &= \frac{\partial MSE}{\partial o} \frac{\partial s(x_1w_1 + x_2w_2 + b)}{\partial (x_1w_1 + x_2w_2 + b)} \frac{\partial (x_1w_1 + x_2w_2 + b)}{\partial b} = \\ &= 2(y - o) \frac{\partial s(o)}{\partial o}\end{aligned}$$

(b) Aggiornamento dei pesi e del bias

$$\begin{aligned}w_1 &= w_1 - lr \frac{\partial MSE}{\partial w_1} \\ w_2 &= w_2 - lr \frac{\partial MSE}{\partial w_2} \\ b &= b - lr \frac{\partial MSE}{\partial b}\end{aligned}$$

6. Reitero il processo L volte.

Nelle reti neurali, si propaga in avanti per ottenere l'output e confrontarlo con il valore reale ottenendo l'errore e successivamente, per ridurre al minimo la funzione di perdita. Si propaga all'indietro trovando la derivata dell'errore rispetto ad ogni peso ed infine sottraendo questo valore ponderato per il learning rate dal valore del peso.

L'algoritmo si arresta o per il raggiungimento dell'errore minimo accettabile o per il raggiungimento del numero massimo di *epoch* (L) stabilite in fase di progetto (per evitare che il processo di addestramento tenda all'infinito). Un'epoca è un ciclo di aggiornamento completo di tutti i pesi, dando in input l'intero set di training.

L'apprendimento profondo (in inglese deep learning) è quel campo di ricerca del machine learning che comprende l'insieme di tecniche basate su reti neurali artificiali organizzate in diversi strati, dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa.

Una Multi-Layers Perceptron net (MLP) combina diversi livelli di elaborazione, utilizzando neuroni artificiali. È formata, come si può osservare in Figura 2.12 da un layer di input, uno o più layers nascosti e un layer di output. I livelli sono interconnessi tramite nodi o neuroni, con ogni layer che utilizza l'output del layer precedente come input.

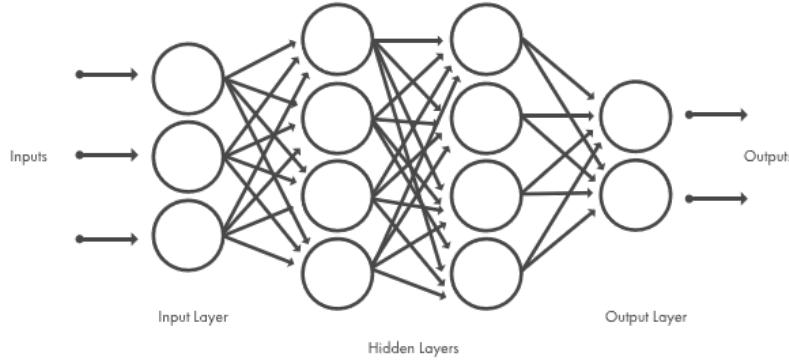


Figura 2.12: Multi-Layers Perceptron net (MLP)

La logica di funzionamento dietro questo metodo è quella spiegata antecedentemente nella costruzione di una rete semplice, ma al posto dei neuroni si utilizzano i percetroni (Perceptron). Il percettrone è un semplice neurone dotato del circuito “teacher” per l’apprendimento:

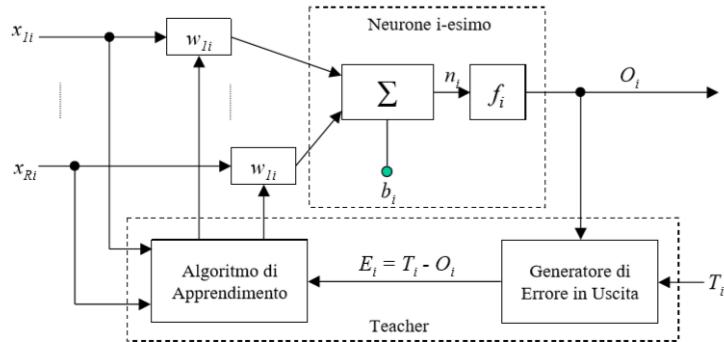


Figura 2.13: Struttura del percettrone

Spesso in questo tipo di reti viene utilizzata come funzione di attivazione la ReLu (Rectified Linear Units):

$$s(x) = \max(0, x)$$

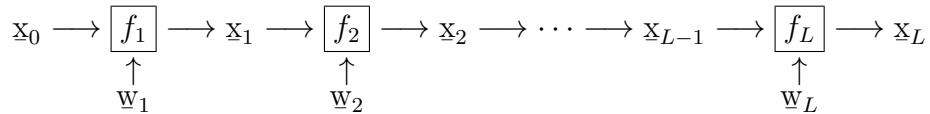
Può essere utilizzata dai neuroni come qualsiasi altra funzione di attivazione, il principale motivo per cui viene utilizzata è dovuto alla sua semplicità

di calcolo ed efficienza computazionale rispetto alle funzioni di attivazione più classiche come il sigmoide e la tangente iperbolica, senza fare una differenza significativa per l'accuratezza di generalizzazione. Questa funzione deriva dalla Rel quando $a_i = 0$:

$$Rel(x_i) = \begin{cases} x_i, & \text{se } x_i > 0 \\ a_i x_i, & \text{se } x_i \leq 0 \end{cases}$$

La ReLu viene utilizzata al posto della sua versione più generale per aggiungere la non linearità alla rete, altrimenti la rete sarebbe sempre in grado di calcolare solo una funzione lineare.

Infatti, una rete neurale feed-forward, può essere considerata come una composizione di funzioni e se tutte le funzioni risultano lineari allora anche la funzione composta sarà lineare.



$$f(x) = f_L(\dots(f_2(f_1(x; w_1), w_2), \dots), w_L)$$

2.2.3 Risultati

Prima di poter implementare i modelli è stato necessario eseguire alcune operazioni di preprocessing. Come si deduce dai grafici nella sezione precedente di analisi esplorativa (Figura 2.1 e Figura 2.2) la variabile “ultrasuoni2” non fornisce nessun elemento discriminante riguardo alla tipologia di rifiuto e considerando che potrebbe essere un elemento di disturbo nella creazione del modello si è deciso di eliminarla. Nella analisi esplorativa le registrazioni dei colori nel campo visivo risultano molto correlate tra loro, per evitare eventuali problemi di autocorrelazione e anche di perdita di informazioni si è deciso di fare la media tra questi valori piuttosto che eseguire una trasformazione in variabile qualitativa per fare diventare i tre colori un HEX code (codice ampiamente utilizzato in linguaggi grafici). Altre due variabili che risultano molto correlate sono le due fotoresistenze e per queste si è deciso di creare un’altra variabile chiamata “luminosità” che corrisponde alla media della luce che attraversa un l’oggetto.

Nel valutare i differenti modelli per tutto il lavoro utilizzeremo le metriche che normalmente vengono usate per la classificazion “multi-class”. In un problema di classificazione le previsioni possono essere rappresentate in una matrice di confusione nel seguente modo:

		CLASSE OSSERVATA				
		carta	indifferenziata	plastica	vetro	
CLASSE PREVISTA	carta	TP_{CARTA}	ERR, FN	ERR, FP	ERR, FP	$PRED_{CARTA}$
	indifferenziata	ERR, FN	TP_{INDIFF}	ERR, FP	ERR, FP	$PRED_{INDIFF}$
	plastica	ERR, FN	ERR, FN	TP_{PLAST}	ERR, FP	$PRED_{PLAST}$
	vetro	ERR, FN	ERR, FN	ERR, FN	TP_{VETRO}	$PRED_{VETRO}$
		$SUPPORT_c$	$SUPPORT_i$	$SUPPORT_p$	$SUPPORT_v$	$SUPPORT_{TOT}$

Figura 2.14: Matrice di confusione per problema di classificazione bivariato

Alcune misure di sintesi sono relative alle classi i-esime ($i = \{Carta, Indifferenziata, Plastica, Vetro\}$). La “Precision” corrisponde alla misura in valori percentuali di quanti dati l’algoritmo di classificazione riesce a prevedere sul totale di quelli che vengono osservati:

$$Precision = \frac{\text{True Positive}_i}{\text{Support}_i}$$

Il “Recall” è il valore percentuale di quante previsioni si rivelano giuste rispetto al totale:

$$Recall = \frac{\text{True Positive}_i}{\text{Prediction}_i}$$

La metrica “f1-score” è una misura di sintesi tra le due metriche precedenti ed indica quanto un modello previsivo riesca a prevedere in modo accurato una determinata classe

$$\text{f1-score}_i = 2 * \frac{\text{Precision}_i * \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

L’accuratezza corrisponde alla media degli “f1-score” e viene reputata una buona metrica per poter confrontare in termini generali due modelli. Per essere più precisi si sono aggiunte anche due medie delle misure singole per ogni classe di rifiuto, la prima è una media semplice mentre la seconda una media ponderata in base alla quantità di elementi della classe corrispondente della misura presenti nel campione.

Modello: GradientBoostingClassifier

Script per l'implementazione nell'Appendice A sezione [A.2.1]

Nello specificare il modello si sono dovuti scegliere i parametri di tuning visti in precedenza. Si è scelto come predittore l'albero di classificazione, un learning rate pari a 0.2, un numero di predittori pari a 500 ed una lunghezza massima di split pari a tre livelli.

Tabella 2.6: Confusion matrix GBM

	carta	indifferenziata	plastica	vetro
carta	113	11	3	1
indifferenziata	0	129	0	1
plastica	1	0	119	2
vetro	1	0	2	117

Tabella 2.7: Classification Report GBM

	precision	recall	f1-score	support
carta	0.98	0.88	0.93	128
indifferenziata	0.92	0.99	0.96	130
plastica	0.96	0.98	0.97	122
vetro	0.97	0.97	0.97	120
accuracy			0.96	500
macro avg	0.96	0.96	0.96	500
weighted avg	0.96	0.96	0.96	500

Modello: MLP

Script per l'implementazione nell'Appendice A sezione [A.2.2]

Si è deciso di creare 11 Layers di cui i 9 Hidden Layers di dimensione (40, 40, 40, 80, 80, 80, 40, 40, 40):

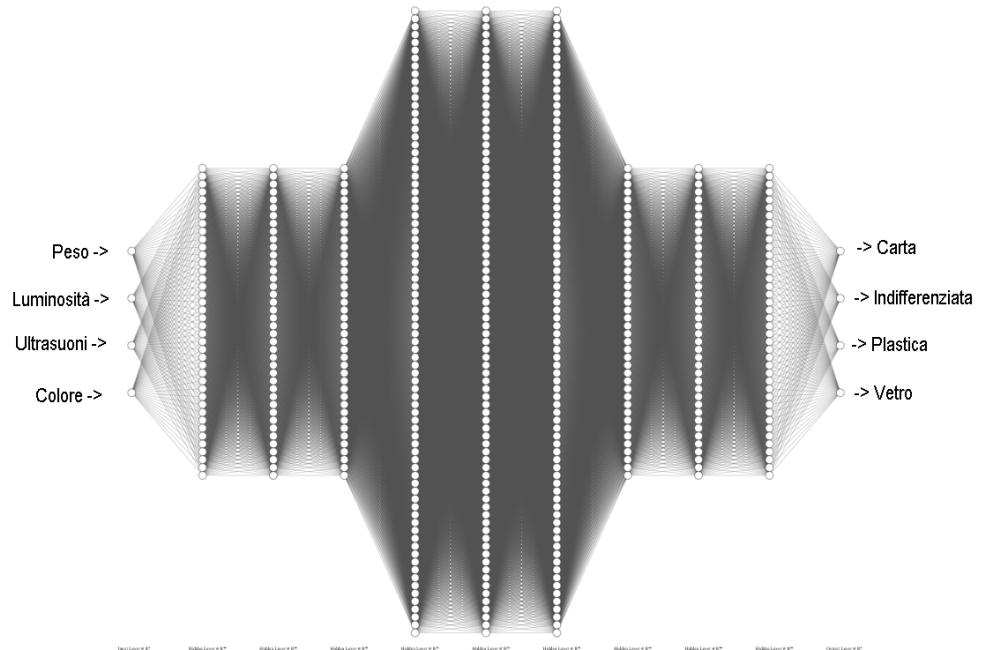


Figura 2.15: MLPClassifier

Tabella 2.8: Confusion matrix MLP

	carta	indifferenziata	plastica	vetro
carta	100	21	6	1
indifferenziata	11	117	1	1
plastica	8	9	103	2
vetro	2	8	8	102

Tabella 2.9: Classification Report MLP

	precision	recall	f1-score	support
carta	0.83	0.78	0.80	128
indifferenziata	0.75	0.90	0.82	130
plastica	0.87	0.84	0.86	122
vetro	0.96	0.85	0.90	120
accuracy			0.84	500
macro avg	0.85	0.84	0.85	500
weighted avg	0.85	0.84	0.85	500

2.3 Dati non strutturati

2.3.1 Convolutional Neural Network (CNN)

Le reti neurali convoluzionali (CNN) sono una classe di Deep Artificial Neural Network che vengono comunemente applicate alle attività di elaborazione delle immagini, ad esempio il rilevamento e l'identificazione degli oggetti.

Le CNN hanno la struttura di fondo in comune con le MLP viste antecedentemente, infatti:

- Struttura Feed Forward: generalmente organizzano i loro nodi (o neuroni) in livelli, con l'output di ogni strato alimentato in avanti allo strato successivo per un'ulteriore elaborazione.
- Apprendimento tramite retropropagazione: durante il training si utilizza una funzione di perdita per calcolare il margine di errore tra l'output effettivo e l'output desiderato. Questo margine di errore viene quindi utilizzato per aggiornare i pesi tramite un processo noto come retropropagazione, con l'obiettivo di ridurre la funzione di perdita e quindi il margine di errore.

Le MLP sono in grado di lavorare con le immagini solo se vengono prima convertite in un vettore di valori di pixel. Il problema di questa trasformazione è la potenziale perdita di integrità spaziale infatti si potrebbero perdere le informazioni su come i pixel si combinano tra loro.

Le reti neurali convoluzionali mantengono l'integrità spaziale delle immagini di input perchè sono in grado di ricavare le informazioni derivanti da una matrice di dati attraverso i filtri convoluzionali; inoltre sono in grado di analizzare immagini a colori (quindi in tre dimensioni) elaborando ogni canale singolarmente, ma mantenendole raggruppate come lo stesso input. In pratica, i filtri convoluzionali sono un insieme di pesi che vengono applicati ai valori dei pixel nella nostra immagine di input. Questi pesi vengono appresi e perfezionati da una retropropagazione durante la fase di allenamento.

L'uso di filtri convoluzionali nelle CNN si ispira al lavoro di Hubel e Wiesel, due neuroscienziati che hanno studiato come le immagini inneschino l'attivazione neuronale nella corteccia visiva. Nel loro studio hanno suggerito che l'elaborazione visiva è un processo gerarchico che inizia con l'identificazione delle caratteristiche di base che vengono poi combinate in costrutti più complessi. In modo analogo, i filtri nelle CNN consentono alla rete neurale di identificare le funzionalità di base ed evidenziarle poi queste possono essere inviate in avanti per un'ulteriore elaborazione, ma prima devono essere corrette da una funzione di attivazione.

Il processo di filtraggio, come si può osservare nell'esempio in Figura 2.16, consiste nello scorrimento di un filtro convoluzionale su un'immagine e nella generazione di una versione filtrata dell'immagine.

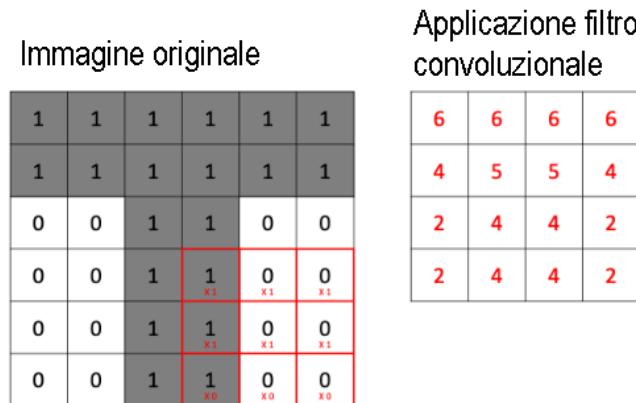


Figura 2.16: Esempio applicazione filtro convoluzionale

I filtri convoluzionali possono accentuare e smorzare caratteristiche specifiche presenti nelle immagini di input (ad esempio curve, bordi o colori).

I filtri all'interno di una CNN possono differenziarsi principalmente per 4 caratteristiche:

- Il numero di filtri convoluzionali: le CNN possono avere decine o addirittura centinaia di filtri. Ogni filtro convoluzionale genera la propria mappa di entità geografiche che viene portata in avanti per un'ulteriore elaborazione.
- Dimensioni del filtro: I filtri convoluzionali devono essere sufficientemente grandi da tenere conto di funzionalità che si estendono su più pixel, ma sufficientemente piccole da poter essere riutilizzabili in un'immagine.
- Stride: questo parametro indica il numero di pixel costituenti l'unità di spostamento per elaborazione. Passi più lunghi si tradurranno in una maggiore velocità computazionale ma in matrici più piccole con una potenziale perdita di caratteristiche importanti.
- Padding (imbottitura): consente, come esemplificato in Figura 2.17, di modificare l'immagine di input riempendola con pixel vuoti per garantire che l'immagine filtrata mantenga le stesse dimensioni dell'immagine di input originale.

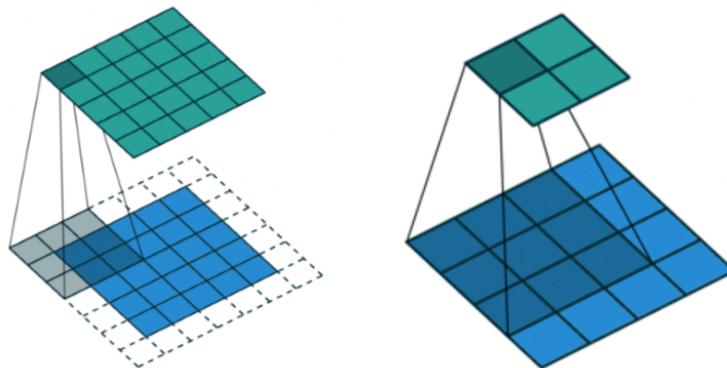


Figura 2.17: Esempio Padding: a sinistra applicazione di un filtro convoluzionale con imbottitura a destra senza Padding

In una rete neurale convoluzionale gli inputs I sono immagini, quindi matrici in tre dimensioni. Formalmente I ha dimensione $H \times W \times C$ dove $H \times W$ sono i pixel mentre C i canali, quindi:

$$I \in \mathbb{R}^{H \times W \times C}$$

Ipotizzando un filtro $K \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$ ed un bias $b \in \mathbb{R}^D$ l'output della procedura convoluzionale è:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^C K_{m,n,c} \cdot I_{i+m,j+n,c} + b$$

I pesi e il bias vengono aggiornati attraverso una procedura di forward-propagation e back-propagation generalmente inserendo alla fine dei filtri una MLP fully connected.

Nelle CNN la funzione di attivazione più comune è la ReLU, questo perchè assicura che solo i nodi con un'attivazione positiva inviano i loro valori in avanti, ciò garantisce che:

- Con meno nodi attivati le elaborazioni da fare sono minori.
- I nodi che vengono attivati positivamente indicano aspetti significativi. Concentrarsi su questi può portare a modelli migliori con punteggi di precisione più elevati.
- C'è meno rumore all'interno della rete, riducendo i pericoli di overfitting o errori di correlazione tra input e uscita desiderata.

Un attributo comune nelle reti neurali convoluzionali è quello del Pooling, esemplificato in Figura 2.18. Il pooling avviene normalmente dopo che le mappe delle funzionalità sono state passate tramite la funzione di attivazione ReLU. L'obiettivo del pooling è ridurre le dimensioni della matrice dell'immagine senza perdita di informazioni. A sua volta, questa operazione, riduce la quantità di elaborazione richiesta, risparmiando tempo e risorse nella fase di formazione. Le varietà di pooling più comuni sono:

- Max Pooling: Accetta il valore massimo in pixel all'interno del filtro.

$$y_{ijk} = \max\{y_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p\}$$

- Average pooling: Prende il valore medio dei pixel all'interno del filtro.

$$y_{ijk} = \text{mean}\{y_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p\}$$

- Sum Pooling: Somma i valori dei pixel all'interno del filtro.

$$y_{ijk} = \text{sum}\{y_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p\}$$

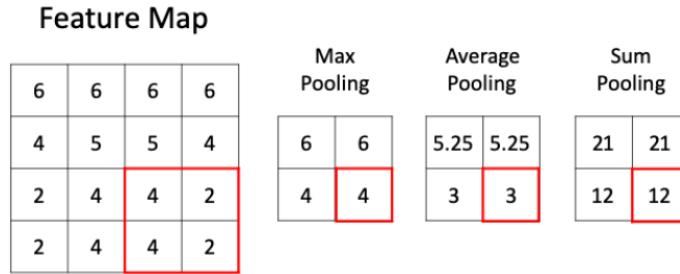


Figura 2.18: Esempio Pooling

Un altro importante elemento costitutivo della CNN è la normalizzazione (normalization batch). Per aumentare la stabilità di una rete neurale si sottrae dai valori in input la loro media batch e successivamente si dividono per la deviazione standard. La normalizzazione viene, di norma, attuata prima di un livello di attivazione ed è utilizzata per velocizzare il processo di riduzione al minimo la funzione di perdita.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta == BN_{\gamma, \beta}(x_i)$$

L'offset β e lo scale factor γ sono parametri apprendibili che vengono aggiornati nella back-propagation come fossero normali pesi. Se si imposta la γ come 1 e β come 0 l'intero processo è solo standardizzazione.

2.3.2 Residual Neural Network (ResNet)

La ResNet è un'architettura CNN di Microsoft Research ideata da Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun [11]. Questa architettura è nata nel 2015 e ha vinto il concorso ImageNet 2015. Fino a qualche anno prima dell'invenzione di questo modello si pensava che più aumentassero gli strati della rete più questa riusciva a captare tutti i segnali per avere una previsione più accurata, infatti tutti i modelli precedenti utilizzavano reti neurali in cui si sovrapponevano molti strati di convoluzione uno dopo l'altro. Con l'aumento dell'utilizzo di questi modelli ci si è accorti che all'aumentare della profondità della rete, la precisione si satura e si degrada rapidamente. Questo problema è detto "problema di degradazione"

Per affrontare questi problemi, gli autori dell'architettura ResNet hanno avuto l'idea di saltare delle connessioni con l'ipotesi che gli strati più profondi dovrebbero essere in grado di imparare qualcosa di uguale agli strati meno profondi. Una possibile soluzione è la copia delle attivazioni da livelli più superficiali e l'impostazione di livelli aggiuntivi nella mappatura delle identità. Queste connessioni bypassano le connessioni intermedie come mostrato in Figura 2.19.

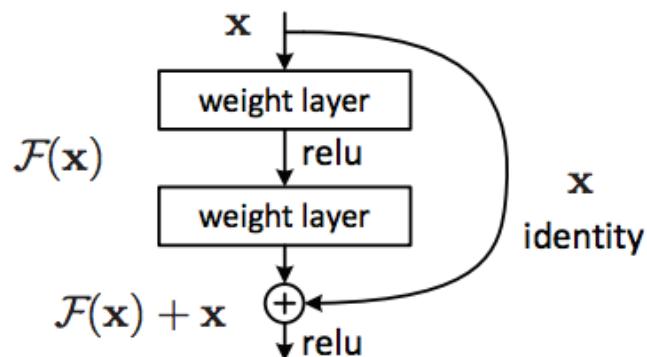


Figura 2.19: Block Residual

Il ruolo di queste connessioni è quello di svolgere la funzione di identità sull'attivazione di un layer più superficiale, che a sua volta produce la stessa attivazione. Questo output viene quindi aggiunto con l'attivazione al livello successivo. Per abilitare queste connessioni o essenzialmente abilitare questa operazione di aggiunta, è necessario garantire le stesse dimensioni delle convoluzioni attraverso la rete, ed è per questa ragione che nelle ResNet i filtri convoluzionali hanno l'imbottitura (il padding) mentre ogni tre blocchi dei residui gli studiosi hanno pensato che fosse necessario un cambio di dimensionalità per riuscire ad addestrare meglio i dati quindi si è pensato di introdurre due filtri convoluzionali senza imbottitura prima dell' "addition" e all'interno del blocco dei residui.

Utilizzando blocchi residui nella rete, è possibile costruire reti di qualsiasi profondità con l'ipotesi che i nuovi livelli stiano effettivamente aiutando ad apprendere nuovi schemi sottostanti nei dati di input. Gli autori dell'articolo sono stati in grado di creare una rete profonda fino a 152 strati.

In questa parte si andrà ad analizzare la ragione per cui le reti residue hanno successo e consentono l'aggiunta più layer senza danneggiare le prestazioni della rete. Si consideri una semplice rete neurale (A) senza rete residua, come mostrato in Figura 2.20. Quindi nella rete (A) l'ingresso X viene inserito in una rete neurale (NN) per ricevere come output A_1 . Ora, si consideri una rete più profonda (B) in cui un blocco residuo (con 2 livelli extra e una connessione salta) viene aggiunto nella rete precedente. Quindi ora A_1 passa al blocco residuo, che, a sua volta dà la nuova attivazione A_3 .

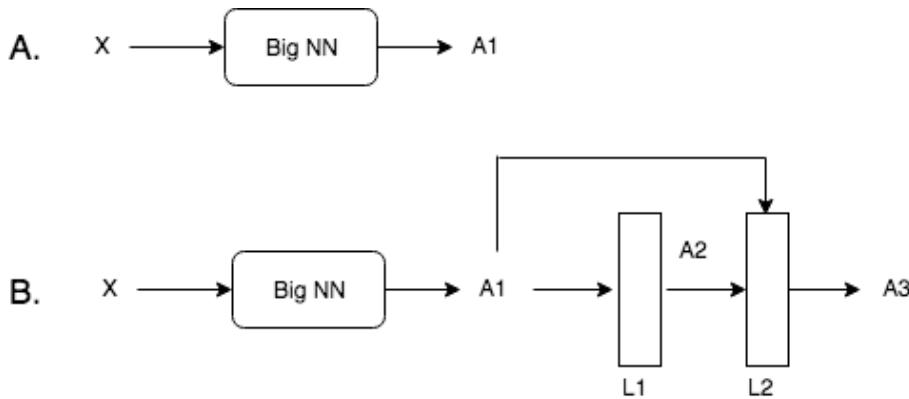


Figura 2.20: Spiegazione ResNet

Se non ci fosse il salto della connessione allora A_3 sarebbe:

$$A_3 = \text{relu}(W_2 A_2 + b_2)$$

dove W_2 e b_2 sono rispettivamente i pesi e la distorsione associati al livello $L2$. Con il salto della connessione viene aggiunto un altro termine A_1 di conseguenza l'equazione di A_3 verrà modificata come:

$$A_3 = \text{relu}(W_2 A_2 + b_2 + A_1)$$

La regolarizzazione $L2$ o i metodi di riduzione del peso, forzeranno W_2 e b_2 ad avvicinarsi allo zero. Nel peggior dei casi, se questi diventano zero, allora

$$A_3 = \text{relu}(A_1)$$

Con l'aggiunta di blocchi residui, la complessità del modello non è aumentata, poiché la suddetta aggiunta sta solo copiando l'attivazione precedente nei livelli successivi. Le prestazioni della rete miglioreranno in quanto è possibile che questi livelli aggiuntivi imparino qualcosa di nuovo dai dati riducendo il problema di degradazione.

Quindi, l'aggiunta dei blocchi residui non pregiudica le prestazioni della rete, ma aumenta le possibilità che nuovi livelli imparino qualcosa di utile.

In questo progetto si è deciso di utilizzare la ResNet con 18 layer con 9 blocchi residui e alcune modificazioni pensate per aumentare l'efficacia delle previsioni.

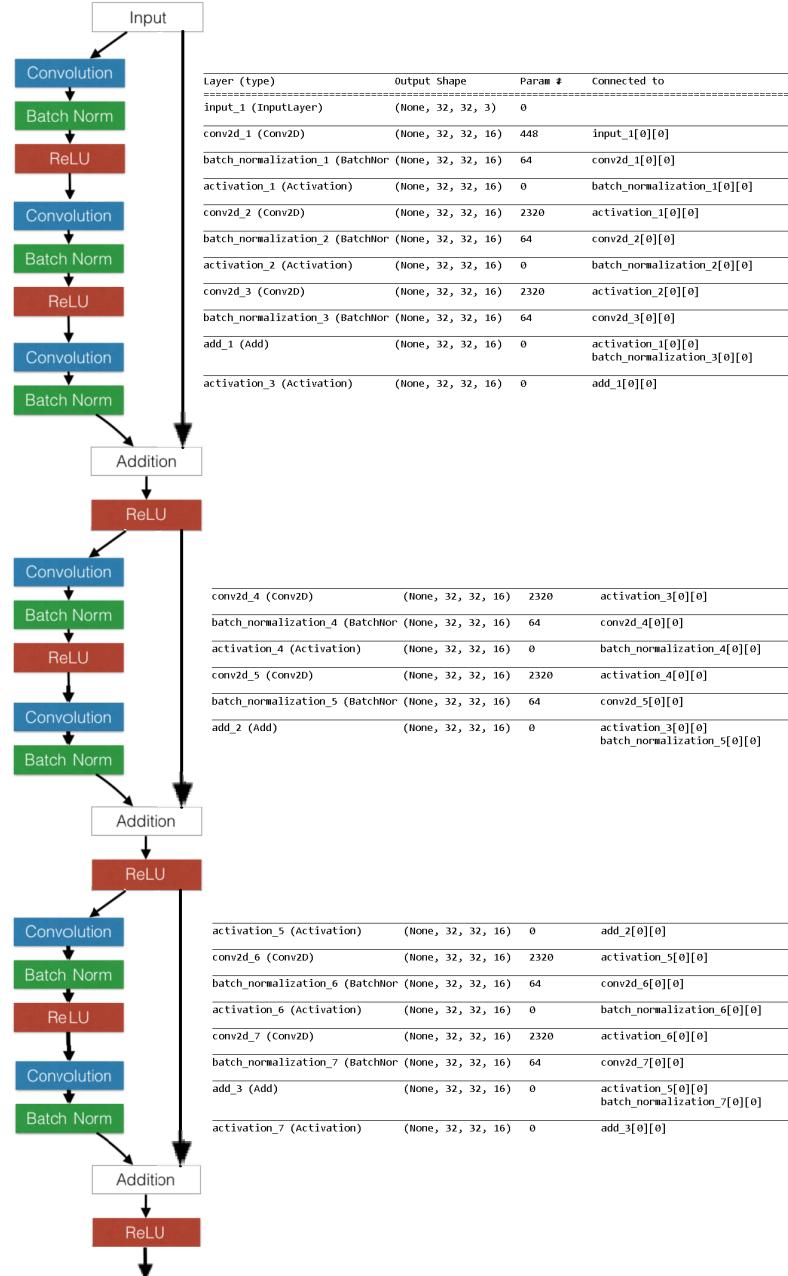


Figura 2.21: Struttura ResNet 1

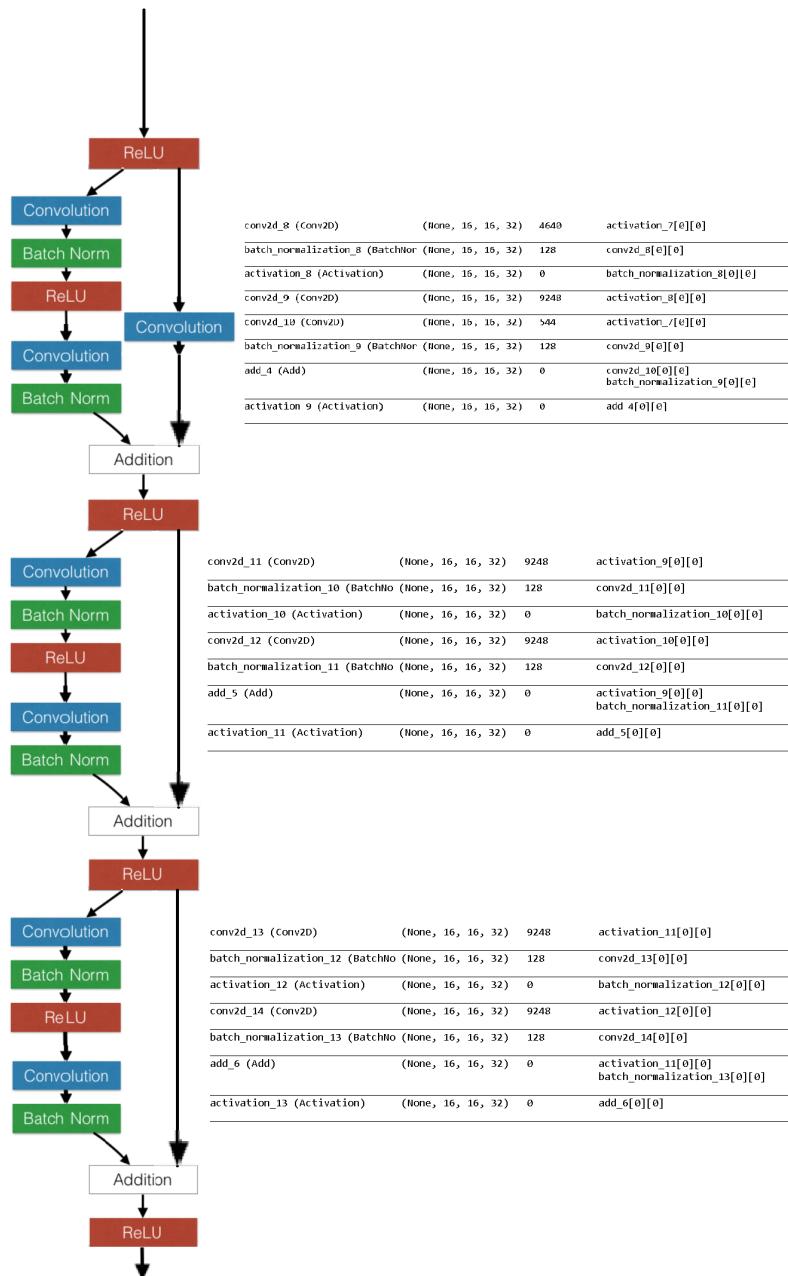


Figura 2.22: Struttura ResNet 2

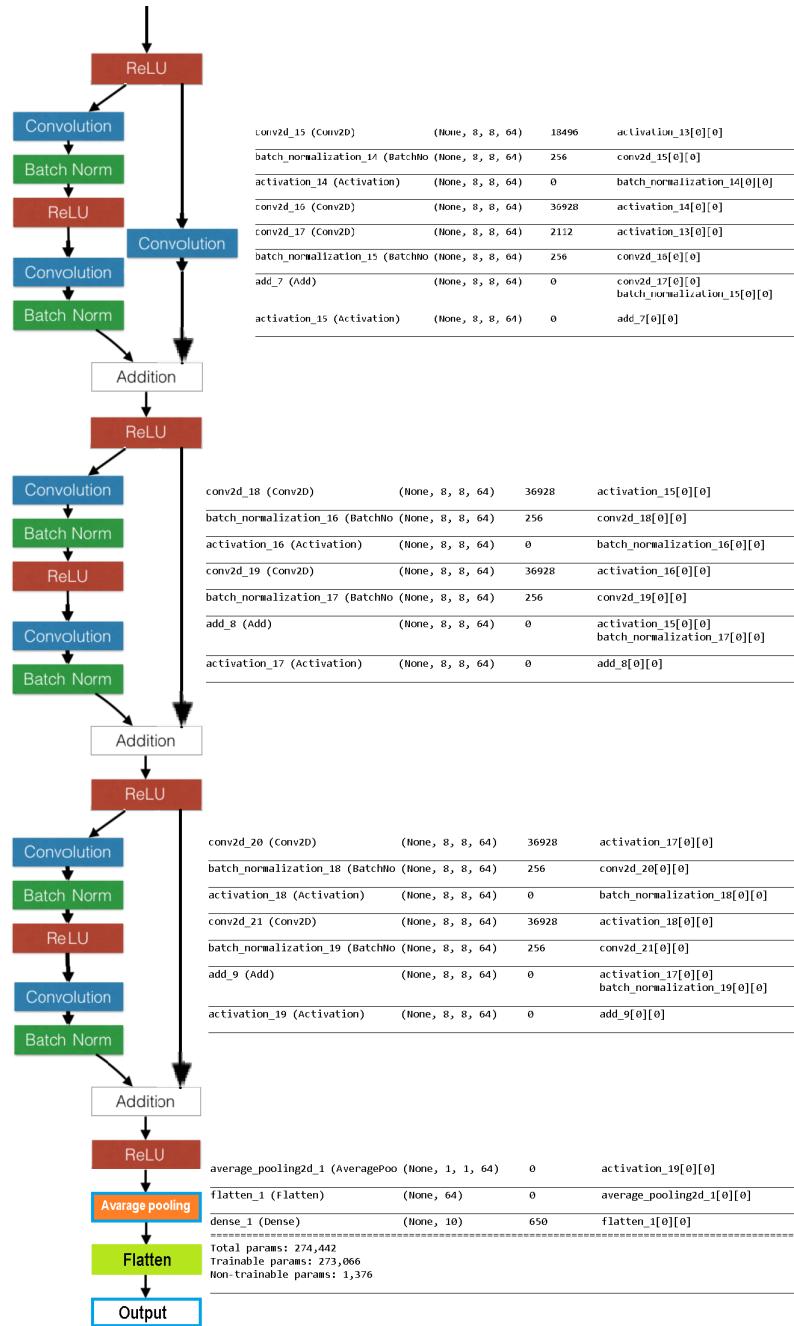


Figura 2.23: Struttura ResNet 3

2.3.3 Risultati

Modello: ResNet su immagini del webscraping

Script per l'implementazione nell'Appendice A sezione [A.2.3]

Tabella 2.10: Confusion matrix ResNet1

	carta	indifferenziata	plastica	vetro
carta	78	13	52	8
indifferenziata	1	274	51	12
plastica	0	3	396	2
vetro	0	1	9	100

Tabella 2.11: Classification Report ResNet1

	precision	recall	f1-score	support
carta	0.99	0.52	0.68	151
indifferenziata	0.94	0.81	0.87	338
plastica	0.78	0.99	0.87	401
vetro	0.82	0.91	0.86	110
accuracy			0.85	1000
macro avg	0.88	0.81	0.82	1000
weighted avg	0.87	0.85	0.84	1000

Modello: ResNet su immagini del Cestino + ChatBot
Script per l'implementazione nell'Appendice A sezione [A.2.3]

Tabella 2.12: Confusion matrix ResNet2

	carta	indifferenziata	plastica	vetro
carta	142	132	21	2
indifferenziata	0	139	0	0
plastica	0	0	297	0
vetro	0	51	6	210

Tabella 2.13: Classification Report ResNet2

	precision	recall	f1-score	support
carta	1	0.48	0.66	197
indifferenziata	0.43	1	0.6	139
plastica	0.92	1	0.96	297
vetro	0.99	0.79	0.88	267
accuracy			0.79	1000
macro avg	0.83	0.82	0.77	1000
weighted avg	0.89	0.79	0.79	1000

Capitolo 3

Performance e previsioni

3.1 Confronto modelli

L'obiettivo di questa parte del progetto è quello di combinare i modelli del capitolo precedente attraverso modelli creati con lo scopo di combinare le previsioni per ottenere un accuratezza più elevata. In particolare, si vuole creare un modello che possa combinare gli algoritmi ResNet addestrati su immagini per creare poi una applicazione che possa dalla foto del rifiuto individuare la classe corrispondente ad esso; inoltre, si vuole costruire un modello che possa combinare dati strutturati e foto per riuscire a sfruttare tutte le informazioni ricavate da questo studio.

Come spiegato nel Capitolo 2.1 per poter pensare di riuscire a combinare modelli creati su basi di dati strutturati insieme a modelli che analizzano immagini era importante creare un dataset che potesse contenere entrambe le informazioni nel crearlo con l'intento di validare gli algoritmi in condizioni differenti si è deciso di cambiare la posizione del cestino così da tentare di valutare e correggere quanto le informazioni prese ed analizzate dal GBM e dalla MLP fossero dipendenti dall'ambiente che circonda il cestino; inoltre, si è deciso per il 50% degli oggetti di registrarli nel cestino ma di inserire le foto fatte con il bot così da valutare come le ResNet si comportassero in ambienti differenti. Innanzitutto occorre valutare le performance dei modelli su questi nuovi dati. Una volta valutate le perfomance sarà obiettivo delle sezioni seguenti trovare un modo per poter mischiare questi risultati per ottenere

delle prestazioni migliori. Sul dataset strutturato sono state eseguite le stesse operazioni di preprocessing attuate nel Capitolo 2.3 sul dataset originale.

Tabella 3.1: Dati non strutturati

	GBM			MLP			
	precision	recall	f1-score	precision	recall	f1-score	support
carta	0.54	0.63	0.58	0.41	0.45	0.43	60
indifferenziata	0.64	0.78	0.71	0.47	0.58	0.52	60
plastica	0.83	0.63	0.72	0.64	0.42	0.51	60
vetro	0.92	0.78	0.85	0.61	0.62	0.61	60
accuracy			0.71			0.52	240
macro avg	0.73	0.71	0.71	0.53	0.52	0.52	240
weighted avg	0.73	0.71	0.71	0.53	0.52	0.52	240

Tabella 3.2: Dati non strutturati

	Modello Web Scraping			Modello Cestino + ChatBot			
	precision	recall	f1-score	precision	recall	f1-score	support
carta	0.79	0.37	0.50	0.66	0.88	0.79	60
indifferenziata	0.74	0.72	0.73	0.97	0.88	0.98	60
plastica	0.50	0.92	0.65	1.00	0.88	0.86	60
vetro	0.70	0.52	0.60	1.00	0.88	0.82	60
accuracy			0.63			0.86	240
macro avg	0.68	0.63	0.62	0.91	0.86	0.86	240
weighted avg	0.68	0.63	0.62	0.91	0.86	0.86	240

Come si può notare dai risultati sul nuovo Validation Set c'è un abbassamento dell'accuratezza globale dovuta dal cambio di ambiente. Per quanto riguarda i dati strutturati si può osservare in Tabella 3.1 come la MLP non riesca a prevedere in modo accurato, infatti abbiamo un abbassamento dell'accuratezza di 32 punti percentuale, anche se non in modo così am-

pio, anche il Gradient Boosting Algorithm non reagisce in modo flessibile al cambiamento dell’ambiente infatti l’accuratezza cala fino a 0.71.

Le ResNet addestrate sui dati non strutturati reagiscono in modo differente. Come era preventivabile il Modello addestrato sulle foto raccolte dal WEB ha un abbassamento dell’accuratezza di previsione importante dovuto alla poca specificità delle foto; mentre la ResNet addestrata sulle foto fatte ad hoc per il progetto reagisce in modo molto accurato anche oltre le aspettative. Dalla performance di questa ultima rete si può intuire quanto queste tipologie di modelli siano potenti anche con una quantità più ridotte di dati.

3.2 Ensemble Learning

3.2.1 Modelli Stacking

I metodi di ensemble vengono comunemente utilizzati per migliorare la precisione predittiva combinando le previsioni di più modelli di apprendimento automatico. Quando questi metodi vengono ideati per l’idea era quella di combinare i cosiddetti predittori “deboli” (Breiman Leo, “Bagging Predictors”; 1996). Tuttavia, un approccio più moderno è quello di creare un insieme di una collezione ben scelta di modelli forti ma diversi (Van der Laan, Polley e Hubbard; 2007).

Con tutte le differenze del caso, la costruzione di potenti modelli di ensemble ha molti parallelismi con la creazione di team di persone esperte in diversi settori (nel mondo degli affari, della scienza, della politica, dello sport, ecc.). Nel cooperare per un progetto ogni membro del team fornisce un contributo significativo e le debolezze e le distorsioni individuali vengono compensate dai punti di forza degli altri membri, così è per i modelli di Ensemble che sfruttano Stronger Learner.

Il tipo più semplice di ensemble è la media non ponderata delle previsioni dei modelli che formano una serie di modelli. In una media non ponderata, ogni modello assume lo stesso peso quando viene costruito un modello di ensemble. Più in generale, si può pensare di utilizzare le medie ponderate così da poter conferire più peso a quei modelli migliori. Ma un approccio ancora migliore potrebbe essere quello di stimare i pesi in modo più intelligente usando un altro livello di stima attraverso un algoritmo di classificazione. Questo approccio è chiamato stacking dei modelli.

Lo stacking dei modelli è un metodo di ensemble efficiente in cui le previsioni, generate utilizzando algoritmi di machine learning, vengono utilizzate come input in un algoritmo di apprendimento di secondo livello. Questo algoritmo di secondo livello viene addestrato per combinare in modo ottimale le previsioni del modello per formare una nuova serie di previsioni. Ad esempio, quando la regressione lineare viene utilizzata come modellazione di secondo livello, stima questi pesi minimizzando gli errori minimi quadrati. Tuttavia, la modellazione del secondo livello non è limitata ai soli modelli lineari; la relazione tra i predittori può essere più complessa, aprendo le porte all'impiego di altri algoritmi di apprendimento automatico.

L'overfitting è un problema particolarmente importante nello stacking dei modelli, perché vengono combinati tanti predittori che prevedono tutti lo stesso target questo può essere parzialmente causato dalla collinearità tra i predittori. Una possibile soluzione a questo problema è quella di generare delle serie diversificate di modelli utilizzando vari metodi (come alberi decisionali, reti neurali e/o altri metodi) oppure utilizzare diversi tipi di dati così da cercare di inserire nel modello finale informazioni nuove non presenti nei modelli precedenti.

L'applicazione di modelli sovrapposti ai problemi dei Big Data può produrre una maggiore accuratezza e robustezza delle previsioni rispetto ai singoli modelli. L'approccio dello stacking dei modelli sposta l'obiettivo dalla modellizzazione e dalla ricerca del modello migliore a quello di trovare una raccolta di modelli complementari validi. Naturalmente, questo metodo comporta costi computazionali aggiuntivi sia perché è necessario addestrare un gran numero di modelli sia perché spesso risulta necessario utilizzare la convalida incrociata per evitare un eccesso di adattamento.

L'algoritmo per creare un modello a più livelli con il metodo stacking è costituito da tre fasi:

- Impostare l'ensemble:
 - Specificare un elenco di L Base Learner (con un insieme specifico di parametri del modello).
 - Specificare un meta-algoritmo di apprendimento.
- Creare l'ensemble:
 - Addestrare ciascuno degli L Base Learner sul training set $Y = F(X)$ e trovare delle previsioni Z di primo livello.

- Eseguire il training del meta-algoritmo di apprendimento sui dati di livello 1 $Y = F(Z)$ con $Z = F(X)$. Il “modello di ensemble”, quindi, sarà costituito dagli L modelli di apprendimento di base e il modello di stacking che combina i modelli precedenti.
- Prevedere nuovi dati:
 - Per generare previsioni dell’insieme, si generano innanzitutto le previsioni dai Base Learner.
 - Si inseriscono tali previsioni nello modello di Ensemble per generare la previsione dell’insieme.

3.2.2 Risultati

Il meta-algoritmo scelto in questa parte del progetto corrisponde al K-nearest neighbors (KNN), questo può essere utilizzato sia per problemi predittivi di classificazione che di regressione. Tuttavia, è più ampiamente utilizzato nei problemi di classificazione per il suo approccio semplice che gli permette di non fare assunzioni circa la struttura delle variabili.

L’intuizione alla base dell’algoritmo KNN è quella di calcolare la distanza di un nuovo punto di dati a tutti gli altri punti nel training e selezionare quindi i punti dati K-più vicini, dove K può essere qualsiasi numero intero. Infine l’algoritmo assegna il punto dati alla classe a cui appartiene la maggior parte dei punti compresi nell’intervallo dei K più vicini.

Quindi i parametri di tuning del modello risultano essere la misura di distanza adottata e il numero dei K punti da selezionare per classificare.

- Le misure di distanza possono essere di qualsiasi tipo anche se solitamente si utilizzano quella Euclidea, di Manhattan o di Minkowski.
- Il parametro K può essere scelto a priori in modo arbitrario oppure attraverso tecniche di validazione come Cross Validation.

Con l’aiuto di un semplice esempio si andrà a comprendere la validità di questo modello. Si supponga di disporre di un set di dati con due variabili, che quando viene tracciato, è simile a quello in Figura 3.1. Lo scopo è quello di classificare un nuovo punto dati con “ X ” in classe “Verde” o “Rosso”. Supponiamo di fissare il valore di K uguale a 3. L’algoritmo KNN inizia calcolando la distanza del punto X da tutti i punti e, come mostrato in

Figura 3.1 nell'immagine di sinistra, trova i $K = 3$ punti più vicini con la distanza minima rispetto al punto X .

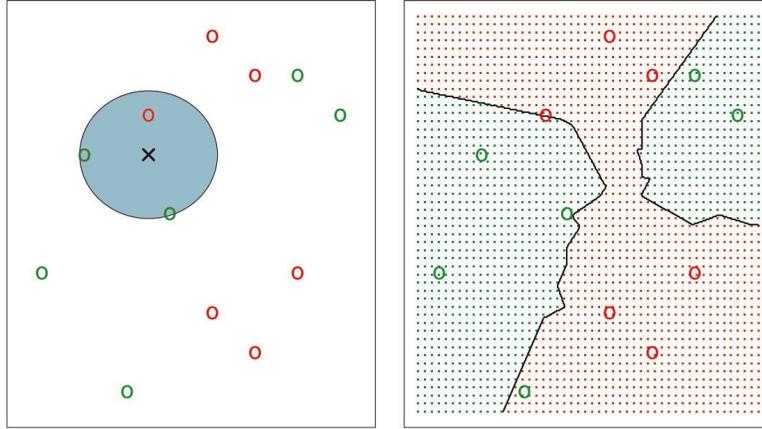


Figura 3.1: Esempio K-nearest neighbors (KNN)

Il passaggio finale dell'algoritmo KNN consiste nell'assegnare un nuovo punto alla classe a cui appartiene la maggior parte dei $K = 3$ punti più vicini. Dalla figura precedente possiamo vedere che due dei tre punti più vicini appartengono alla classe “Verde” mentre uno appartiene alla classe “Rosso”. Pertanto il nuovo punto dati verrà classificato come “Verde”. Questo procedimento viene reiterato per ogni punto, come mostrato in Figura 3.1 nell'immagine di destra, così da riuscire a classificare operando una divisione del piano.

Per costruire il modello si è diviso il set in train e test data successivamente si è utilizzata una k-fold cross validation con $k = 5$ per stimare il parametro K dell'algoritmo sul training set che poi è stato utilizzato per stimare i valori di test. Come misura di distanza si è utilizzata la “distanza di Minkowski” con $p = 2$, la quale per una serie di punti monodimensionali $p_x = \{p_{x,1}, \dots, p_{x,n}\}$ rispetto ad un punto q_x risulta essere:

$$\left(\sum_{i=1}^n |p_{x,i} - q_x|^p \right)^{\frac{1}{p}}$$

Modello: Ensemble Learning tra ResNet

Script per l'implementazione nell'Appendice A sezione [A.3.1]

Tabella 3.3: Confusion matrix

	carta	indifferenziata	plastica	vetro
carta	12	0	0	0
indifferenziata	0	19	0	0
plastica	3	30	14	0
vetro	2	10	0	10

Tabella 3.4: Classification Report

	precision	recall	f1-score	support
carta	0.71	1.00	0.83	12
indifferenziata	1.00	1.00	1.00	19
plastica	1.00	0.82	0.90	17
vetro	1.00	0.83	0.91	12
accuracy			0.92	60
macro avg	0.93	0.91	0.92	60
weighted avg	0.94	0.92	0.92	60

Modello: Ensemble Learning su dati strutturati e dati non strutturati

Script per l'implementazione nell'Appendice A sezione [A.3.1]

Tabella 3.5: Confusion matrix

	carta	indifferenziata	plastica	vetro
carta	11	0	1	20
indifferenziata	0	19	0	0
plastica	0	0	17	0
vetro	1	0	1	10

Tabella 3.6: Classification Report

	precision	recall	f1-score	support
carta	1	0.92	0.92	12
indifferenziata	1.00	1.00	1.00	19
plastica	0.89	1.00	0.94	17
vetro	1.00	0.83	0.91	12
accuracy			0.95	60
macro avg	0.95	0.94	0.94	60
weighted avg	0.95	0.95	0.95	60

3.3 Esempi pratici di applicazioni

3.3.1 Bot di Telegram

spiegare idea
tempi

3.3.2 Cestino

spiegare idea
funzionamento
problema con il motore
tempi

Svolta solo in fase progettuale per un problema causato dal funzionamento di un motore

Nella terza pagina indicizzata “Form3” bisogna inizializzarla come visto per la precedente e poi una volta inserito l’oggetto cliccare sul pulsante riconosci per attivare il cestino al riconoscimento. L’output della previsione verrà inviato al cestino e scritto nella TextBox posta sopra il pulsante.



Figura 3.2: Form3

Conclusioni

- pregi del lavoro buoni risultati dei modelli utilizzo di modelli nuovi
 - punti critici (umido) (robotica- ambiente non completamente isolato e non completamente stabile, problema con il motore)
 - possibili miglioramenti (qui soffermarsi un po')
 - aumentare test set ensemble
 - con resnet 34 o + o inception 3 o 4 miglioramento dei modelli
 - Reinforced learning (apprendere dagli errori)
 - Amazon Web Services
 - Lettura bar code

Appendice A

Script di programmazione

A.1 Reperimento dei dati

A.1.1 Web Scraping (R)

```
library(xml2)
library(rvest)
library(stringr, warn.conflicts = F)

webscraing_img <-function(img){
  txt<-"img"
  for(i in 1:length(img)){
    # 1
    try({page<-read_html(paste0("https://www.google.com/search?source=&q="
      ,gsub(" ", "+",img[i]), "+jpg&source=lnms&tbo=isch"))}
    for (t in 1:length(html_nodes(page,xpath = '//img'))){ 
      txt<-c(txt,xml_attrs(html_nodes(page, xpath =
        "//img")[[t]])[["src"]])
    })
    # 2
    try({page<-read_html(paste0("https://www.bing.com/images/search?q="
      ,gsub(" ", "+",img[i]), "+jpg&FORM=HDRSC2"))}
    for (t in 1:length(html_nodes(page,xpath = '//img'))){ 
      txt<-c(txt,xml_attrs(html_nodes(page, xpath =
        "//img")[[t]])[["src"]])
    })
  }
}
```

```

# 3
try({page<-read_html(paste0("https://pixabay.com/it/images/search/"
  ,gsub(" ", "%20", img[i])))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1a
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
  ", "+,img[i]), "+jpg&source=lnms&tbo=isch&tbs=isz:l"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1b
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
  ", "+,img[i]), "+jpg&source=lnms&tbo=isch&tbs=isz:m"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1c
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
  ", "+,img[i]), "+jpg&source=lnms&tbo=isch&tbs=isz:s"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.1
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
  ", "+,img[i]), "+jpg&tbas=0&tbo=isch&tbs=ic:specific,isc:white"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.2

```

```

try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:teal"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.3
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:black"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.4
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:green"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.5
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:red"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.6
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:blue"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.7

```

```

try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:gray"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.8
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:yellow"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.9
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:pink"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
# 1.10
try({page<-read_html(paste0("https://www.google.com/search?source=&q="
  ,gsub(
    ", "+,img[i]), "+jpg&tbas=0&tbm=isch&tbs=ic:specific,isc:brown"))
for (t in 1:length(html_nodes(page,xpath = '//img')))) {
txt<-c(txt,xml_attrs(html_nodes(page, xpath =
  "//img")[[t]])[["src"]])
}})
}
d<-substr(txt,1,3)=="htt"
immaginidascaricare<-txt[d]
return(immaginidascaricare)
}

```

```




---



```

A.1.2 Bot Telegram per salvare i dati (Python)

```

import telepot
from datetime import datetime
import time

def rispondi(msg):
    content_type, chat_type, chat_id = telepot.glance(msg)
    if content_type == 'photo':
        bot.download_file(msg['photo'][1]['file_id'], 'dati/image'+
            str(datetime.now())[0:19] +'.png')
        bot.sendMessage(chat_id, 'Thanks for send me the photo! See
            you soon!')

bot =telepot.Bot('###---TOKEN---###')
bot.message_loop(rispondi)

print 'I am listening ...'

```

```
while 1:  
    time.sleep(10)
```

A.1.3 Arduino (C++)

```
#include <NewPing.h>  
#include <Servo.h>  
#include "HX711.h"  
  
// HX711.DOUT - pin #A4  
// HX711.PD_SCK - pin #A5  
  
HX711 scale(A4, A5); // parameter "gain" is omitted; the default  
// value 128 is used by the library  
  
  
NewPing sonar1(10, 9, 200);  
NewPing sonar2(12, 11, 200);  
int magnetic, md, luce1, luce2, distanza1 ;  
const int s0 = 3;  
const int s1 = 4;  
const int s2 =7;  
const int s3 = 5;  
const int out = 6;  
const int motorpin = 52;  
Servo motoreservo;  
int frequency = 0;  
int color=0;  
  
void setup() {  
    // put your setup code here, to run once:  
  
    pinMode(A2,INPUT);  
    pinMode(A3,INPUT);  
    pinMode(A0,INPUT);  
  
    Serial.begin(9600);
```

```

pinMode(s0, OUTPUT);
pinMode(s1, OUTPUT);
pinMode(s2, OUTPUT);
pinMode(s3, OUTPUT);
pinMode(out, INPUT);
motoreservo.attach(motorpin);
digitalWrite(s0, HIGH);
digitalWrite(s1, LOW);

// set HX711
scale.set_scale(2280.f);
scale.tare();
}

void loop() {
    // put your main code here, to run repeatedly:

    magnetic=analogRead(A0);
    luce1=analogRead(A2);
    luce2=analogRead(A3);
    unsigned int distanza1 = sonar1.ping();
    distanza1=distanza1 / US_ROUNDTRIP_CM;

    unsigned int distanza2 = sonar2.ping();
    distanza2=distanza2 / US_ROUNDTRIP_CM;

    Serial.print(luce1);
    Serial.print(",");
    Serial.print(luce2);
    Serial.print(",");
    Serial.print(distanza1);
    Serial.print(",");
    Serial.print(distanza2);
    Serial.print(",");
}

```

```

color = readColor();
Serial.print(scale.get_units(10)*28.3495231, 1);
Serial.println(" ");

color=0;
scale.power_down();
delay(5000);
scale.power_up();

char a;
a = Serial.read ();
switch (a) {
    case '1': // tira su
        motoreservo.write(80);
        Serial.println("SU");
        break;
    case '2': // tira giu
        motoreservo.write(15);
        Serial.println("GIU");
        break;
    case '3': // Ricalcola
        motoreservo.write(60);
        delay(1000);
        motoreservo.write(70);
        Serial.println("Ricalcola");
        break;
    default : // In tutti gli altri casi visualizzo un messaggio
        break;
}
}

```

```

//Read-Color Function
int readColor() {

    //Setting red filtered photodiodes to be read
    digitalWrite(s2, LOW);
    digitalWrite(s3, LOW);

    //Reading the output frequency
    frequency = pulseIn(out, LOW);
    int R = frequency;

    //Printing the value on the serial monitor
    Serial.print(frequency); //printing rosso color frequency
    Serial.print(",");
    Serial.print(" ");

    //Setting Green filtered photodiodes to be read
    digitalWrite(s2, HIGH);
    digitalWrite(s3, HIGH);

    //Reading the output frequency
    frequency = pulseIn(out, LOW);
    int G = frequency;

    //Printing the value on the serial monitor
    Serial.print(frequency);
    //printing verde color frequency
    Serial.print(",");
    Serial.print(" ");

    //Setting Blue filtered photodiodes to be read
    digitalWrite(s2, LOW);
    digitalWrite(s3, HIGH);

    //Reading the output frequency
    frequency = pulseIn(out, LOW);
    int B = frequency;

    //Printing the value on the serial monitor
    Serial.print(frequency); //printing blu color frequency
}

```

```

Serial.print(",");
if(R<260 & R>230 & G<860 & G>800){
    color = 1; // Rosso
}
if(G<420 & G>370 & B<350 & B>305){
    color = 2; // Blu
}
if(R<450 & R>420 & G<420 & G>390){
    color = 3; // Verde
}
return color;
}

```

A.1.4 Software per salvare i dati (VB.NET)

Form1: Funzione

```

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs)
        Handles Button1.Click
        Form2.Show()
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs)
        Handles Button2.Click
        Form3.Show()
    End Sub
End Class

```

Form1: Grafica

```

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>
-
Partial Class Form1

```

```

Inherits System.Windows.Forms.Form

'Form esegue l'override del metodo Dispose per pulire l'elenco
dei componenti.
<System.Diagnostics.DebuggerNonUserCode()> _
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
    End If
    Finally
        MyBase.Dispose(disposing)
    End Try
End Sub

'Richiesto da Progettazione Windows Form
Private components As System.ComponentModel.IContainer

'NOTA: la procedura che segue è richiesta da Progettazione
Windows Form
'Può essere modificata in Progettazione Windows Form.
'Non modificarla mediante l'editor del codice.
<System.Diagnostics.DebuggerStepThrough()> _
Private Sub InitializeComponent()
    Me.Button1 = New System.Windows.Forms.Button()
    Me.Button2 = New System.Windows.Forms.Button()
    Me.SuspendLayout()
    '
    'Button1
    '
    Me.Button1.Location = New System.Drawing.Point(1, 1)
    Me.Button1.Name = "Button1"
    Me.Button1.Size = New System.Drawing.Size(456, 297)
    Me.Button1.TabIndex = 0
    Me.Button1.Text = "Addestramento"
    Me.Button1.UseVisualStyleBackColor = True
    '
    'Button2
    '
    Me.Button2.Location = New System.Drawing.Point(530, 1)

```

```

Me.Button2.Name = "Button2"
Me.Button2.Size = New System.Drawing.Size(456, 297)
Me.Button2.TabIndex = 1
Me.Button2.Text = "Riconoscimento"
Me.Button2.UseVisualStyleBackColor = True
,
'Form1
,
Me.AutoScaleDimensions = New System.Drawing.SizeF(16.0!,
31.0!)
Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
Me.ClientSize = New System.Drawing.Size(996, 312)
Me.Controls.Add(Me.Button2)
Me.Controls.Add(Me.Button1)
Me.Name = "Form1"
Me.Text = "Form1"
Me.ResumeLayout(False)

End Sub

Friend WithEvents Button1 As Button
Friend WithEvents Button2 As Button
End Class

```

Form2

```

Imports AForge
Imports AForge.Video
Imports AForge.Video.DirectShow
Imports System.IO
Imports System.IO.Ports
Imports System.Threading
Imports System.Data
Imports VB = Microsoft.VisualBasic

```

```

Public Class Form2

    Private Sub wait(ByVal Seconds As Double, Optional ByRef
                    BreakCondition As Boolean = False)
        Dim l_WaitUntil As Date
        l_WaitUntil = Now.AddSeconds(Seconds)
        Do Until Now > l_WaitUntil
            If BreakCondition Then Exit Do
        Loop
    End Sub

    Dim CAMERA As VideoCaptureDevice
    Dim bmp As Bitmap

    Private Sub Button7_Click(sender As Object, e As EventArgs)
        Handles Button7.Click
        Dim cameras As VideoCaptureDeviceForm = New
            VideoCaptureDeviceForm
        If cameras.ShowDialog() = Windows.Forms.DialogResult.OK Then
            CAMERA = cameras.VideoDevice
            AddHandler CAMERA.NewFrame, New
                NewFrameEventHandler(AddressOf Captured)
            CAMERA.Start()
        End If
    End Sub

    Private Sub Captured(sender As Object, eventArgs As
        NewFrameEventArgs)

        bmp = DirectCast(eventArgs.Frame.Clone(), Bitmap)
        PictureBox1.Image = DirectCast(eventArgs.Frame.Clone(),
            Bitmap)
    End Sub

```

```

End Sub

Private Sub Button8_Click(sender As Object, e As EventArgs)
    Handles Button8.Click
    PictureBox2.Image = PictureBox1.Image
    Dim timeStamp As DateTime = DateTime.Now
    ' Dim exePath As String = Application.StartupPath()
    Dim path As String = "PATH_prova" + timeStamp.ToString(" MMM
        ddd d HH_mm yyyy") + ".jpg"
    PictureBox2.Image.Save(path, Imaging.ImageFormat.Jpeg)
End Sub

Private Sub Button1_Click(sender As Object, e As EventArgs)
    Handles Button1.Click
    '1

    Button1.Enabled = False
    Button2.Enabled = False
    Button3.Enabled = False
    Button6.Enabled = False

    Dim testo As String
    Dim dataArr() As String

    Dim h As Integer = 0
    Dim m As Integer = 0

    Dim index As Integer = 0

    For i As Integer = 1 To 10

        PictureBox2.Image = PictureBox1.Image

        testo = "carta," + SerialPort1.ReadExisting
        dataArr = testo.Split(",")
        SerialPort1.WriteLine("0")
        wait(14)
        For t As Integer = 1 To 1000
            testo = "carta," + SerialPort1.ReadExisting
            dataArr = testo.Split(",")
        
```

```

        If dataArr.Length = 11 Then
            Exit For
        End If
    Next
    If dataArr.Length = 11 Then
        DataGridView1.Rows.Add(dataArr)
        Dim timeStamp As DateTime = DateTime.Now
        Dim ora As String = timeStamp.ToString
        ora = ora.Replace(":", "_")
        ora = ora.Replace("/", "_")
        ora = ora.Replace(" ", "_")
        Dim exePath As String = Application.StartupPath()
        Dim path As String = exePath + "\carta" + ora +
            i.ToString + ".jpg"
        PictureBox2.Image.Save(path, Imaging.ImageFormat.Jpeg)
        h = h + 1
    End If
    m = i
    SerialPort1.WriteLine("3")
    wait(10)
Next

Button1.Enabled = True
Button2.Enabled = True
Button3.Enabled = True
Button6.Enabled = True

MessageBox.Show("Fatto, i record registrati sono " &
    h.ToString & " su " & m & " tentativi")

End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs)
    Handles Button2.Click
    '1

    Button1.Enabled = False
    Button2.Enabled = False

```

```

Button3.Enabled = False
Button6.Enabled = False

Dim testo As String
Dim dataArr() As String

Dim h As Integer = 0
Dim m As Integer = 0

Dim index As Integer = 0

For i As Integer = 1 To 10

    PictureBox2.Image = PictureBox1.Image

    testo = "plastica," + SerialPort1.ReadExisting
    dataArr = testo.Split(",")
    SerialPort1.WriteLine("0")
    wait(14)
    For t As Integer = 1 To 1000
        testo = "plastica," + SerialPort1.ReadExisting
        dataArr = testo.Split(",")
        If dataArr.Length = 11 Then
            Exit For
        End If
    Next
    If dataArr.Length = 11 Then
        DataGridView1.Rows.Add(dataArr)
        Dim timeStamp As DateTime = DateTime.Now
        Dim ora As String = timeStamp.ToString
        ora = ora.Replace(":", "_")
        ora = ora.Replace("/", "_")
        ora = ora.Replace(" ", "_")
        Dim exePath As String = Application.StartupPath()
        Dim path As String = exePath + "\plastica" + ora +
            i.ToString + ".jpg"
        PictureBox2.Image.Save(path, Imaging.ImageFormat.Jpeg)
        h = h + 1
    End If
    m = i

```

```

        SerialPort1.WriteLine("3")
        wait(10)
    Next

    Button1.Enabled = True
    Button2.Enabled = True
    Button3.Enabled = True
    Button6.Enabled = True

    MessageBox.Show("Fatto, i record registrati sono " &
        h.ToString() & " su " & m & " tentativi")

End Sub

Private Sub Button3_Click(sender As Object, e As EventArgs)
    Handles Button3.Click
    '1

    Button1.Enabled = False
    Button2.Enabled = False
    Button3.Enabled = False
    Button6.Enabled = False

    Dim testo As String
    Dim dataArr() As String

    Dim h As Integer = 0
    Dim m As Integer = 0

    Dim index As Integer = 0

    For i As Integer = 1 To 10

        PictureBox2.Image = PictureBox1.Image

        testo = "vetro," + SerialPort1.ReadExisting
        dataArr = testo.Split(",")
        SerialPort1.WriteLine("0")
        wait(14)
        For t As Integer = 1 To 1000

```

```

testo = "vetro," + SerialPort1.ReadExisting
dataArr = testo.Split(",")
If dataArr.Length = 11 Then
    Exit For
End If
Next
If dataArr.Length = 11 Then
    DataGridView1.Rows.Add(dataArr)
    Dim timeStamp As DateTime = DateTime.Now
    Dim ora As String = timeStamp.ToString
    ora = ora.Replace(":", "_")
    ora = ora.Replace("/", "_")
    ora = ora.Replace(" ", "_")
    Dim exePath As String = Application.StartupPath()
    Dim path As String = exePath + "\vetro" + ora +
        i.ToString + ".jpg"
    PictureBox2.Image.Save(path, Imaging.ImageFormat.Jpeg)
    h = h + 1
End If
m = i
SerialPort1.WriteLine("3")
wait(10)
Next

Button1.Enabled = True
Button2.Enabled = True
Button3.Enabled = True
Button6.Enabled = True

MessageBox.Show("Fatto, i record registrati sono " &
    h.ToString & " su " & m & " tentativi")

End Sub

Private Sub Button6_Click(sender As Object, e As EventArgs)
    Handles Button6.Click
'1

```

```

Button1.Enabled = False
Button2.Enabled = False
Button3.Enabled = False
Button6.Enabled = False

Dim testo As String
Dim dataArr() As String

Dim h As Integer = 0
Dim m As Integer = 0

Dim index As Integer = 0

For i As Integer = 1 To 10

    PictureBox2.Image = PictureBox1.Image

    testo = "indifferenziata," + SerialPort1.ReadExisting
    dataArr = testo.Split(",")
    SerialPort1.WriteLine("0")
    wait(14)
    For t As Integer = 1 To 1000
        testo = "indifferenziata," + SerialPort1.ReadExisting
        dataArr = testo.Split(",")
        If dataArr.Length = 11 Then
            Exit For
        End If
    Next
    If dataArr.Length = 11 Then
        DataGridView1.Rows.Add(dataArr)
        Dim timeStamp As DateTime = DateTime.Now
        Dim ora As String = timeStamp.ToString
        ora = ora.Replace(":", "_")
        ora = ora.Replace("/", "_")
        ora = ora.Replace(" ", "_")
        Dim exePath As String = Application.StartupPath()
        Dim path As String = exePath + "\indifferenziata" + ora
            + i.ToString + ".jpg"
        PictureBox2.Image.Save(path, Imaging.ImageFormat.Jpeg)
        h = h + 1
    End If
Next

```

```

End If
m = i
SerialPort1.WriteLine("3")
wait(10)
Next

Button1.Enabled = True
Button2.Enabled = True
Button3.Enabled = True
Button6.Enabled = True

MessageBox.Show("Fatto, i record registrati sono " &
    h.ToString() & " su " & m & " tentativi")

End Sub

Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Me.WindowState = FormWindowState.Maximized
    Me.CenterToParent()
    Button1.Enabled = False
    Button2.Enabled = False
    Button3.Enabled = False
    Button6.Enabled = False
    Button10.Enabled = False
    Button10.BringToFront()
    Button11.Enabled = False
    Button11.SendToBack()

    Dim exePath As String = Application.StartupPath()

    Dim fname As String = exePath + "\DATI.csv"

    Dim dt As New DataTable
    DataGridView1.ColumnCount = 11

```

```

DataGridView1.Columns(0).Name = "Type"
DataGridView1.Columns(1).Name = "Peso"
DataGridView1.Columns(2).Name = "luce1"
DataGridView1.Columns(3).Name = "luce2"
DataGridView1.Columns(4).Name = "VARdistanza1"
DataGridView1.Columns(5).Name = "MEANDistanza1"
DataGridView1.Columns(6).Name = "VARdistanza2"
DataGridView1.Columns(7).Name = "MEANDistanza2"
DataGridView1.Columns(8).Name = "rosso"
DataGridView1.Columns(9).Name = "verde"
DataGridView1.Columns(10).Name = "blu"

```

```

Using MyReader As New Microsoft.VisualBasic.
    FileIO.TextFieldParser(fname)
    MyReader.TextFieldType = FileIO.FieldType.Delimited
    MyReader.SetDelimiters(",")  

    Dim currentRow As String()
    While Not MyReader.EndOfData
        Try
            currentRow = MyReader.ReadFields()

            DataGridView1.Rows.Add(currentRow)
        Catch ex As Microsoft.VisualBasic.
            FileIO.MalformedLineException
            MsgBox("Line " & ex.Message &
                "is not valid and will be skipped.")
        End Try
    End While
End Using

End Sub

Private Sub Button9_Click(sender As Object, e As EventArgs)
    Handles Button9.Click
    ComboBox1.Items.Clear()
    Dim myPort As Array

```

```

Dim i As Integer
myPort = IO.Ports.SerialPort.GetPortNames()
ComboBox1.Items.AddRange(myPort)
i = ComboBox1.Items.Count
i = i - i
Try
    ComboBox1.SelectedIndex = i
Catch ex As Exception
    Dim result As DialogResult
    result = MessageBox.Show("Com port not detected", "Warning
        !!!", MessageBoxButtons.OK)
    ComboBox1.Text = ""
    ComboBox1.Items.Clear()
    Call Form2_Load(Me, e)
End Try

Button10.Enabled = True
Button10.BringToFront()
ComboBox1.DroppedDown = True

End Sub

Private Sub Button10_Click(sender As Object, e As EventArgs)
Handles Button10.Click

Button10.Enabled = False
Button10.SendToBack()

SerialPort1.BaudRate = 9600
SerialPort1.PortName = ComboBox1.SelectedItem
SerialPort1.Open()
' per capire il testo (serial.println)
' Dim testo As Single = SerialPort1.ReadExisting
' Dim testo1 As String = testo.ToString

Button11.Enabled = True
Button11.BringToFront()

Button1.Enabled = True

```

```

        Button2.Enabled = True
        Button3.Enabled = True
        Button6.Enabled = True
    End Sub

    Private Sub Button11_Click(sender As Object, e As EventArgs)
        Handles Button11.Click

        Button11.Enabled = False
        Button11.SendToBack()

        SerialPort1.Close()

        Button10.Enabled = True
        Button10.BringToFront()

        Button1.Enabled = False
        Button2.Enabled = False
        Button3.Enabled = False
        Button6.Enabled = False

    End Sub

    Private Sub Button12_Click(sender As Object, e As EventArgs)
        Handles Button12.Click

            'create empty string
            Dim thecsvfile As String = String.Empty
            'get the column headers
            For Each column As DataGridViewColumn In DataGridView1.Columns
                thecsvfile = thecsvfile & column.HeaderText & ","
            Next
            'trim the last comma
            thecsvfile = thecsvfile.TrimEnd(",")
            'Add the line to the output
            thecsvfile = thecsvfile & vbCr & vbLf

```

```

'get the rows
For Each row As DataGridViewRow In DataGridView1.Rows
    'get the cells
    For Each cell As DataGridViewCell In row.Cells
        thecsvfile = thecsvfile &
            cell.FormattedValue.replace(", ", "") & ","
    Next
    'trim the last comma
    thecsvfile = thecsvfile.TrimEnd(",")
    'Add the line to the output
    thecsvfile = thecsvfile & vbCr & vbLf
Next
'write the file
thecsvfile = thecsvfile.Substring(0, thecsvfile.Length - 2)

Dim exePath As String = Application.StartupPath()
Dim fileName As String = exePath + "\DATI.csv"

My.Computer.FileSystem.WriteAllText(exePath + "\DATI.csv",
    thecsvfile, False)
MessageBox.Show("Il file è stato salvato!")
End Sub

Private Sub Button4_Click(sender As Object, e As EventArgs)
    Handles Button4.Click
    Me.Close()
End Sub

Private Sub Button5_Click(sender As Object, e As EventArgs)
    Handles Button5.Click
    SerialPort1.WriteLine("s")
End Sub
End Class

```

A.2 Addestramento dei modelli

A.2.1 Gradient boosting algorithm (Python)

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics

gb = GradientBoostingClassifier(learning_rate=0.2,
                               n_estimators=500,max_depth=3)
gb.fit(X_train, y_train)
predictions = gb.predict(X_test)

print("Confusion Matrix:")
print(confusion_matrix(y_test, predictions))
print()
print("Classification Report")
print(classification_report(y_test, predictions))
```

A.2.2 Multi-Layer Perceptron (Python)

```
from sklearn.neural_network import MLPClassifier
from sklearn import metrics

mlp = MLPClassifier(hidden_layer_sizes=(40, 40, 40, 80, 80, 80,
                                         40, 40, 40))
mlp.fit(X_train, y_train)
predictions = mlp.predict(X_test)

print(mlp)
print("Confusion Matrix:")
print(confusion_matrix(y_test, predictions))
print()
print("Classification Report")
print(classification_report(y_test, predictions))
```

A.2.3 Residual Neural Network (Python)

```
def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):

    conv = Conv2D(num_filters,
                 kernel_size=kernel_size,
                 strides=strides,
                 padding='same',
                 kernel_initializer='he_normal',
                 kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
    else:
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
        x = conv(x)
    return x
```

```

def resnet(input_shape, depth, num_classes=10): #per
    l'addestramento della resnet su cestino e chatbot abbiamo messo
    4
    # Definiamo il modello.
    num_filters = 16
    num_res_blocks = int((depth - 2) / 6)

    inputs = Input(shape=input_shape)
    x = resnet_layer(inputs=inputs)
    # blocco dei residui
    for stack in range(3):
        for res_block in range(num_res_blocks):
            strides = 1
            if stack > 0 and res_block == 0: # primo layer
                strides = 2
            y = resnet_layer(inputs=x,
                              num_filters=num_filters,
                              strides=strides)

            y = resnet_layer(inputs=y,
                              num_filters=num_filters,
                              activation=None)

            if stack > 0 and res_block == 0: # primo layer
                x = resnet_layer(inputs=x,
                                  num_filters=num_filters,
                                  kernel_size=1,
                                  strides=strides,
                                  activation=None,
                                  batch_normalization=False)

            x = keras.layers.add([x, y])
            x = Activation('relu')(x)
            num_filters *= 2

    # aggiungiamo classifier
    x = AveragePooling2D(pool_size=8)(x)
    y = Flatten()(x)
    outputs = Dense(num_classes,
                   activation='softmax',

```

```
        kernel_initializer='he_normal')(y)

    # restituisce il modello costruito.
    model = Model(inputs=inputs, outputs=outputs)
    return model
```

A.3 Applicazioni

A.3.1 Modello di Ensemble Lening (Python)

```
num(0).
num(s(X)) :- num(X).
```

A.3.2 Software per differenziare (VB.NET)

Form3

```
num(0).
num(s(X)) :- num(X).
```

A.3.3 Bot Telegram per differenziare (Python)

```
num(0).
num(s(X)) :- num(X).
```

Bibliografia

- [1] Duccio Bianchi (2019), L'Economia Circolare in Italia: la filiera del riciclo asse portante di un'economia senza rifiuti, Edizioni Ambiente.
- [2] Mancini Giovanna (2019), Economia circolare: il riciclo del legno vale 1,4 miliardi e 6mila posti di lavoro, Il Sole 24 Ore.
- [3] Simon Munzert - Christian Rubba - Peter Meibner - Dominic Nyhuis (2014), Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining, John Wiley & Sons Inc.
- [4] Paolo Guidi (2018), Elementi di Domotica, Zanichelli.
- [5] Paolo Guidi (2017), Fondamenti di robotica, Zanichelli.
- [6] Luigi Lo Russo - Elena Bianchi (2016), Arduino, Hoepli.
- [7] Paolo Di Leo (2017), Sensori & Arduino, Libri Sandit.
- [8] Adelchi Azzalini - Bruno Scarpa (2004), Analisi dei dati e Data Mining, Springer.
- [9] Yoav Freund - Robert E. Schapire (1999), A Short Introduction to Boosting, Journal of Japanese Society for Artificial Intelligence.
- [10] Kaiming He - Xiangyu Zhang - Shaoqing Ren - Jian Sun (2015), Deep Residual Learning for Image Recognition, Microsoft Research.
- [11] Christian Szegedy - Vincent Vanhoucke - Sergey Ioffe - Jonathon Shlens - Zbigniew Wojna (2015), Rethinking the Inception Architecture for Computer Vision, Google inc. - University College London.

- [12] Kaiming He - Xiangyu Zhang - Shaoqing Ren - Jian Sun (2016), Identity Mappings in Deep Residual Networks, Microsoft Research.
- [13] Sergey Zagoruyko - Nikos Komodakis (2017), Wide Residual Networks, Université Paris-Est.
- [14] Saining Xie - Ross Girshick - Piotr Dollàr - Zhuowen Tu - Kaiming He (2017), Aggregated Residual Transformations for Deep Neural Networks, UC San Diego - Facebook AI Research.
- [15] Furong Huang - Jordan T. Ash - John Langford - Robert E. Schapire (2018), Learning Deep ResNet Blocks Sequentially using Boosting Theory, University of Maryland - Princeton University - Microsoft Research.
- [16] Kazuki Osawa - Yohei Tsuji - Yuichiro Uen - Akira Naruse - Rio Yokota - Satoshi Matsuoka (2019), Large-Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks, Tokyo Institute of Technology - NVIDIA - RIKEN.
- [17] Gareth James - Daniela Witten - Trevor Hastie - Robert Tibshirani (2017), An Introduction to Statistical Learning, Springer.
- [18] Willi Richert - Luis Pedro Coelho (2013), Building Machine Learning Systems with Python, Packt Publishing.
- [19] Ethem Alpaydin (2010), Introduction to Machine Learning, Massachusetts Institute of Technology.
- [20] Shai Shalev-Shwartz - Shai Ben-David (2014), Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press.