

Matr. 08419

UNIVERSITÀ
CAMPUS BIO-MEDICO DI ROMA

Facoltà di ingegneria
Corso di laurea triennale in
Ingegneria Industriale

Sviluppo di un sistema di
individuazione degli interventi
rilevanti per il Corpo Nazionale dei
Vigili del Fuoco

Computer Systems
&
Bioinformatics Laboratory

Relatore

Prof. Paolo Soda

Correlatore

Ing. Rosa Sicilia

Correlatore

Ing. Marcello Esposito

Laureando
Matteo Tortora

ANNO ACCADEMICO 2016-2017

Alla mia famiglia

Indice

Elenco delle figure	6
Elenco delle tabelle	7
Abstract	8
Sommario	10
1 Introduzione	13
1.1 Problema	13
1.2 Obiettivi	14
1.3 Organizzazione della tesi	14
2 Stato dell'arte	16
2.1 Dataset: nozioni e terminologia generali	18
2.1.1 Tipi di features	20
2.2 Tipi di apprendimento automatico	21
2.3 Tipi di compiti	22
2.3.1 Classificazione	22
2.3.2 Regressione	24
2.3.3 Clustering	24
2.4 Metriche di valutazione	25
2.5 Principal component analysis, PCA	27
2.6 Feature extraction nell'elaborazione delle immagini	32
2.6.1 Local binary pattern	32
2.6.2 Gray Level Co-occurrence Matrix	38
2.7 Algoritmi di Machine Learning utilizzati	42
2.7.1 Alberi decisionali	43
2.7.2 Random Forest	47

3	Materiali	50
3.1	Dati disponibili	50
3.1.1	Database interventi	50
3.1.2	Database sms	51
3.1.3	Istat	53
3.2	Assemblaggio Dataset	54
4	Classificazione immagini satellitari	57
4.1	Raccolta dati	57
4.2	Pre-elaborazione	58
4.2.1	Feature extraction	58
4.3	Addestramento	62
4.4	Valutazione	63
5	Metodi	65
5.1	Statistiche dataset	65
5.2	Modello	68
6	Risultati	71
7	Conclusioni	76
	Bibliografia	87
	Indice analitico	88

Elenco delle figure

2.1	Algoritmo per definire evento rilevante	17
2.2	Addestramento di un sistema di apprendimento automatico.	21
2.3	Schema di apprendimento per rinforzo	22
2.4	Compito di classificazione.	23
2.5	Analisi cluster	25
2.6	Direzioni di massima varianza	28
2.7	Proiezione campioni sul versore \mathbf{u}	30
2.8	Operatore LBP versione originale base	33
2.9	Operatore LBP generalizzato	34
2.10	Esempio di un pattern uniforme e non uniforme	35
2.11	Diverse texture primitive individuate dall'operatore $LBP_{P,R}^{u2}$. .	36
2.12	Immagine satellitare di Firenze (dati mappa: Google, CNE-S/Airbus, DigitalGlobe)	38
2.13	Iistogramma delle occorrenze dei pattern LBP	39
2.14	Esempio di albero decisionale binario ($B=2$)	43
2.15	Tre misure di impurità in funzione della probabilità di una delle classi nel caso binario.	47
4.1	Tipi di zone possibili	58
4.2	Schema a blocchi del procedimento di classificazione delle immagini satellitari	59
4.3	Estrazione vettore feature	60
4.4	Matrice di confusione modello addestrato con dataset GLCM-2 .	63
5.1	Distribuzione degli interventi in base alla tipologia dell'operazione	66
5.2	Distribuzione degli interventi in base al momento del giorno . .	66
5.3	Distribuzione degli interventi in base al primo tipo di mezzo viene inviato	67
5.4	Distribuzione nello spazio delle feature numeriche	68
5.5	Pipeline finale del nostro modello	69

5.6	Convalida Leave-One-Out	70
6.1	Matrice di confusione del nostro modello	72
6.2	Curva ROC del nostro modello	73
6.3	Distribuzione della matrice di confusione per tipo d'intervento. TN: True Negative; TP: True Positive; FP: False Positive; FN: False Negative.	73
6.4	Distribuzione delle matrice di confusione per tipo di mezzo. TN: True Negative; TP: True Positive; FP: False Positive; FN: False Negative.	74

Elenco delle tabelle

2.1	Tipologia di attributi	20
2.2	Matrice di confusione per una classificazione binaria	25
2.3	Metriche per una classificazione binaria	27
2.4	Feature introdotte da Haralick	42
3.1	Riepilogo database interventi	51
3.2	Esempio di aggiunta feature <i>prov.</i>	53
3.3	Resoconto database degli SMS	53
3.4	Descrizione attributi database ISTAT utilizzato nel presente lavoro	54
4.1	Riepilogo dei dataset ricavati dalle immagini satellitari	61
4.2	Performance dei diversi classificatori. Acc.: Accuracy; Rec.: Recall; Pre.: Precision	62
4.3	Performance dei classificatori basati sui descrittori di Haralick .	64
5.1	Descrizione dataset	65
6.1	Performance dei vari classificatori. DO: dataset originale quindi senza SMOTE e CBFS; DFS: dataset con feature selection CBFS; DS: dataset con SMOTE; DSC: dataset con SMOTE e CBFS.	71
6.2	Performance dei vari classificatori senza considerare l'attributo <i>tipozona</i> . DO: dataset originale quindi senza SMOTE e CBFS; DFS: dataset con feature selection CBFS; DS: dataset con SMOTE; DSC: dataset con SMOTE e CBFS.	72

Abstract

This thesis presents a system to automatically detect relevant interventions of the Italian National Fire Corps (*Corpo nazionale dei vigili del fuoco*). This is the institutional agency for fire and rescue service and it is part of the Ministry of the Interior, Department of Firefighters, Public Rescue and Civil Protection (*Dipartimento dei Vigili del Fuoco, del Soccorso Pubblico e della Difesa Civile*). This system will be useful to firefighters to automatically report relevant interventions to competent authorities.

To this aim, the national firefighters Corps provided us with two databases. The first one contains a complete description of the operations that took place in the provinces of Milan, Naples, Rome and Turin during the 2016, and in the case of Rome's province the dataset also has the interventions from January 2017 up to June 2017. The second dataset contains the SMS messages sent to report a relevant intervention. This information was used in the labelling process to mark the samples of the first database as *relevant* or *not relevant*.

We built a new representation of our samples extracting several features, such as the number of updates and the overall duration of an operation, the statistical description of the geographic characteristics of the territory (from Istat databases) and information about the venue where the intervention took place. For this latter attribute we built a satellite image classifier able to distinguish between *agricultural*, *industrial* and *urban area*. This last feature was one of the most important descriptors selected by CBFS method, which computes the clearness of features expressing separability among classes in a feature.

With this best set of descriptors, we trained different supervised Machine Learning algorithms to predict the label of the interventions, choosing the best suited for our problem. The results show that the proposed approach can detect relevant interventions, but the error rate achieved as well as the consequent results analysis suggest that the real nature of the problem and the categorical features engineered could affect the recognition rate. The results and the specific knowledge collected on the issues suggest that further investigations

can be directed towards models based on finite-state machines.

Sommario

Questa tesi presenta un sistema che permette il rilevamento automatico degli interventi rilevanti per il Corpo Nazionale dei Vigili del Fuoco, una struttura dello Stato ad ordinamento civile, incardinata nel Ministero dell'interno Dipartimento dei vigili del fuoco, del soccorso pubblico e della difesa civile, per mezzo del quale il Ministero dell'interno assicura, anche per la difesa civile, il servizio di soccorso pubblico e di prevenzione ed estinzione degli incendi su tutto il territorio nazionale. Questo sistema permetterà ai vigili del fuoco di poter segnalare, in maniera automatica, gli interventi rilevanti alle autorità competenti.

A tale proposito, il corpo nazionale dei vigili del fuoco ci ha fornito due database. Il primo contiene una descrizione completa delle operazioni svolte nella provincia di Milano, Napoli, Roma e Torino durante il 2016, e nella provincia di Roma nel periodo Gennaio-Giugno 2017. Il secondo dataset contiene i messaggi SMS inviati per segnalare un intervento rilevante, ed è stato utilizzato nella fase di etichettatura per marcire i campioni del primo database come *rilevanti* oppure come *non rilevanti*.

Nel lavoro di tesi abbiamo costruito una nuova rappresentazione dei nostri campioni estraendo un certo numero di feature, come il numero di aggiornamenti e la durata complessiva di un'operazione, la descrizione statistica delle caratteristiche del territorio (da un database dell'Istat) e la sede dove ha avuto luogo l'intervento. Per determinare il valore di quest'ultimo attributo abbiamo costruito un classificatore di immagini satellitari capace di distinguere tra *zona agricola*, *zona industriale* e *zona urbana*. Quest'ultima feature è risultata essere uno dei descrittori più importanti selezionati dal metodo CBFS, il quale indica la capacità delle feature nel separare le classi. Con il miglior set di feature selezionate, abbiamo addestrato differenti algoritmi di Machine Learning utilizzati per predire l'etichetta degli interventi, ed infine è stato scelto quello che meglio si adatta al nostro problema.

I risultati mostrano che il nostro approccio riesca a rilevare gli interventi rile-

vanti, ma il tasso di errore raggiunto, così come l'analisi successiva dei risultati, suggerisce che la vera natura del problema e le feature categoriche utilizzate potrebbero influenzare il tasso di riconoscimento. I risultati e le conoscenze raccolte sul problema suggeriscono che ulteriori lavori potrebbero essere indirizzati verso modelli basati sulle macchine a stati-finiti.

Capitolo 1

Introduzione

Questa lavoro di tesi ha avuto luogo grazie alla collaborazione con il Corpo Nazionale dei Vigili del Fuoco il quale presentava un'esigenza urgente da risolvere. Il Centro Operativo Nazionale dei Vigili del Fuoco ci ha fornito due database che abbiamo utilizzato successivamente per sviluppare il nostro progetto. Nella prima parte del capitolo mostreremo e spiegheremo il problema che ha dato vita al nostro lavoro, successivamente passeremo ad illustrate i nostri obiettivi, infine nell'ultima parte del capitolo presenteremo l'organizzazione della tesi.

1.1 Problema

Il lavoro di questa tesi nasce da un'esigenza reale ed attuale che riguarda il Corpo Nazionale dei Vigili del Fuoco. Per i vigili del fuoco risulta essenziale poter etichettare in modo automatico e tempestivo un intervento come *rilevante* oppure come *non rilevante* in modo da poter avvertire le autorità competenti e gestire al meglio tale evento. In base alla descrizione dataci dai vigili del fuoco, gli interventi che possono essere definiti *rilevanti* sono principalmente quelli che rappresentano un pericolo per la collettività e generano interesse da parte dei mass media. Quindi riuscire a cogliere tempestivamente la rilevanza di un evento può evitare sia problemi burocratici che politici.

Attualmente, per risolvere questo problema i vigili del fuoco utilizzano un sistema deterministico basato su un vademecum che cerca di cogliere tutte le possibili caratteristiche che deve avere un intervento per essere classificato come *rilevante*. Il problema principale di questo approccio è che il sistema porta ad un'eccessiva generazione di falsi positivi. Quindi, la principale problematica che si trovarono ad affrontare fu riguardo la definizione stessa di intervento rilevante, cioè su quali caratteristiche doveva avere un intervento per essere

definito *rilevante*.

La gestione e la segnalazione degli interventi è un problema di interesse ancora aperto in cui il Corpo Nazionale dei Vigili del Fuoco investe risorse e tempo per lo sviluppo di servizi. Un esempio di servizio sviluppato quest'anno è rappresentato dall'App EasyAlert, rilasciata per dispositivi Android e iOS su iniziativa della Protezione Civile della Regione Calabria. Grazie a tale app, ogni cittadino sarà in grado di indicare, in tempo reale, il verificarsi di un evento calamitoso, con la possibilità di segnalarlo attraverso l'invio di foto e l'utilizzo di strumenti efficaci in modo da garantire un intervento tempestivo e localizzato. Il problema maggiore di questo sistema è che dipende dalla volontà e dalla capacità di un utente non esperto nel segnalare un intervento, tuttavia può però fornire dati utili che possono essere analizzati.

1.2 Obiettivi

L'obiettivo di questa tesi è quindi quello di sviluppare un sistema che permetta il rilevamento automatico degli interventi rilevanti per il Corpo Nazionale dei Vigili del Fuoco. Questo sistema permetterà ai vigili del fuoco di segnalare, in maniera automatica, gli interventi rilevanti alle autorità competenti, le quali potranno organizzare al meglio le risorse disponibili con lo scopo di ottimizzare la gestione di un intervento.

Dopo una profonda analisi dei dati a nostra disposizione e del genere di task che dobbiamo risolvere, si è deciso di applicare un sistema basato su tecniche di Machine Learning.

1.3 Organizzazione della tesi

Viene fornita ora una breve descrizione dell'organizzazione di questa tesi.

Nel Capitolo 2 parleremo dello stato dell'arte. Inizieremo parlando del sistema che attualmente i vigili del fuoco utilizzano per discriminare un intervento *rilevante* da uno *non rilevante*. Successivamente mostreremo una breve panoramica del Machine Learning e della sua terminologia e discuteremo anche di qualche nozione generale. Infine parleremo dei vari approcci e algoritmi che utilizzeremo successivamente per sviluppare il nostro sistema.

Nel Capitolo 3 discuteremo i dati forniteci dai vigili del fuoco e di quelli che abbiamo calcolato. In questo capitolo parleremo anche delle feature estratte e che successivamente serviranno ad addestrare il nostro modello.

Nel Capitolo 4 discuteremo dell'algoritmo di classificazione delle immagini satellitari sviluppato da noi e che servirà a fornire una descrizione approfondita del luogo in cui avviene un intervento.

Nel Capitolo 5 discuteremo del nostro modello finale e dei vari approcci che hanno permesso il suo sviluppo.

Nel Capitolo 6 valuteremo il nostro modello e parleremo dei risultati finali raggiunti. Infine nel Capitolo 7 trarremo le nostre conclusioni e ipotizzeremo lavori futuri riguardanti il progetto.

Capitolo 2

Stato dell'arte

Il nostro lavoro per il Corpo Nazionale dei Vigili del Fuoco risulta essere originale e nuovo nel suo genere, dal momento che non è mai stato affrontato prima e non esistono lavori a riguardo. Quindi, durante le nostre scelte per costruire il modello, non è stato possibile fare riferimento ad uno stato dell'arte dal momento che quest'ultimo non esiste per questo tipo di problema. Di conseguenza il nostro progetto aprirà un nuovo filone di ricerche e rappresenterà il primo modello nello stato dell'arte relativo.

Il nostro lavoro si basa sulla definizione ben posta di quando un intervento è inteso essere *rilevante*. Innanzitutto si è partiti con la distinzione fra evento *interessante* ed evento *rilevante*, per esempio quasi tutti gli incendi rappresentano un evento interessante ma non tutti sono intesi come rilevanti, quindi la classe degli eventi rilevanti è inclusa nella classe degli eventi interessanti. L'anno scorso il centro operativo nazionale ha rilasciato un vademecum nel quale si cerca di definire quando un evento può essere classificato come rilevante, quindi, supponendo che il vademecum non permetta falsi negativi, solo gli eventi che ricadono nei punti dettati dal vademecum vanno intesi come rilevanti.

Al momento il Corpo Nazionale dei Vigili del Fuoco procede nel seguente modo: quando si verifica un certo evento viene creata la scheda relativa all'intervento e se tale evento rispecchia determinate condizioni il Centro Operativo Nazionale dei vigili del fuoco (CON-VVF) viene avvertito dell'operazione automaticamente. Inizialmente come condizione affinchè un evento fosse considerato rilevante si era imposto l'utilizzo di 4 mezzi di soccorso, ma questa condizione generava troppi falsi positivi, quindi si è passati all'utilizzo di due condizioni legate dall'operatore logico **OR**. Tali condizioni sono l'utilizzo di 6 mezzi o la presenza, nella scheda dell'intervento, di almeno un termine contenuto in un vocabolario composto da determinate parole che si sono rivelate essere le più

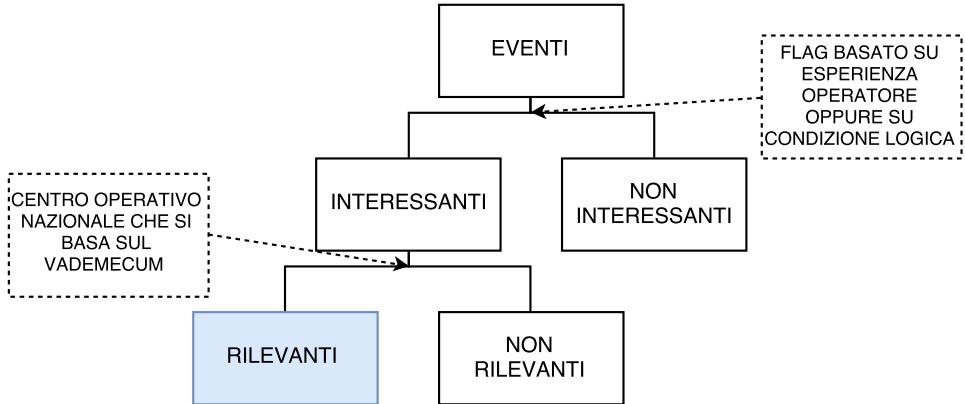


Figura 2.1: Algoritmo per definire evento rilevante

associate ad interventi rilevanti, come ad esempio *esplosione, stazione, presidente, SAF, elicotteristi, bomba*, ecc. Alternativamente gli operatori che si sono occupati dell'intervento, grazie alla loro esperienza pregressa, possono flaggare un evento come interessante. Un volta ricevuta la segnalazione di un evento interessante, il Centro Operativo Nazionale dei vigili del fuoco lo analizza sulla base del vademecum e nel caso rispecchia qualche condizione contenuta in quest'ultimo informa le autorità competenti tramite un messaggio sms. L'attuale modo di procedere è raffigurato in Figura 2.1. Il Centro Operativo Nazionale per ogni evento classificato come rilevante informa le autorità competenti mandando un messaggio sms a circa 100 persone. Nello scorso anno sono stati mandati circa 200 mila messaggi sms, corrispondenti a circa 2000 interventi rilevanti, approssimativamente 4/5 al giorno.

Va tenuto presente che se il Centro Operativo etichetta un intervento come non rilevante questo continua a rimanere comunque un evento interessante, quindi il Centro deve continuare a monitorare dal momento che può passare da non rilevante a rilevante se la situazione peggiora; ogni aggiornamento dell'intervento da parte della squadra incaricata dell'evento richiede una nuova analisi da parte del Centro Operativo. A conclusione di un intervento la scheda relativa viene chiusa e se l'intervento era stato classificato come rilevante un messaggio di chiusura viene mandato alle autorità competenti.

Secondo l'attuale modo di procedere è estremamente importante sia la capacità dell'operatore umano di rilevare il grado di importanza di un intervento, grazie alla sua esperienza maturata nel corso degli anni, sia la ben dichiarazione del vademecum. Il centro operativo attualmente ipotizza che il vademecum riesca a raccogliere più del 95% degli interventi rilevanti. Il dataset a nostra disposizione sarà quindi intrinsecamente composto da campioni che contengono una

forte soggettività umana e basati su un vademecum non privo di errori.

Il nostro compito sarà quello di sviluppare un sistema automatico che sia capace di apprendere dai dati e di etichettare gli interventi, flaggati come interessanti, come *rilevanti* oppure come *non rilevanti*, il nostro algoritmo dovrà quindi sostituirsi al lavoro svolto dal Centro Operativo Nazionale nell'ultima parte del sistema a cascata raffigurato in Figura 2.1.

A tale scopo si è deciso di applicare le tecniche del *Machine Learning*. A partire dalla seconda metà del ventesimo secolo, il Machine Learning si è evoluto come una branca dell'*intelligenza artificiale* e prevede lo sviluppo di algoritmi di autoapprendimento, *apprendimento automatico*, in grado di acquisire conoscenze a partire dai dati, con lo scopo di effettuare delle previsioni. Il termine *Machine Learning* è stato coniato nel 1959 da Arthur Samuel [19] che voleva indicare l'abilità dei computer di imparare senza che fossero stati esplicitamente programmati per farlo.

In questo capitolo illustreremo quindi alcuni concetti di base del Machine Learning e spiegheremo alcuni degli algoritmi e delle tecniche che vedremo poi applicate nel presente lavoro di tesi.

2.1 Dataset: nozioni e terminologia generali

Prima di passare a descrivere i compiti tipici del Machine Learning descriviamo brevemente l'elemento su cui opera questa disciplina.

Il *dataset* è l'elemento fondamentale nell'apprendimento automatico, utile nel rappresentare e dare forma ai dati ed essenziale nella fase di addestramento di un modello.

Un dataset, in base al contesto in cui è impiegato, può assumere diversi nomi:

Training set il *Training set* è il dataset utilizzato nella fase di apprendimento e ci permette di determinare il valore dei parametri del nostro modello.

Validation set Il *Validation set*, chiamato in italiano *set di convalida*, è il dataset, utilizzato in fase di addestramento, per convalidare il nostro modello.

Test set Il *Test set* è il dataset utilizzato per testare le performance finali del nostro modello dopo la fase di addestramento. È essenziale che il test set e il validation set non contengano campioni utilizzati per l'addestramento dei

parametri del modello. Questo risulterebbe infatti essere un errore metodologico conosciuto come ‘*test con il set di addestramento*’ [10].

Per rappresentare un dataset faremo uso della notazione matriciale. Un dataset è formato da n *campioni*, chiamati anche *istanze* od *osservazioni*, che formano le righe del dataset e sono descritti da m *attributi* (*feature*, *caratteristiche*, *dimensioni*, *misurazioni*) che formano le colonne del dataset. Le feature rappresentano delle misurazioni grazie alle quali andiamo a descrivere il nostro campione. Dunque supponiamo di avere un dataset $\{\mathbf{x}^{(i)}; i = 1, \dots, n\}$, composto da n campioni, dove ogni istanza $\mathbf{x}^{(i)} \in \Re^m$ può essere considerata come un vettore $\{\mathbf{x}_j^{(i)}; j = 1, \dots, m\}$ giacente su uno spazio m -dimensionale. Per quei dataset di cui si ha una conoscenza pregressa della classe dei campioni si ha anche un vettore delle classi $\mathbf{y} \in \Re^n$ definito come:

$$\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}$$

Per esempio, il dataset Iris [7] che è formato da 150 istanze e 4 attributi $\{\text{Sepal length}, \text{Sepal width}, \text{Petal length}, \text{Petal width}\}$ può essere rappresentato come una matrice $\mathbf{X} \in \Re^{n \times m}$:

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{pmatrix}$$

Con un vettore delle classi del tipo:

$$\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(150)} \end{pmatrix}, \quad (y^{(i)} \in \{\text{Setosa}, \text{Versicolor}, \text{Virginica}\}; i = 1, \dots, n)$$

Da questo momento in poi utilizzeremo le lettere minuscole in grassetto per far riferimento ai vettori ($\mathbf{x} \in \Re^{n \times 1}$) e le lettere maiuscole in grassetto per far riferimento alle matrici ($\mathbf{X} \in \Re^{n \times m}$). Per riferirsi ad un generico campione

utilizzeremo la seguente notazione $\mathbf{x}^{(i)}$ e per far riferimento al singolo elemento di un vettore utilizzeremo le lettere in corsivo ($x_j^{(i)}$).

2.1.1 Tipi di features

Gli attributi di un dataset possono essere principalmente di due tipi, *numerici* oppure *categorici*.

Gli attributi numerici sono dati di tipo quantitativi e possono essere divisi in due tipi, *discreti* o *continui*. Esempi di dati discreti possono essere l'età di una persona oppure un voto scolastico, invece esempi di dati continui possono essere la temperatura in una stanza, la distanza fra due oggetti o anche il valore di un'azione bancaria.

Quando si parla di dati categorici intendiamo variabili di tipo qualitativo e li si deve ulteriormente distinguere fra dati *ordinali* e *nominali*. Le caratteristiche ordinali sono dati categorici che possono essere ordinati, esempi di caratteristiche ordinali possono essere la taglia di una maglietta dove $XL > L > M > S > XS$ o anche una valutazione letterale dove *ottimo* > *distinto* > *buono* > *discreto*. Viceversa le caratteristiche nominali sono rappresentati da quei dati categorici che non ammettono un ordinamento, esempi tipici di dati nominali possono essere il colore degli occhi, il nome del gruppo sanguigno o il nome di un partito, infatti in questi casi non ha senso parlare di un ordinamento fra i dati.

Attributi numerici	
Continui	Operazioni ammissibili: $\{=; \neq; <; \leq; >; \geq; +; -; \times; \div\}$ Esempi: Temperatura, distanza, valore azione bancaria
Discreti	Operazioni ammissibili: $\{=; \neq; <; \leq; >; \geq; +; -; \times\}$ Esempi: Età, votazione scolastica numerica
Attributi categorici	
Ordinali	Operazioni ammissibili: $\{=; \neq; <; \leq; >; \geq\}$ Esempi: Taglia abiti, valutazione scolastica letterale
Nominali	Operazioni ammissibili: $\{=; \neq\}$ Esempi: Colore, gruppo sanguigno, nome di un partito

Tabella 2.1: Tipologia di attributi

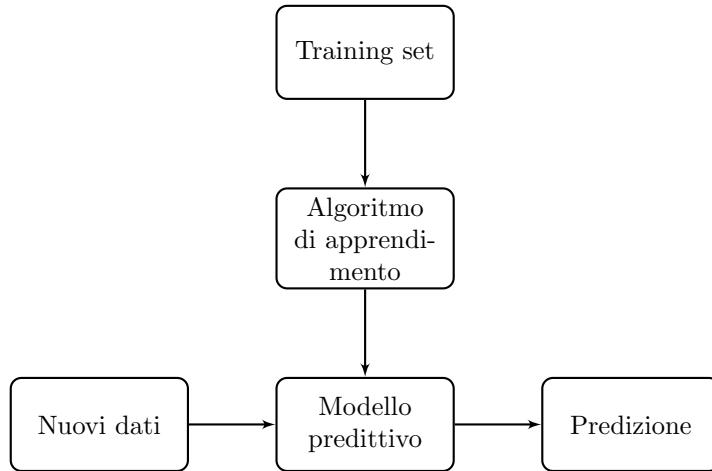


Figura 2.2: Addestramento di un sistema di apprendimento automatico.

2.2 Tipi di apprendimento automatico

L'apprendimento automatico, in base al tipo di addestramento, si può suddividere principalmente in tre categorie: *apprendimento supervisionato*, *apprendimento non supervisionato* e *apprendimento per rinforzo*.

Apprendimento supervisionato Nell'apprendimento supervisionato l'addestramento del modello è effettuato utilizzando un dataset di istanze pre-etichettate. Compiti tipici dell'apprendimento supervisionato sono la classificazione e la regressione. In Figura 2.2 è mostrato lo schema di un generico algoritmo di apprendimento automatico.

Apprendimento non supervisionato Nell'apprendimento non supervisionato non si effettua un addestramento a priori del modello dal momento che non si possiede una conoscenza preeressa dei campioni usati nell'apprendimento supervisionato per addestrare il modello. Non avendo dati etichettati per l'addestramento dell'algoritmo, tramite l'apprendimento non supervisionato si cerca di trovare la struttura incognita dei dati in ingresso al modello. Compito tipico dell'apprendimento non supervisionato è il clustering, come si è visto nella Sezione 2.3.3.

Apprendimento per rinforzo Nell'apprendimento per rinforzo si cerca di costruire un sistema, detto *agente*, le cui interazioni con l'ambiente dinamico che lo circonda hanno la massima *ricompensa* possibile. Questa si cerca

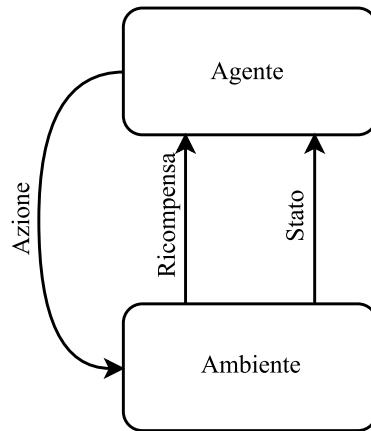


Figura 2.3: Schema di apprendimento per rinforzo

di ottenerla massimizzando la *funzione di ricompensa*, in altre parole si vuole ottimizzare l’interazione dell’agente con l’ambiente circostante. In Figura 2.3 vediamo un esempio di addestramento di un sistema di apprendimento per rinforzo.

L’apprendimento per rinforzo può essere considerato un apprendimento semi-supervisionato dal momento che non si ha in partenza un training set, etichettato, con cui addestrare il sistema, ma si ha a disposizione solamente un feedback che riporta se l’output è giusto o sbagliato; in altre parole il feedback rappresenta la misura della qualità con cui l’azione compiuta sull’ambiente è stata misurata dalla funzione ricompensa. Quindi un agente può migliorare le proprie prestazioni tramite un processo esplorativo del tipo *trial and error*. Esempi tipici di apprendimento per rinforzo si trovano nel gioco degli scacchi, dove un agente può migliorare le proprie prestazioni giocando contro un avversario, il feedback in questo caso è rappresentato dalla vittoria o dalla sconfitta dell’agente alla fine del gioco.

2.3 Tipi di compiti

In base al tipo di output atteso si possono individuare principalmente 3 tipi di compiti svolti dagli algoritmi di apprendimento automatico: *classificazione*, *regressione* e *clustering*.

2.3.1 Classificazione

La classificazione è un compito dell’apprendimento automatico nel quale si addestra un modello capace di assegnare una classe a delle nuove istanze, pri-

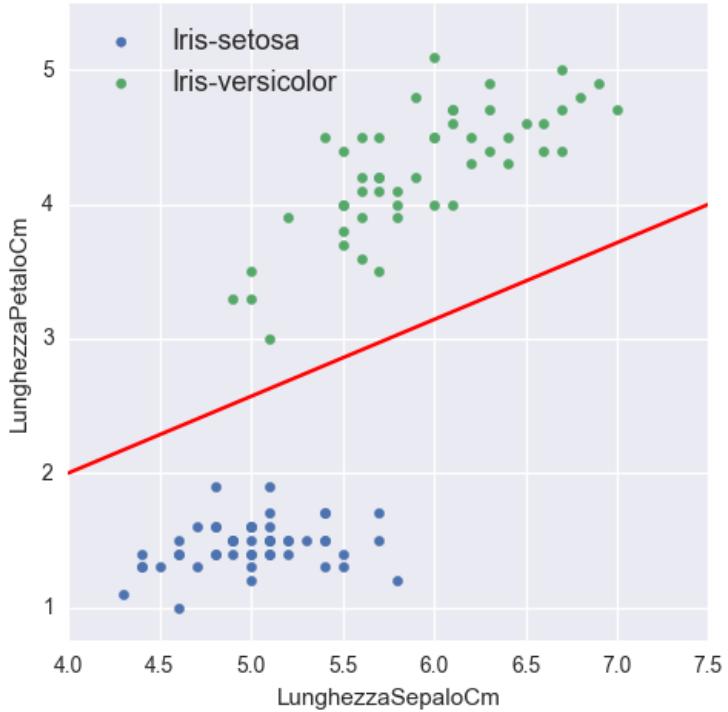


Figura 2.4: Compito di classificazione.

ve di etichetta, in ingresso al modello stesso. Nel caso della classificazione supervisionata l’addestramento del modello è effettuato tramite dei campioni etichettati.

In Figura 2.4 si ha un esempio di classificazione binaria effettuata sul dataset Iris [7]. Si nota in rosso il confine decisionale ricavato durante la fase di apprendimento che permette di dividere in modo lineare le due classi e di classificare i nuovi campioni in ingresso in una delle due.

Esempi tipici della classificazione si possono trovare nella rilevazione dei messaggi spam in base al contenuto di un'email oppure nella categorizzazione di un neo come neoplasia o no.

Tipi di classificazione

Considerando istanze con etichette categoriche, cioè di tipo qualitativo come il colore di una maglietta o la forma di un frutto, il compito di classificazione si può suddividere in 4 categorie [23]:

Binaria Il campione di input al sistema deve essere classificato in una sola classe C_1 fra due classi (C_1, C_2) disgiunte, $C_1 \cap C_2 = \emptyset$.

Il nostro problema di classificazione è di tipo binario, infatti avremo solamente

due classi disgiunte, cioè gli eventi rilevanti e gli eventi non rilevanti, in cui classificare i campioni in ingresso.

Multi-classe Il campione di input al sistema deve essere classificato in una sola classe C_i fra l classi (C_1, C_2, \dots, C_l) disgiunte, $C_i \cap C_j = \emptyset, \forall i \neq j$.

Un esempio di classificazione multi-classe è la classificazione in una delle tre specie di fiori nel dataset Iris [7].

Multi-etichetta Il campione di input al sistema può essere classificato in più classi contemporaneamente tra l classi (C_1, C_2, \dots, C_l) disgiunte, $C_i \cap C_j = \emptyset, \forall i \neq j$.

La classificazione multi-etichetta può essere applicata ad una varietà molto vasta di domini, dalla categorizzazione dei testi ([11],[20],[21]) dove un testo può appartenere a più generi, classi, alla volta, alla diagnosi medica [2] dove un paziente può soffrire simultaneamente di più malattie.

Gerarchica Il campione di input al sistema deve essere classificato in una sola classe C_i divisa in sottoclassi o raggruppata in superclassi, quindi le classi che devono essere predette sono raggruppate in modo gerarchico.

La categorizzazione dei testi e la bioinformatica forniscono molti esempi sull'uso della classificazione gerarchica che può essere utilizzata per esempio nella classificazione delle funzioni proteiche [6].

2.3.2 Regressione

La regressione è un compito dell'apprendimento supervisionato nel quale le variabili che si cercano di predire sono di tipo continuo e non di tipo discreto come nel caso della classificazione.

La regressione per esempio può essere utilizzata per prevedere il valore di un immobile o di un bene in base ad una conoscenza pregressa.

2.3.3 Clustering

Il clustering è un compito tipicamente non supervisionato che consente di organizzare le istanze in ingresso ad un sistema in gruppi significativi, detti *cluster*, senza avere una conoscenza a priori di tali gruppi.

Ogni cluster raggruppa un insieme di dati in base al grado di similarità, quindi due campioni che si trovano nello stesso cluster saranno più simili tra loro rispetto a due campioni che si trovano in due cluster diversi; quindi il clustering

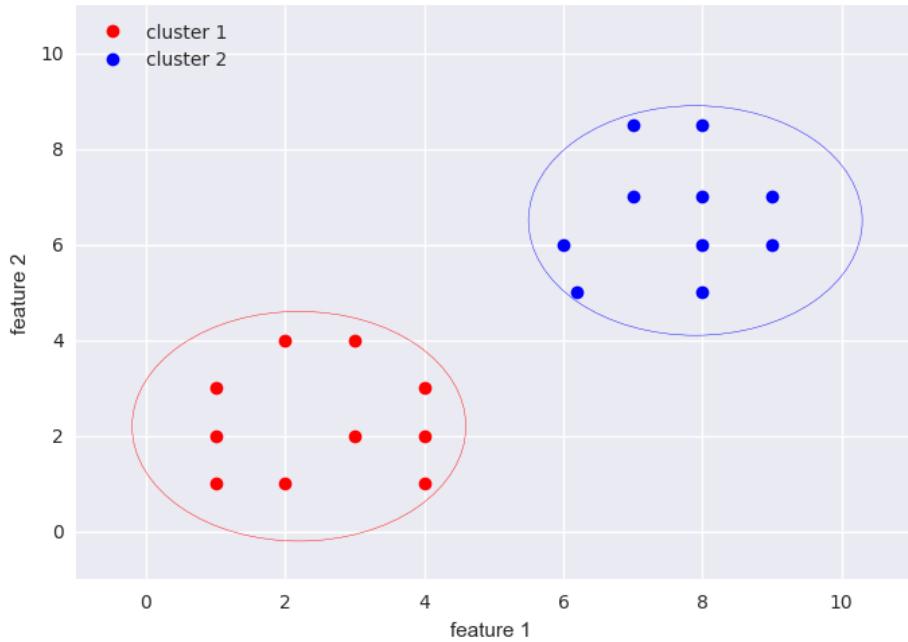


Figura 2.5: Analisi cluster

Classe	Classificati come <i>pos</i>	Classificati come <i>neg</i>
<i>pos</i>	veri positivi (<i>tp</i>)	falsi negativi (<i>fn</i>)
<i>neg</i>	falsi positivi (<i>fp</i>)	veri negativi (<i>tn</i>)

Tabella 2.2: Matrice di confusione per una classificazione binaria

è una tecnica utilizzata per la strutturazione dell'informazione.

Questa tecnica può essere utilizzata nello studio delle comunità, anche digitali, per il raggruppamento di individui, utenti che presentano comportamenti simili.

In Figura 2.5 si può notare un esempio di clustering grazie al quale i dati vengono suddivisi in due cluster in base ad un criterio di somiglianza.

2.4 Metriche di valutazione

In questa sezione discutiamo delle varie metriche utilizzate per la misurazione delle prestazioni di un modello.

Una rappresentazione delle performance di un classificatore è la *matrice di confusione*. In Tabella 2.2 è mostrata una matrice di confusione per una classificazione binaria. La matrice di confusione è una matrice quadrata in cui è riportato il numero dei campioni classificati correttamente come positivi (veri positivi), il numero dei campioni classificati erroneamente come positivi (falsi

positivi), il numero dei campioni classificati erroneamente come negativi (falsi negativi) ed infine il numero di campioni classificati correttamente come negativi (veri negativi).

L'*accuratezza*, accuracy, è la misura più comunemente utilizzata per quantificare le prestazioni di un modello in modo generale [23], tuttavia esistono altre metriche impiegate per valutare la rilevanza di un modello: la *precisione*, il *recall* e *F-score*.

Accuratezza L'accuratezza, come già detto, è la metrica più comunemente utilizzata per la valutazione delle prestazioni di un classificatore. Questa rappresenta il numero di campioni correttamente classificati rispetto al numero totale di istanze classificate:

$$ACC = \frac{tp + tn}{tp + fn + fp + tn}$$

Definito l'errore come la somma delle false previsioni divisa per il numero totale delle previsioni, lo si può calcolare facilmente a partire dall'accuratezza:

$$ERR = 1 - ACC = \frac{fp + fn}{tp + fn + fp + tn}$$

Richiamo Il richiamo, dall'inglese *Recall*, chiamato anche *TPR (true positive rate)* oppure *sensibilità*, rappresenta il numero di campioni classificati correttamente come positivi diviso il numero totale dei campioni positivi:

$$REC = \frac{tp}{tp + fn}$$

Precisione La precisione, detta anche *PPV (Positive predictive value)*, è la metrica che rappresenta il numero di campioni classificati correttamente come positivi diviso il numero dei campioni etichettati dal sistema come positivi:

$$PRE = \frac{tp}{tp + fp}$$

Il recall e la precisione sono metriche utilizzate ampiamente nella diagnosi medica, nel recupero delle informazioni (*Information retrieval*) e nel compito di text classification.

F-score L’F-score è una metrica che consiste nella combinazione di precisione e recall:

$$S = (1 + \beta^2) \frac{PRE \cdot REC}{\beta^2 \cdot PRE + REC} = \frac{(1 + \beta^2)tp}{(1 + \beta^2)tp + \beta^2 fn + fp}$$

in altre parole la l’F-score consiste in un media armonica pesata fra queste due metriche.

In Tabella 2.3 sono riassunte le proprietà delle metriche più largamente utilizzate per stimare la bontà di un modello nel compito di classificazione binaria.

2.5 Principal component analysis, PCA

Esistono due approcci per ridurre la dimensionalità di un dataset nella fase di pre-elaborazione, l’*estrazione delle caratteristiche*, dall’inglese *feature extraction*, e la *selezione delle caratteristiche*, dall’inglese *feature selection*. Motivo principale per il quale si vuole ridurre la dimensionalità di un dataset è quello di evitare il problema della *curse of dimensionality*, il quale si verifica quando si hanno dataset descritti da un numero elevato di feature. Aumentando la dimensionalità di un dataset la densità di questo nello spazio cartesiano diminuirà, questo comporta un peggioramento complessivo delle performance di un algoritmo di apprendimento. Per aumentare la densità dei campioni nello spazio si potrebbe aumentare il numero di essi ma quest’ultima operazione non è sempre possibile, quindi un’altra operazione sempre possibile consiste nel ridurre la dimensionalità del dataset.

L’*estrazione delle caratteristiche* raccoglie un insieme di metodiche che con-

Misura	Formula	Informazioni mostrate della metrica
Accuratezza	$\frac{tp+tn}{tp+fn+fp+tn}$	Rappresenta l’efficacia complessiva del classificatore
Recall	$\frac{tp}{tp+fn}$	Efficacia di un classificatore nell’identificare classi positive
Precisione	$\frac{tp}{tp+fp}$	Mette in risalto la concordanza delle etichette positive predette con la vera classe delle etichette
Fscore	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$	Rappresenta una combinazione fra Recall e Precisione

Tabella 2.3: Metriche per una classificazione binaria

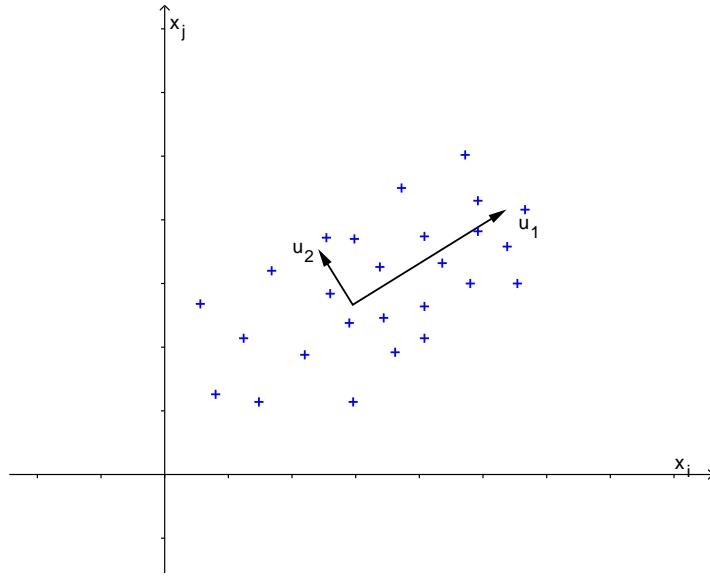


Figura 2.6: Direzioni di massima varianza

sentono di condensare il contenuto di un dataset proiettandolo in un nuovo sottospazio di dimensionalità, in genere, minore. Un approccio alternativo per la riduzione della dimensionalità consiste nella *selezione delle caratteristiche*, che a differenza del feature extraction, permette di mantenere le caratteristiche originali senza proiettarle in un nuovo spazio vettoriale.

L'*analisi delle componenti principali*, dall'inglese *principal component analysis (PCA)*, anche conosciuta come *trasformata di Karhunen-Loéve* è l'estrattore delle caratteristiche lineare più famoso [10] e opera una riduzione della dimensionalità in un sottospazio in modo lineare a prescindere dalle etichette dei campioni, quindi può essere considerato un algoritmo non supervisionato.

La principal component analysis fu formulata inizialmente da Karl Pearson nel 1901 per poi essere ripresa e sviluppata nella sua forma finale da Harold Hotelling nel 1933.

Supponiamo di avere un dataset $\mathbf{X} \in \Re^{n \times m}$ composto da n campioni $\{\mathbf{x}^{(i)}; i, \dots, n\}$ ognuno con m attributi, $\mathbf{x}^{(i)} \in \Re^m$. Siano le caratteristiche x_k e x_z , rispettivamente, la lunghezza espressa in centimetri e la lunghezza espressa in pollici. Questi due attributi sono altamente correlati dal momento che sono linearmente dipendenti a meno di una componente di rumore intrinseca nei dati. Quindi i dati, idealmente, giacciono su un sottospazio \Re^{m-1} a dimensionalità minore. Tramite la PCA andremo a togliere questa ridondanza nei dati.

L'analisi PCA, tramite uno studio della correlazione tra le caratteristiche, mira

a trovare le direzioni di massima varianza all'interno dei dati. In Figura 2.6 le componenti \mathbf{u}_1 e \mathbf{u}_2 rappresentano le *componenti principali* di massima varianza che formano la nuova base ortogonale che ci consentirà di mappare il generico campione $\mathbf{x}^{(i)}$ nel nuovo sottospazio.

Prima di spiegare in dettaglio l'algoritmo per la PCA dobbiamo pre-elaborare i dati normalizzandoli e portare sulla stessa scala le diverse caratteristiche. Quindi calcoliamo la media dei campioni come:

$$\mu = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \quad (2.1)$$

μ è un vettore colonna con m righe (stiamo considerando i generici vettori $\mathbf{x}^{(i)}$ come vettori colonna). Ora sostituiamo ogni vettore \mathbf{x}^i con $\mathbf{x}^{(i)} - \mu$ in questo modo ciascun vettore $x^{(i)}$ avrà media nulla. Ora calcoliamo la *varianza* di ciascuna caratteristica come:

$$\sigma_j^2 = \frac{1}{n} \sum_{i=1}^n (x_j^{(i)})^2 \quad (2.2)$$

e dividiamo ogni caratteristica per quest'ultima $\frac{x_j^{(i)}}{\sigma_j}$ (nota che σ_j^2 è uno scalare). Quest'ultimo passaggio può essere omesso se siamo sicuri che tutte le caratteristiche si trovano sulla stessa scala, ad esempio quando i campioni del dataset sono immagini in toni di grigio, quindi ogni caratteristica è composta da un pixel il cui valore è compreso nell'intervallo $\{0, \dots, 255\}$.

Conclusa la pre-elaborazione con la quale abbiamo standardizzato il dataset possiamo passare a trovare le direzioni di massima varianza. Per iniziare consideriamo il vettore unitario $\mathbf{u} \in \mathbb{R}^m$; il fatto che il vettore sia unitario, $\mathbf{u}^T \mathbf{u} = \mathbf{u} \cdot \mathbf{u} = 1$, non implica una perdita di generalità.

Proiettando i dati del nostro dataset sul versore \mathbf{u} otteniamo una situazione simile a quella riportata in Figura 2.7, dove il dataset di partenza è rappresentato in uno spazio 2-dimensionale e la proiezione è effettuata su uno sottospazio ad 1-dimensione. La distanza dall'origine della proiezione del nostro generico campione $\mathbf{x}^{(i)}$ su \mathbf{u} è pari al valore scalare $\mathbf{u}^T \mathbf{x}^{(i)}$. La varianza dei dati proiettati è pari a:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}^{(i)})^2 \quad (2.3)$$

Questa, scomponendo il quadrato può essere riscritta come:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}^{(i)}) (\mathbf{u}^T \mathbf{x}^{(i)}) \quad (2.4)$$

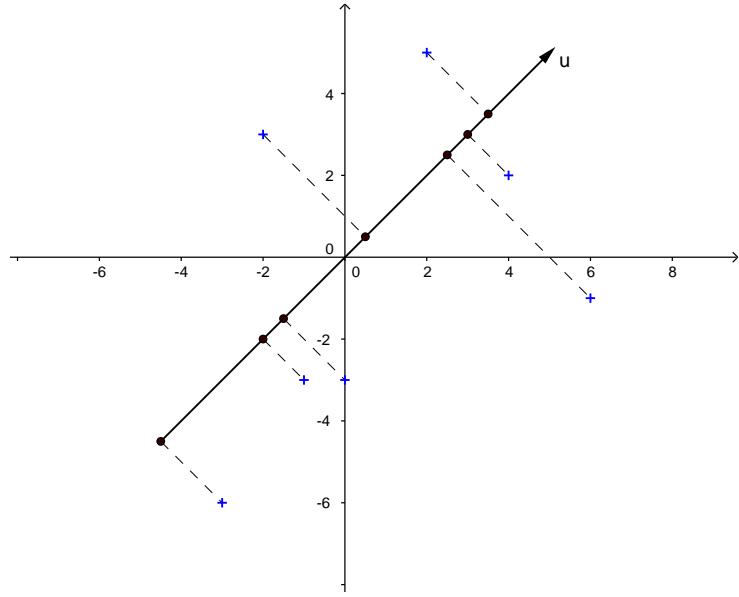


Figura 2.7: Proiezione campioni sul versore \mathbf{u}

e dal momento che $\mathbf{u}^T \mathbf{x}^{(i)} = (\mathbf{x}^{(i)})^T \mathbf{u}$ possiamo riscriverla come:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n \mathbf{u}^T \mathbf{x}^{(i)} \mathbf{x}^{(i)T} \mathbf{u} \quad (2.5)$$

infine, possiamo portare \mathbf{u} fuori dalla sommatoria, si ha:

$$\sigma^2 = \mathbf{u}^T \frac{1}{n} \left(\sum_{i=1}^n \mathbf{x}^{(i)} \mathbf{x}^{(i)T} \right) \mathbf{u} = \mathbf{u}^T \Sigma \mathbf{u} \quad (2.6)$$

con Σ si è indicata la *matrice di covarianza* dei campioni (è una matrice $m \times m$). Ora dobbiamo massimizzare la varianza $\sigma^2 = \mathbf{u}^T \Sigma \mathbf{u}$ rispetto a \mathbf{u} per trovare le direzioni di massima varianza. Questo si presenta come un problema di ottimizzazione vincolata, dove il vincolo è rappresentato dalla condizione $\mathbf{u}^T \mathbf{u} = 1$. Possiamo rimediare utilizzando il *teorema dei moltiplicatori di Lagrange*. Introducendo il *moltiplicatore di Lagrange* λ definiamo una nuova funzione:

$$\sigma^2 = \mathbf{u}^T \Sigma \mathbf{u} + \lambda(1 - \mathbf{u}^T \mathbf{u}) \quad (2.7)$$

e su questa effettueremo un'ottimizzazione non vincolata.

Derivando rispetto a \mathbf{u} ed uguagliando a zero otteniamo:

$$\Sigma \mathbf{u} - \lambda \mathbf{u} = 0 \quad (2.8)$$

e quindi:

$$\Sigma \mathbf{u} = \lambda \mathbf{u} \quad (2.9)$$

Da quest'ultima equazione apprendiamo che \mathbf{u} è *autovettore* di Σ . Moltiplicando a sinistra per \mathbf{u}^T e ricordandosi la relazione $\mathbf{u}^T \mathbf{u} = 1$ otteniamo:

$$\sigma^2 = \mathbf{u}^T \Sigma \mathbf{u} = \lambda \quad (2.10)$$

quindi la varianza è massima quando \mathbf{u} è uguale all'autovettore di Σ avente l'*autovalore* più grande.

Quindi risolvendo il problema agli autovalori:

$$\det(\Sigma - \lambda \mathbf{I}) \mathbf{u} = 0 \quad (2.11)$$

ci troveremo l'*autovalore* λ (il più grande fra gli autovalori) e il relativo autovettore \mathbf{u} .

Nel caso generale, se desideriamo proiettare i nostri dati in un sottospazio k -dimensionale ($k \leq m$) scegliamo $\mathbf{u}_1, \dots, \mathbf{u}_k$ che rappresentano i primi k autovettori della matrice di covarianza ordinati in ordine decrescente sul valore del relativo *autovalore*. Gli autovettori scelti sono chiamati anche *componenti principali*, da cui il nome Principal Component Analysis. Questi autovettori formeranno la nuova base ortogonale \mathbf{B} :

$$\mathbf{B} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \Re^{m \times k} \quad (2.12)$$

che servirà a proiettare i dati nel nuovo sottospazio k -dimensionale.

Per proiettare l'intero dataset nel nuovo sottospazio appena trovato dobbiamo solamente calcolare

$$\mathbf{X}' = \mathbf{XB}, \quad \mathbf{X}' \in \Re^{n \times k} \quad (2.13)$$

La Principal Component Analysis ha molte applicazioni possibili. Innanzitutto la PCA può essere effettuata per una compressione dei dati con l'obiettivo di conservare le informazioni più rilevanti contenute nei dati. Inoltre, se riuscissimo a portare i dati che si trovano in uno spazio ad alta dimensionalità ad un sottospazio 2/3-dimensionale potremmo anche visualizzarli. Un'altra applicazione standard della PCA nella fase di pre-elaborazione è nella riduzione di un dataset al fine di migliorare le prestazioni computazionali di un algoritmo di apprendimento automatico, di evitare problemi di *overfitting* e di proteggerci, come già detto, contro la *curse of dimensionality* [1]. Infine

un’altra applicazione della PCA è di rimuovere il rumore presente nel dataset.

2.6 Feature extraction nell’elaborazione delle immagini

Nell’analisi del problema di riconoscimento di eventi rilevanti, è stato necessario esplorare tecniche di machine learning applicate all’elaborazione delle immagini. Lo scopo di questo lavoro è di aggiungere al dataset finale un nuovo attributo, denominato *tipo_zona*, che rappresenta il tipo di zona in cui è avvenuto l’intervento. Nell’ambito dell’elaborazione delle immagini e in quello della *computer vision* la feature extraction raccoglie una serie di algoritmi capaci di rilevare e di estrarre particolari feature dalle immagini oppure dai video digitali.

In questa sezione parleremo di due algoritmi tipicamente utilizzati per l’estrazione di feature nei compiti di classificazioni di immagini, entrambi gli algoritmi utilizzano metodi statistici per l’analisi della tramatura dell’immagine, in inglese *texture*.

2.6.1 Local binary pattern

Il *local binary pattern* (*LBP*) è un operatore introdotto da Ojala nel 1996 [12] con il quale descrivere la tramatura di un’immagine trasformando l’immagine stessa in un array di etichette intere che descrivono gli intorni locali di un pixel. Queste etichette, chiamate anche statistiche dell’immagine, rappresentano le feature estratte. Questo approccio si basa sul fatto che la texture dell’immagine localmente presenta un proprio *pattern*. Un pattern è una struttura ricorrente all’interno di un’entità considerata, nel caso delle immagini digitali un pattern è la distribuzione ripetitiva dei valori assunti dai pixel all’interno dell’immagine stessa.

LBP di base

La versione originale dell’operatore LBP considera intorni 3×3 di un pixel come in Figura 2.8. Il valore di grigio del pixel centrale ha la funzione di valore di soglia per i pixel dell’intorno, ogni valore dell’intorno è sostituito con un 1 se il proprio valore è maggiore del valore di soglia del pixel centrale, con uno 0 se il suo valore è minore della soglia. In questo modo per ogni intorno otteniamo una stringa binaria che rappresenta l’etichetta associata all’intorno

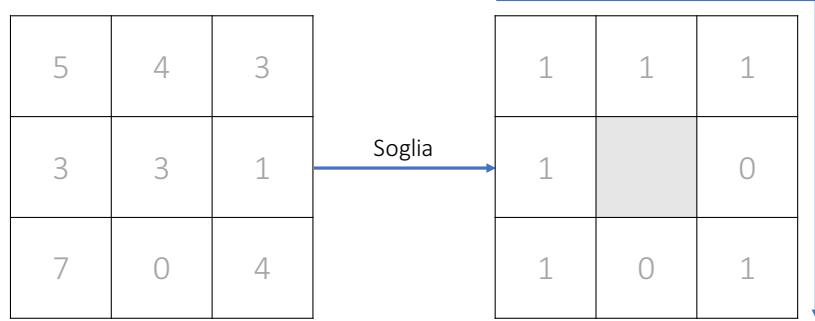


Figura 2.8: Operatore LBP versione originale base

stesso, per l'esempio in figura si ha:

$$LBP_2 = 11101011_2$$

Ecco spiegato il nome ‘*local binary pattern*’, un intorno locale è filtrato dal valore del livello di grigio del pixel centrale e rappresentato con un codice binario.

L'etichetta intera dell'intorno sarà la conversione in decimale della stringa binaria, nel nostro caso:

$$LBP_{10} = 235_{10}$$

l'etichetta ottenuta è invariante rispetto ai cambiamenti locali uniformi dei livelli di grigio e identifica univocamente il pattern a cui questa è associata.

LBP generalizzato

Nel 2002 Ojala [13] riprese e rielaborò il suo operatore generalizzandolo. Con questa nuova versione dell'operatore è possibile avere intorni di dimensioni diverse.

In questa versione l'intorno è formato da P punti disposti sul perimetro di un cerchio di raggio R centrato sul pixel di riferimento centrale. Ogni intorno quindi sarà definito da un valore di P ed R e lo si indicherà con la seguente notazione: (P, R) . Se i punti che si trovano sulla circonferenza non ricadono su un unico pixel il loro valore sarà determinato da un interpolazione bilineare. In Figura 2.9 sono mostrati 2 intorni con diversi valori di P e R . Con questa formulazione non ci sono limitazioni sulle dimensioni dell'intorno.

Consideriamo un'immagine monocromatica $I(x, y)$, se le coordinate del pixel centrale sono (x_c, y_c) , le coordinate dei suoi P campioni che si trovano sulla

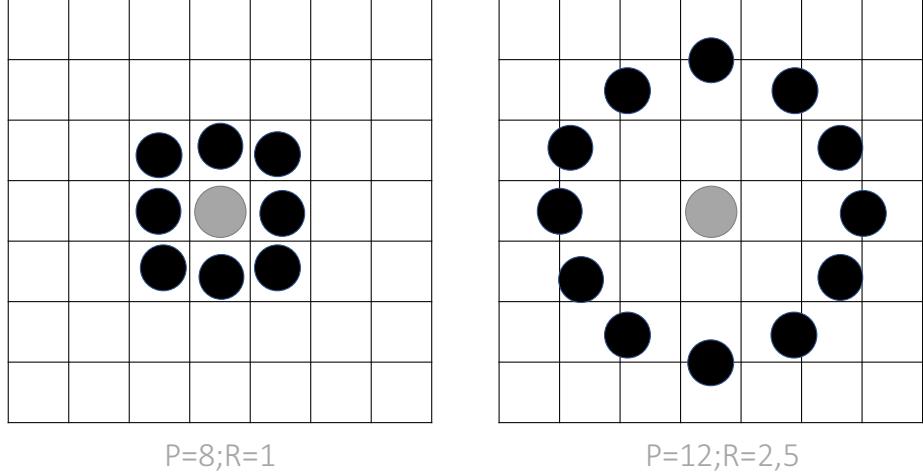


Figura 2.9: Operatore LBP generalizzato

circonferenza saranno descritte da:

$$\begin{cases} x_p = x_c + R \cos\left(\frac{2\pi p}{P}\right) \\ y_p = y_c + R \sin\left(\frac{2\pi p}{P}\right) \end{cases}, \quad p = 0, \dots, P-1 \quad (2.14)$$

Con g_c indichiamo il valore del pixel di coordinate (x_c, y_c) e con g_p il valore di grigio del punto campionato di coordinate (x_p, y_p) . La texture locale dell'intorno del punto (x_c, y_c) è caratterizzata dalla distribuzione congiunta dei livelli di grigio di $P+1$ punti, il pixel centrale e i campioni sulla circonferenza, ed è definita come:

$$T = t(g_c, g_0, g_1, \dots, g_{P-1}) \quad (2.15)$$

Un'altra rappresentazione della texture T può essere ottenuta sottraendo ai livelli di grigio dei campioni il livello di grigio g_c del pixel centrale e questa operazione si può fare senza perdita di informazione:

$$T = t(g_c, g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c) \quad (2.16)$$

Assumendo che g_c sia statisticamente indipendente dalle differenze possiamo approssimare la texture T fattorizzandola:

$$T \cong t(g_c)t(g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c) \quad (2.17)$$

Il termine $t(g_c)$ esprime la luminanza globale di un'immagine $I(x, y)$ [17] e quindi dal momento che non possiede informazioni rilevanti per l'analisi della

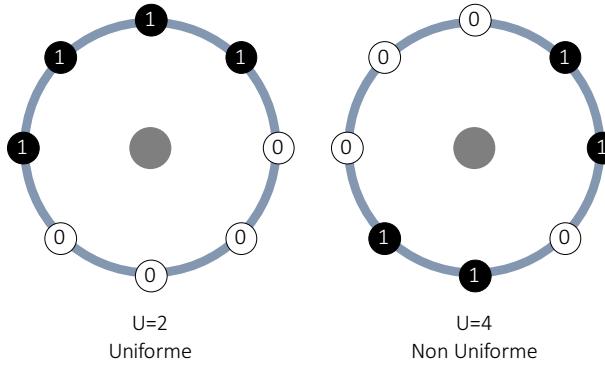


Figura 2.10: Esempio di un pattern uniforme e non uniforme

texture può essere trascurata:

$$T \cong t(g_0 - g_c, g_1 - g_c, \dots, g_{P-1} - g_c) \quad (2.18)$$

Per permettere l'invarianza di T alle trasformazioni monotoniche locali della scala di grigi andiamo a considerare solamente i segni delle differenze:

$$T \cong t(s(g_0 - g_c), s(g_1 - g_c), \dots, s(g_{P-1} - g_c)) \quad (2.19)$$

dove con $s(z)$ indichiamo la funzione segno:

$$s(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 1 \end{cases} \quad (2.20)$$

Per ottenere l'etichetta intera dell'intorno del pixel (x_c, y_c) , come nel caso del LBP base, andiamo a definire l'operatore $LBP_{P,R}(x_c, y_c)$ come:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} (g_p - g_c) 2^p \quad (2.21)$$

Quindi siamo passati dall'Equazione 2.19 ad un unico valore $LBP_{P,R}$ che descrive la struttura spaziale della texture in un intorno locale dell'immagine. Si nota facilmente che per un intorno del pixel ci sono 2^P possibili valori dell'etichetta intera che identifica univocamente il pattern a cui questa è associata, questi pattern sono dati dalle diverse combinazioni possibili nel codice binario. I valori $LBP_{P,R}$ sono calcolati per ogni pixel dell'immagine $I(x, y)$ e la distribuzione di questi valori è usata come vettore delle feature, indicato con S .

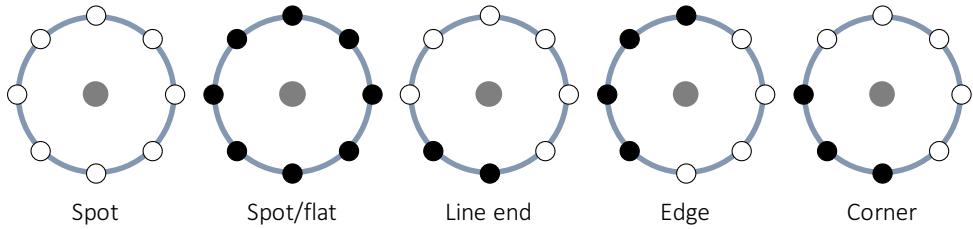


Figura 2.11: Diverse texture primitive individuate dall'operatore $LBP_{P,R}^{u2}$

Quindi definita $I(x, y)$, immagine $N \times M$, definiamo S come:

$$S = t(LBP_{P,R}(x, y)), \quad \begin{cases} x \in \{\lceil R \rceil, \dots, N - 1 - \lceil R \rceil\} \\ y \in \{\lceil R \rceil, \dots, m - 1 - \lceil R \rceil\} \end{cases} \quad (2.22)$$

Pattern uniformi Definiamo U come la misura di uniformità di un pattern e questa sarà uguale alla somma delle transizioni dal bit 1 al bit 0, o viceversa, nel codice binario. Un pattern si dirà *uniforme* se avrà al più $U = 2$, per $U > 2$ il pattern si dirà *non uniforme*. In Figura 2.10 possiamo vedere un esempio di un pattern uniforme e di uno non uniforme. I pattern uniformi rappresentano la maggioranza dei pattern riscontrati nelle texture delle immagini esaminate con intorni $(8, 1)$, spesso oltre il 90% [13], quindi considerando solamente i pattern uniformi non si commette un errore rilevante. L'operatore LBP, con intorno circolare (P, R) , che prende in esame solamente pattern uniformi si indica con $LBP_{P,R}^{u2}$.

Come abbiamo visto poco fa nel caso generale in cui consideravamo pattern generici per ogni intorno di un pixel erano possibili 2^P diversi pattern, ora nel considerare solamente pattern uniformi il numero di pattern possibili è sceso a $P \cdot (P - 1) + 2$ quindi si ha una riduzione di dimensionalità del vettore di feature estratto dalla distribuzione dei pattern. Altro beneficio nell'usare $LBP_{P,R}^{u2}$ è che si riesce a trascurare il rumore e ad individuare texture locali importanti dette *micro-texture* primitive e rappresentanti i pattern uniformi più frequenti. In Figura 2.11 vediamo le diversi primitive individuate dall'operatore LBP.

Invarianza alla rotazione L'operatore così costruito è invariante alle variazioni locali della scala di grigio ma non alla rotazione dell'immagine. Durante la rotazione i valori di grigio g_P ruotano lungo la circonferenza centrata su (x_c, y_c) , questo causa la generazione di diversi pattern binari per uno stesso intorno. Per eliminare l'effetto della rotazione possiamo mappare tutte le possibili rotazioni di un pattern assegnando a queste un unico valore. Per fare

questo definiamo l'operatore LBP invariante alla rotazione:

$$LBP_{P,R}^{ri} \min_i \{ROR(LBP_{P,R}, i) | i = 0, \dots, P - 1\} \quad (2.23)$$

dove con $ROR(x, i)$ si è indicata la rotazione circolare bit a bit della sequenza binaria x di i bit. L'operatore $LBP_{P,R}^{ri}$ ruota ogni pattern fino a trovare la configurazione a valore minimo, per esempio 10000010_2 , 00101000_2 e 00000101_2 sono tutti mappati con lo stesso valore minimo 00000101_2 .

Invarianza alla rotazione e alla variazione di scala Ora possiamo definire un operatore invariante sia alle variazioni della scala di grigi, sia alla rotazione. Quindi definiamo $LBP_{P,R}^{riu2}$ come:

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{se } U(LBP_{P,R}) \leq 2 \\ P + 1 & \text{altrimenti} \end{cases} \quad (2.24)$$

dove con $U(x)$ avevamo definito la misura di uniformità di un pattern:

$$U(LBP_{P,R}) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)|$$

L'operatore così definito identifica $P + 1$ pattern *uniformi* etichettati con valori interi compresi nel range $\{0, \dots, P\}$, invece i pattern *non uniformi* sono raggruppati in un'unica etichetta dal valore intero pari a $P + 1$. Quindi l'operatore $LBP_{P,R}^{riu2}$ identifica in totale $P + 2$ etichette diverse, molte meno delle 2^P identificate dall'operatore $LBP_{P,R}$.

Vettore delle feature

Come già detto dobbiamo applicare l'operatore LBP, nel nostro caso applicheremo l'operatore $LBP_{P,R}^{riu2}$ invariante alle rotazioni e alle variazioni di scala, ad ogni pixel dell'immagine. Quindi definita un'immagine $I(x, y) N \times M$ applicheremo l'operatore $LBP_{P,R}^{riu2}(x, y)$ ad ogni punto dell'immagine compreso nel range $\{x \in \{\lceil R \rceil, \dots, N - 1 - \lceil R \rceil\}; y \in \{\lceil R \rceil, \dots, m - 1 - \lceil R \rceil\}\}$. Il vettore delle feature sarà l'istogramma delle occorrenze dei pattern interi all'interno dell'immagine.

Esempio Andiamo a trovare l'istogramma dell'immagine in Figura 2.12. Prima di applicare l'operatore LBP convertiamo l'immagine RGB in un immagine



Figura 2.12: Immagine satellitare di Firenze (dati mappa: Google, CNES/Airbus, DigitalGlobe)

in toni di grigio dove ogni pixel ha un valore compreso tra 0 e 255. Nell'applicare l'operatore consideriamo intorni del pixel centrale $(8, 1)$, 8 punti sulla circonferenza di raggio 1. Applicando l'operatore $LBP_{P,R}^{riu^2}$, come già detto, andremo ad identificare $P + 2$ pattern che si presenteranno all'interno dell'immagine I con frequenza diversa. In Figura 2.13 vediamo l'istogramma delle occorrenze di tali pattern all'interno dell'immagine, questo istogramma rappresenta il vettore delle feature estratte.

2.6.2 Gray Level Co-occurrence Matrix

La matrice di co occorrenza dei livelli di grigio, in inglese chiamata *gray level co-occurrence matrix (GLCM)*, proposta da Haralick nel 1973 [8] rappresenta uno dei metodi più utilizzati per l'analisi della distribuzione spaziale dei livelli di grigio all'interno di un'immagine.

Definita un'immagine $I(x, y) N \times M$ il GMCL consiste nel calcolare la matrice C , detta *matrice di occorrenza*, nella quale ciascun punto (i, j) della matrice rappresenta il numero di volte che il livello di grigio i si trova ad una distanza δ e ad una certa inclinazione θ dal livello di grigio j . Così definita la matrice C è una matrice $L \times L$, dove L rappresenta il numero dei livelli di grigio disponibili. L'operatore GLCM è parametrizzato mediante due parametri (δ, θ) :

- Un **offset** δ che rappresenta la distanza relativa fra due pixel misurata in pixel;

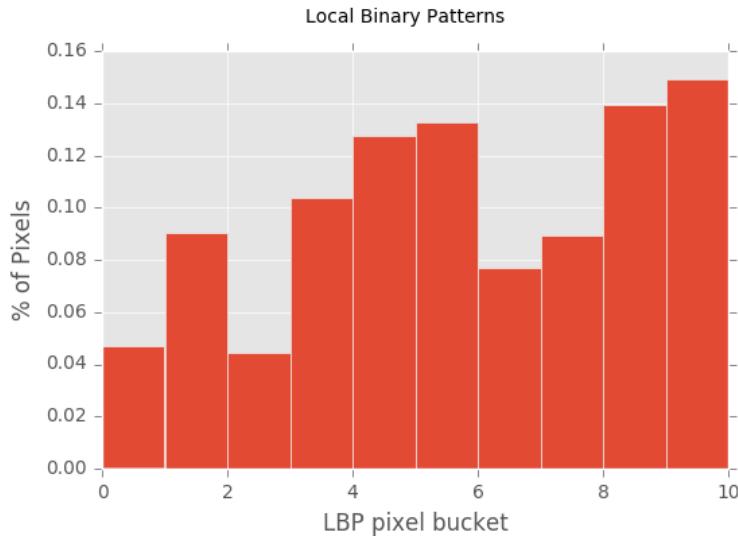


Figura 2.13: Istogramma delle occorrenze dei pattern LBP

- Un **angolo** θ che rappresenta l’orientazione relativa fra due pixel.

Per δ di solito si sceglie un valore tra 1 e 2, valori bassi di δ sono scelti per un’analisi più fine della texture. Dal momento che un pixel è circondato da 8 pixel diversi i valori di θ sono compresi nel range $\{0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ\}$ ma per come è definito l’operatore possiamo avere solamente 4 orientazioni $0^\circ, 45^\circ, 90^\circ, 135^\circ$, rispettivamente orizzontale, diagonale destra, verticale e diagonale sinistra, infatti scegliendo un pixel orientato a θ oppure a $\theta+180^\circ$ si ha lo stesso comportamento.

Esempio Ora facciamo un esempio per far capire come funziona l’operatore GCML. Definiamo un’immagine I 4×4 con $L = 3$ (abbiamo cioè 3 possibili livelli di grigio per pixel):

$$I = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 \\ 1 & 2 & 2 & 2 \end{bmatrix}$$

Applicando a questa l’operatore GCML parametrizzato $(1, 0^\circ)$, otteniamo la matrice di co occorrenza C 3×3 :

$$C = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 5 \end{bmatrix}$$

Nel caso l'operatore fosse stato parametrizzato $(1, 90^\circ)$ avremmo ottenuto la seguente matrice C :

$$C = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

Descrittori di Haralick

Per stimare la somiglianza fra due matrici di co-occorrenza Haralick propose [8] 14 feature statistiche estratte da queste matrici. In questa sezione discuteremo solamente di 6 feature che saranno usate in seguito nella classificazione delle immagini satellitari.

Il primo passo dell'estrazione delle features è la normalizzazione della matrice C con la seguente equazione di normalizzazione $P(i, j)$:

$$P(i, j) = \frac{C(i, j)}{\sum_{i=1}^L \sum_{j=1}^L C(i, j)} \quad (2.25)$$

dove con $C(i, j)$ abbiamo definito il valore nella cella (i, j) della matrice C , con L il numero di righe, colonne della matrice quadrata C e con $P(i, j)$ la probabilità di avere $C(i, j)$ coppie che rispettano la relazione imposta dall'operatore GLCM. Quindi se abbiamo una matrice C definita nel seguente modo:

$$C = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

la matrice P sarà:

$$P = \frac{1}{10} \cdot \begin{bmatrix} 1 & 0 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

Ora passiamo ad illustrare sei delle feature introdotte da Haralick. Queste 6 feature rappresentano delle statistiche del secondo ordine dal momento che i pixel sono considerati a coppie. In Tabella 2.4 abbiamo le 6 feature raccolte in modo riassuntivo.

Contrasto Il *contrasto* è una feature che misura le variazioni locali di intensità della scala di grigi all'interno dell'immagine. Questa è definita nel

seguente modo:

$$\sum_{i=1}^L \sum_{j=1}^L (i - j)^2 P(i, j) \quad (2.26)$$

Bassi valori di contrasto si ottengono quando l'immagine ha dei livelli di grigio quasi costanti in tutta l'immagine, al contrario il valore del contrasto è alto quando i livelli di grigio sono non costanti.

Diversità La *diversità*, dall'inglese *dissimilarity*, è definita in modo simile al contrasto:

$$\sum_{i=1}^L \sum_{j=1}^L |i - j| P(i, j) \quad (2.27)$$

Omogeneità L'*omogeneità*, dall'inglese *homogeneity*, anche chiamata *inverse difference moment* è una feature che è inversamente proporzionale al contrasto e quindi rappresenta una misura dell'omogeneità dell'immagine.

$$\sum_{i=1}^L \sum_{j=1}^L \frac{P(i, j)}{1 + (i - j)^2} \quad (2.28)$$

Elevati valori di homogeneity si ottengono quando la maggior parte delle occorrenze si trovano lungo la diagonale della matrice C sulla quale si trovano le frequenze delle coppie di grigio con lo stesso valore di grigio.

ASM L'*angular second moment*, ASM , è definito nel seguente modo:

$$ASM = \sum_{i=1}^L \sum_{j=1}^L P(i, j)^2 \quad (2.29)$$

Questa feature misura quanto la texture dell'immagine è omogenea, infatti se tutti i pixel hanno lo stesso valore k allora $P(k, k) = 1$ e $P(i, j) = 0$, con $i \neq k$ e $j \neq k$ e quindi $ASM = 1$.

Energia L'*energia* è definita come la radice quadrata dell'ASM, $EN = \sqrt{ASM}$, quindi:

$$EN = \sqrt{\sum_{i=1}^L \sum_{j=1}^L P(i, j)^2} \quad (2.30)$$

Correlazione La *correlazione* è una feature che misura il grado di correlazione tra i pixel dell’immagine ed è definita nel seguente modo:

$$COR = \sum_{i=1}^L \sum_{j=1}^L P(i, j) \left[\frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \quad (2.31)$$

Dove con:

- $\mu_i = \sum_{i=1}^L \sum_{j=1}^L i \cdot P(i, j)$ e $\mu_j = \sum_{i=1}^L \sum_{j=1}^L j \cdot P(i, j)$ abbiamo indicato rispettivamente la media per righe e per colonne;
- $\sigma_i = \sum_{i=1}^L \sum_{j=1}^L (i - \mu_i)^2 \cdot P(i, j)$ e $\sigma_j = \sum_{i=1}^L \sum_{j=1}^L (j - \mu_j)^2 \cdot P(i, j)$ abbiamo indicato rispettivamente la deviazione standard per righe e per colonne;

La correlation può assumere valori compresi nell’intervallo $[-1, +1]$.

Feature	Formula
Contrast	$\sum_{i=1}^L \sum_{j=1}^L (i - j)^2 P(i, j)$
Dissimilarity	$\sum_{i=1}^L \sum_{j=1}^L i - j P(i, j)$
Homogeneity	$\sum_{i=1}^L \sum_{j=1}^L \frac{P(i, j)}{1+(i-j)^2}$
ASM	$\sum_{i=1}^L \sum_{j=1}^L P(i, j)^2$
Energy	$\sqrt{\sum_{i=1}^L \sum_{j=1}^L P(i, j)^2}$
Correlation	$\sum_{i=1}^L \sum_{j=1}^L P(i, j) \left[\frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$

Tabella 2.4: Feature introdotte da Haralick

2.7 Algoritmi di Machine Learning utilizzati

In questa sezione descriviamo alcuni algoritmi di addestramento che saranno successivamente utilizzati sia per il classificatore di immagini satellitari sia per quello degli interventi rilevanti. I due classificatori si basano principalmente sull’utilizzo delle Random Forest che sono un’estensione del Decision Tree quindi inizieremo descrivendo ampiamente quest’ultimo algoritmo per poi passare all’implementazione di questo nella Random Forest.

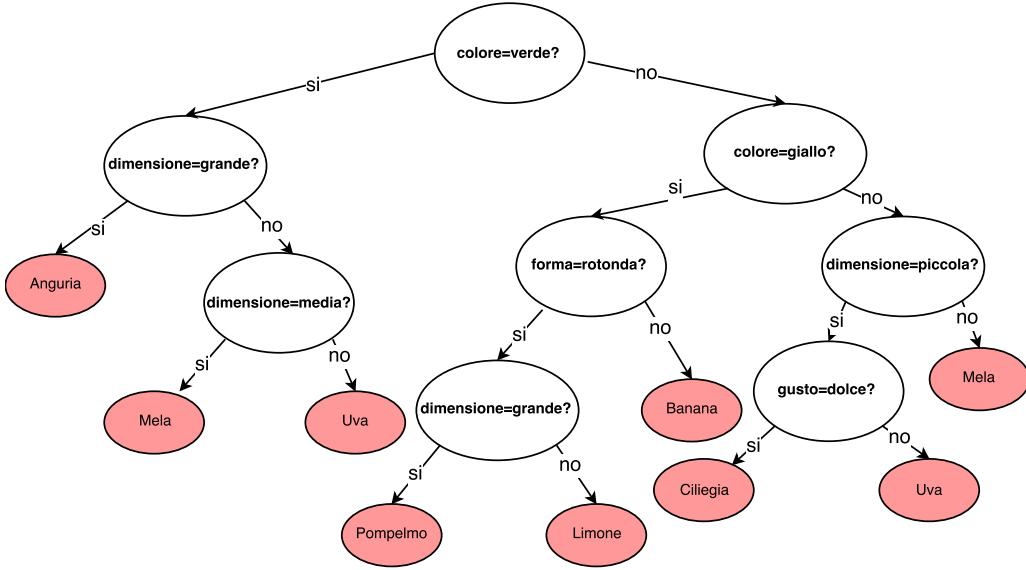


Figura 2.14: Esempio di albero decisionale binario ($B=2$)

2.7.1 Alberi decisionali

Gli alberi decisionali, dall’inglese *Decision Tree*, sono una tipologia di algoritmi di apprendimento supervisionato che consentono di sviluppare un sistema basato sulla successione di una serie di domande che definiscono le regole decisionali atte a stabilire l’output del sistema. La sequenza di domande è disposta in un albero orientato dove per convenzione il nodo principale, detto radice dell’albero (dall’inglese *root node*), è posto in cima. Il nodo principale, così come i nodi successivi, sono connessi tra loro tramite delle connessioni chiamate rami, dall’inglese *branch*. I nodi terminali che non sono seguiti da ulteriori nodi sono chiamati *nodi foglia*, dall’inglese *leaf node*, ad ogni nodo foglia è associato un valore oppure un’etichetta che sarà successivamente assegnato al campione in ingresso al modello nella fase di predizione. Il nodo da cui discendono altri nodi è chiamato *nodo genitore*, invece i nodi discendenti sono chiamati *nodi figli*, questi possono essere considerati i root node dei sotto-alberi che seguono. Il numero di rami che discendono da un nodo viene chiamato rapporto di ramificazione, dall’inglese *branching factor* oppure *branching ratio*, e lo si indica con B . Comunque ogni decisione può essere fatta usando solamente split di tipo binario ($B = 2$). In Figura 2.14 vediamo un esempio di albero decisionale che utilizza solamente split di tipo binario.

Dal momento che le domande possono essere poste in modo da avere delle risposte del tipo "vero/falso", "si/no" oppure "l’attributo ha un determinato valore" che non richiedono alcuna nozione di metrica, gli alberi decisionali ap-

partengono alla categoria dei metodi non metrici. Questo significa che non richiedono alcuna nozione di metrica e quindi di distanza fra i dati, pertanto accettano sia dati di tipo categorico sia dati di tipo numerico.

Gli alberi decisionali possono essere applicati sia per compiti di regressione, dove l'output del sistema sarà di tipo quantitativo, sia per compiti di classificazione nei quali si cerca di predire la classe di un campione di ingresso al sistema.

I principali algoritmi di addestramento per gli alberi decisionali sono il *CART* [3], acronimo di *classification and regression trees*, l'algoritmo ID3 [16], *Iterative Dichotomiser 3*, e il C4.5 [16] evoluzione dell'algoritmo ID3.

Il beneficio principale nell'uso di un decision tree è l'interpretabilità, infatti le informazioni contenute nell'albero possono essere espresse come operazioni logiche. L'interpretabilità si manifesta in due forme principali, per prima cosa per ogni campione in ingresso al sistema si giunge all'etichetta predetta attraverso un percorso di decisioni che dal nodo radice dell'albero porta al nodo foglia. Quindi se abbiamo i seguenti attributi $\{gusto, colore, forma, dimensione\}$, facendo riferimento alla Figura 2.14, un campione $\mathbf{x} = \{dolce, verde, rotonda, grande\}$ sarà classificato come *anguria* dal momento che $\{colore=verde\}$ **AND** $\{dimensione=grande\}$. Un secondo aspetto dell'interpretabilità è quello di ottenere una più chiara rappresentazione delle classi stesse, per esempio *melone* = (*verde AND media*) **OR** (**NOT** *verde AND NOT giallo AND NOT piccola*). Più gli alberi diventano grandi e profondi più la complessità delle regole aumenta fino a venire meno la facilità di interpretazione delle informazioni. Infine un altro beneficio del decision tree è la rapidità con la quale si giunge alla classificazione dei campioni in ingresso al modello.

Crescita di un albero

Il principio fondamentale che sta alla base della crescita di un albero decisionale è quello della semplicità [5], quindi si preferiscono decisioni che portano alla crescita di alberi semplici, compatti e con pochi nodi.

La crescita dell'albero inizia dal nodo principale dove i campioni del training set vengono divisi cercando di raggiungere il massimo *guadagno informativo*, dall'inglese *information gain*. Il fattore di ramificazione B viene scelto dal progettista in fase di sviluppo dell'algoritmo, questo può variare anche lungo tutto l'albero ma solitamente si preferisce avere $B = 2$ così ad ogni nodo la decisione viene trattata come un problema di ottimizzazione monodimensionale. In modo iterativo si può continuare a dividere i dati in ciascun nodo

figlio fino ad ottenere, volendo, nodi foglia puri, cioè che contengono campioni appartenenti unicamente ad una classe.

La funzione obiettivo che ad ogni divisione andiamo ad ottimizzare e che porta al massimo guadagno informativo è definita nel seguente modo:

$$\Delta I(N) = I(N_p) - \sum_{j=1}^m \frac{N_j}{N_p} I(N_j) \quad (2.32)$$

Dove con N_p abbiamo indicato il nodo padre, con N_j il nodo figlio j-esimo, con $I(N_p)$ e $I(N_j)$ rispettivamente l'impurità nel nodo padre e nel nodo figlio, con $\frac{N_j}{N_p}$ indichiamo la frazione dei campioni che sono passati dal nodo padre al nodo figlio j-esimo, infine con m indichiamo il numero di nodi figli. La funzione rappresenta quindi la diminuzione dell'impurità e il massimo guadagno informativo si avrà con la divisione che porta alla massima riduzione di impurità. L'ottimizzazione dell'equazione 2.32 è un'ottimizzazione locale, fatta cioè per ogni singolo nodo padre, quindi appartiene ai metodi di ottimizzazione *greedy*. L'inconveniente dell'equazione 2.32 è che un grande numero di nodi figli è favorito rispetto a scelte che portano ad un numero di divisioni minore. Quindi per evitare questo inconveniente l'equazione 2.32 deve essere scalata:

$$\Delta I_{gr}(N) = \frac{\Delta I(N)}{-\sum_{j=1}^m \frac{N_j}{N_p} \log_2 \frac{N_j}{N_p}} \quad (2.33)$$

Questa viene chiamata *rapporto di guadagno informativo*, dall'inglese *information gain ratio*.

Nel caso di alberi binari ($B = 2$) la funzione obiettivo è del tipo:

$$\Delta I(N) = I(N_p) - \frac{N_{left}}{N_p} I(N_{left}) - \frac{N_{right}}{N_p} I(N_{right}) \quad (2.34)$$

Con $I(N)$ abbiamo indicato l'impurità del nodo N e quindi se tutti i campioni del nodo appartengono alla stessa classe l'impurità sarà nulla.

Le tre *misure di impurità* o *criteri di suddivisione* che vengono più spesso utilizzati negli alberi decisionali sono l'*impurità di Gini*, l'*entropia* e l'*errore di classificazione*.

Impurità di Gini L'impurità di Gini è definita nel seguente modo:

$$I_G(N) = \sum_{i \neq j} P(\omega_i) P(\omega_j) = 1 - \sum_{i=1}^c P(\omega_i)^2 \quad (2.35)$$

dove con $P(\omega_i)$ abbiamo indicato la frazione di campioni appartenente alla classe ω_i presenti nel nodo N. L'impurità risulterà massima se le classi sono perfettamente mescolate, per esempio, considerando un problema binario ($c = 2$) si avrà:

$$I_G(N) = 1 - \sum_{i=1}^2 0.5^2 = 0.5$$

Intuitivamente, l'impurità di Gini rappresenta il tasso di errore previsto nel nodo N se l'etichetta è selezionata in modo casuale dalla distribuzione delle classi presenti in N, quindi può essere visto come un criterio per minimizzare la probabilità di errata classificazione.

Entropia Il criterio di suddivisione più famoso è la misura di Entropia. Partiamo nel definire l'Entropia per tutte le classi non vuote $P(\omega_i) \neq 0$:

$$I_E(N) = - \sum_i^c P(\omega_i) \log_2 P(\omega_i) \quad (2.36)$$

Il valore Entropia sarà nullo quando se tutti i campioni nel nodo N appartengono alla stessa classe. Diversamente, l'Entropia sarà massima se le classi, al nodo N, sono equamente distribuite. L'Entropia quindi cerca di massimizzare l'informazione reciproca all'interno di un nodo N.

Errore di classificazione Un altro misura di impurità è l'Errore di classificazione, definito nel seguente modo:

$$I_{EC}(N) = 1 - \max_i P(\omega_i) \quad (2.37)$$

Questo criterio misura la minima probabilità che un campione sia erroneamente classificato al nodo N. Tra le tre misure questa è quella che, a pari probabilità, presenta un picco più marcato. Inoltre, questa misura presenta una derivata discontinua il che può portare a problemi quando si cerca di trovare la decisione ottima su uno spazio parametrico continuo.

In Figura 2.15 mostriamo le tre misure di impurità, nel caso binario, come funzione della probabilità di una delle due classi.

Idealmente, durante la crescita dell'albero, si può continuare a dividere i campioni dal nodo padre ai nodi figli fino a quando non si ottengono solamente nodi foglia puri. Nella pratica questo può produrre un albero molto profondo, dotato di molti nodi e ciò può portare ad un aumento sproporzionato della

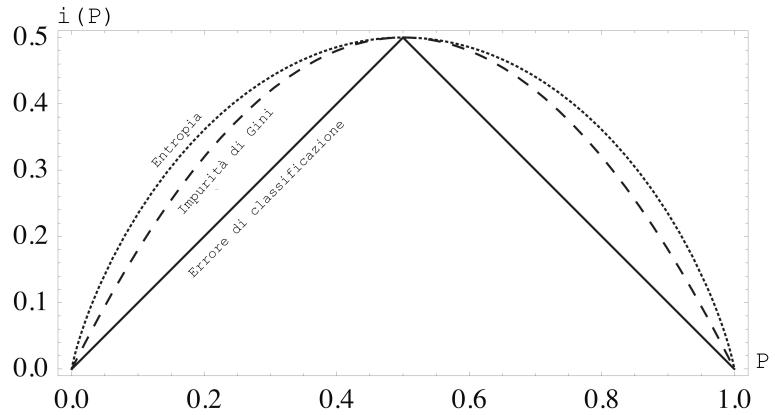


Figura 2.15: Tre misure di impurità in funzione della probabilità di una delle classi nel caso binario.

complessità computazionale del nostro metodo. Un eccessivo aumento delle dimensioni di un albero e della sua complessità influisce sulla bontà di un modello e può portare a problemi di *overfitting* ottenendo un peggioramento complessivo delle performance del modello. Per questi motivi durante la crescita dell’albero vengono definiti dei *criteri di arresto* e dei metodi di *potatura (pruning)* dell’albero che vanno a ridurre le dimensioni massime durante la crescita dell’albero.

2.7.2 Random Forest

La *foresta casuale*, dall’inglese *Random Forest*, è stata proposta per la prima volta nel 1995 da Tin Kam Ho dei Bell Labs [9] e rappresenta un’estensione del Decision Tree.

Le Random Forest appartengono alla categoria dei metodi di *apprendimento ensemble* cioè quei metodi basati sulla combinazione di più *sistemi di apprendimento deboli* per costruire un modello più robusto, un *sistema di apprendimento forte* che offre un minore errore di generalizzazione e meno propenso a problemi di overfitting.

Come suggerisce il nome, il Random Forest è un modello basato sulla combinazione di più Decision Tree. Infatti, il Decision Tree discusso sopra è un metodo che soffre di *alta varianza*, questo significa che se dividiamo il training set in due parti casuali e con queste addestriamo due Decision Tree i risultati che otteniamo potrebbero essere abbastanza diversi. Invece un metodo con *bassa varianza* porterà a risultati simili se addestrato ripetutamente con distinti training set. L’idea su cui si basa il Random Forest è il *bagging* che andremo

ora a discutere.

Bagging Il Bagging, chiamato anche *bootstrap aggregation*, è una procedura generale utilizzata per ridurre la varianza di un metodo statistico. Nei compiti di regressione un metodo naturale per ridurre la varianza del nostro modello di apprendimento è di raccogliere molti training set diversi e con questi addestrare separatamente diversi modelli di addestramento, infine la predizione finale sarà la media dei risultati ottenuti dai diversi modelli. Quindi, prendendo B diversi training set con i quali addestriamo altrettanti modelli, il modello di addestramento finale dotato di bassa varianza sarà:

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$$

dove con $\hat{f}^b(x)$ abbiamo indicato i B diversi modelli addestrati.

Molte volte però non esiste la possibilità di poter avere diversi training set, quindi un metodo alternativo consiste nel creare diversi training set partendo dall'unico training set disponibile scegliendo da esso n campioni con reinserimento, questa procedura è detta *bootstrap*. Ora con questi set appena creati addestriamo diversi modelli, il risultato finale sarà quindi la media dei risultati dei diversi modelli:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$$

questa procedura è chiamata *bagging*.

Per compiti di classificazione l'output finale è scelto tramite un *voto di maggioranza*, quindi la predizione finale sarà quella che globalmente è stata la più frequente.

Le Random Forest propongono un miglioramento rispetto al bagging grazie al quale si de-correlano maggiormente i Decision Tree tra di loro. Infatti a differenza del bagging, durante la crescita di un Decision Tree vengono scelti come candidati per lo split solamente m feature delle p disponibili. Solitamente $m \approx \sqrt{p}$, quindi il numero delle feature considerate durante lo split è approssimativamente uguale alla radice quadrata del numero totale delle feature. In altre parole, nell'addestramento della Random Forest, ad ogni divisione in un Decision Tree, non viene presa in considerazione la totalità delle feature disponibili. Questa considerazione viene fatta dal momento che se nel dataset esistono delle feature forti rispetto alle altre, durante la crescita degli alberi la maggior parte dei Decision Tree nella foresta userà queste feature, quindi tutti

gli alberi saranno abbastanza simili tra di loro e quindi correlati. Facendo la media di molti modelli correlati tra di loro non permette di abbassare di molto la varianza del modello ensemble.

Le Random Forest consentono di sorvolare questo problema concedendo ad ogni split di considerare solamente una parte delle feature. Pertanto, in media $(p - m)/p$ degli split non prenderà conto delle feature forti, permettendo alle altre di essere scelte per lo split. Con questa tecnica riusciamo a de-correlare gli alberi tra di loro ed abbassare così la varianza del modello finale.

Capitolo 3

Materiali

Secondo una ricerca pubblicata nel 2015 dall'International Data Corporation, il mercato dei Big Data raggiungerà un valore di 48,6 miliardi di dollari entro il 2019. In questo capitolo parleremo di quello che secondo molti è considerato l'oro nero del nuovo millennio: i Dati.

I dati rappresentano l'elemento fondamentale del nostro lavoro grazie ai quali saremo in grado di analizzare il nostro problema e di addestrare il nostro modello. In questo capitolo parleremo dei database forniteci dal Corpo Nazionale dei Vigili del Fuoco, dei dati che ci siamo calcolati a partire da questi e di un altro database creato dall'Istat, l'Istituto nazionale di statistica, per fornire una descrizione statistica delle caratteristiche del territorio comunale. Nell'ultima parte del capitolo descriveremo di come siamo giunti alla creazione del nostro dataset utilizzato successivamente durante la fase di addestramento del nostro sistema automatico. Successivamente passeremo ad una descrizione completa delle feature che serviranno a descrivere i campioni contenuti nel dataset.

3.1 Dati disponibili

3.1.1 Database interventi

Il centro operativo dei vigili del fuoco ci ha fornito due tipi di database, la banca dati della centrale operativa contenente gli interventi svolti dai vigili del fuoco e il database degli sms di cui discuteremo nella sezione seguente.

La banca dati contiene gli interventi per le provincie di Milano, Napoli, Roma e Torino per l'anno 2016 e solamente per la provincia di Roma abbiamo anche gli interventi avvenuti durante il 2017 per un periodo che va dal 01/01/2017 al 23/06/17.

Per le provincie di Milano, Napoli, Roma (per il periodo nell'anno 2017) e To-

rino vengono usati 26 attributi $\{intervento, data_intervento, data_chiamata, ora_chiamata, ora_uscita_prima_partenza, desc_tipologia, dettaglio_tipologia, X, Y, nome_strada, desc_comune, sigla_provincia, richiedente, enti_intervenuti, data_chiusura, ora_chiusura, boschi, campi, sterpaglie, num_fonogramma, num_protocollo_fono, genere_mezzo, sigla_squadra, turno_squadra, data_servizio, intervento_rilevante\}$, invece per la provincia di Roma, per l'anno 2016, l'attributo *intervento_rilevante* è sostituito dall'attributo *obiettivo_rilevante*.

Gli attributi *X* e *Y* indicano rispettivamente la longitudine e la latitudine del luogo in cui si è verificato l'evento che ha richiesto l'intervento dei vigili del fuoco.

L'attributo *sigla_provincia*, insieme anche all'attributo *data_intervento*, ci sarà utile per trovare una corrispondenza, un legame, tra i database degli interventi e i database dei messaggi sms in modo da poter rilevare, in modo manuale, se per un dato intervento è stato inviato un sms, condizione che serve a definire se un intervento è *rilevante*.

In Tabella 3.1 troviamo il riepilogo dei 5 database degli interventi dei vigili del fuoco.

3.1.2 Database sms

Il centro nazionale dei vigili del fuoco ci ha fornito due database degli sms grazie ai quali saremo in grado di etichettare, nella fase di assemblaggio del dataset, gli interventi del database degli interventi come *rilevanti* oppure come *non rilevanti*.

I database forniteci sono due, il primo riguarda l'intero anno del 2016 il secondo invece riguarda l'anno 2017 per un periodo che va dal 10/03/17 al 24/05/17, ed entrambi possiedono 22 attributi $\{c_timestamp, id, smslog_id, flag_deleted, p_name, p_gateway, p_src, p_dst, p_footer, p_msg, p_datetime, p_update,$

Provincia	Periodo	Numero di istanze	Numero di attributi
Milano	2016	42336	26
Napoli	2016	38595	26
Roma	2016	67842	26
Roma	da 01/01/2017 a 23/06/2017	30516	26
Torino	2016	35762	26

Tabella 3.1: Riepilogo database interventi

p_status, p_gpid, p_credit, p_sms_type, unicode, queue_code, uid, p_tentativo, p_datetime_originale, alertcon}. I due database contengono 355756 e 69133 sms, rispettivamente per l'anno 2016 e 2017.

I database che ci hanno fornito contengono dei dati ridondanti dal momento che uno stesso messaggio sms viene inviato a più persone (il nome della persona a cui è stato inviato il messaggio si trova sotto l'attributo *p_name*) e quindi lo stesso messaggio si presenta in maniera ripetitiva. Quindi i due database devono subire una prima elaborazione con la quale si va ad eliminare gli sms ridondanti. Questa operazione è stata effettuata tramite uno script in MATLAB utilizzando l'attributo *{queue_code}* per identificare uno stesso sms, in questo modo avremo solamente 4585 sms per il database del 2016 e 672 sms per il database del 2017; si può vedere che in media uno stesso messaggio viene inviato a circa 90 persone.

Ottenuto un database privo di elementi ridondanti andiamo ad elaborarlo nuovamente al fine di estrarre delle informazioni utili per il nostro compito. L'attributo *p_msg* contiene il messaggio testuale dell'sms inviato come visto nell'esempio che segue:

#sms	p_msg
1	CON-VVF ALLARME BOMBA (TE): Facendo seguito sms precedente si comunica che alle ore 12.10 le operazioni sono terminate. Falso Allarme.
2	CON-VVF (CS): evento sismico ore 13,35 m.do 3.5 prof. 280 km, comuni interessati: Orsomarso, Papasidero, Santa Domenica di Talao. Nessuna segnalazione è pervenuta ai VVF.
...	...
n	CON-VVF (FG) incendio capannone, alla periferia cittadina, contenente eco balle e materie plastiche. L'impianto era sotto sequestro. VV.F. dalle 08.40; ben visibile la colonna di fumo. Sul posto anche ARPA.

Quindi andiamo a creare un nuovo attributo, *prov.*, in cui andremo ad inserire la provincia a cui fa riferimento il messaggio sms, quindi otterremo una situazione del tipo in Tabella 3.2.

Concludendo, i database degli sms elaborati contengono un nuovo attributo, nel quale è inserita la provincia in cui è avvenuto l'intervento dei vigili del fuoco: questo ci servirà in seguito per collegare i due database, come detto in precedenza.

In Tabella 3.3 si trovano in modo riassuntivo i database appena trattati.

#sms	p_msg	prov.
1	CON-VVF ALLARME BOMBA (TE): Facendo seguito sms precedente si comunica che alle ore 12.10 le operazioni sono terminate. Falso Allarme.	TE
2	CON-VVF (CS): evento sismico ore 13,35 m.do 3.5 prof. 280 km, comuni interessati: Orsomarso, Papasidero, Santa Domenica di Talao. Nessuna segnalazione è pervenuta ai VVF.	CS
...
n	CON-VVF (FG) incendio capannone, alla periferia cittadina, contenente eco balle e materie plastiche. L'impianto era sotto sequestro. VV.F. dalle 08.40; ben visibile la colonna di fumo. Sul posto anche ARPA.	FG

Tabella 3.2: Esempio di aggiunta feature *prov.*

3.1.3 Istat

Al fine di acquisire una conoscenza del territorio su cui si svolge l'intervento andiamo ad utilizzare un database diffuso dall'*Istat, Istituto nazionale di statistica*. Questo database, tramite una serie di attributi, provvede ad una descrizione statistica delle caratteristiche geografiche del territorio. La risoluzione di queste statistiche è a livello comunale e quindi l'indagine presenta una granularità relativamente ridotta. Attraverso i seguenti undici attributi si descrivono i comuni italiani: {*Codice Regione*, *Codice Istat del Comune* (formato alfanumerico), *Codice Istat del Comune* (formato numerico), *Denominazione* (solo italiano), *Denominazione* (solo tedesco), *Zona altimetrica*, *Altitudine del centro* (metri), *Comune litoraneo*, *Comune Montano*, *Superficie territoriale* (kmq) e *Grado di urbanizzazione*}.

Gli attributi che utilizzeremo in fase di addestramento del nostro modello sono {*Zona altimetrica*, *Altitudine del centro* (metri), *Comune litoraneo*, *Co-*

	SMS_2016	SMS_2017
Periodo	intero anno 2016	da 10/03/17 a 24/05/17
Numero SMS	355756	69133
Numero SMS dopo elaborazione	4585	672
Numero attributi	22	22
Numero attributi dopo elaborazione	23	23

Tabella 3.3: Resoconto database degli SMS

mune Montano, Superficie territoriale (kmq) e Grado di urbanizzazione}. La Tabella 3.4 mostra una descrizione dei seguenti attributi.

3.2 Assemblaggio Dataset

In questa sezione andiamo ad assemblare il nostro dataset, arricchendolo di feature estratte e calcolate a partire dai database sopra descritti.

Per prima cosa andiamo ad arricchire i database degli interventi di un nuovo attributo, *sms_invia*, il quale indicherà se per il relativo intervento è stato inviato oppure no un messaggio sms. Questo attributo quindi potrà assumere solamente due valori *S* oppure *N*, rispettivamente *sms inviato* e *sms non inviato*. L'operazione di aggiunta di tale attributo è stata svolta confrontando ogni messaggio sms con i database degli interventi cercando di trovare una corrispondenza tra i due. Tale processo è stato svolto manualmente dal momento che non è stato predisposto un codice che permetta di instaurare una corrispondenza biunivoca tra sms ed intervento.

Per cercare questa corrispondenza abbiamo fatto ricorso a dei descrittori del database dei messaggi: *p_datatime*, che indica l'ora in cui è stato inviato il

Attributo	Descrizione/Legenda
Zona altimetrica	1=Montagna interna; 2=Montagna litoranea; 3=Collina interna; 4=Collina litoranea; 5=Pianura
Altitudine del centro (metri)	Altitudine s.l.m. (metri) del centro capoluogo rilevata in corrispondenza della sede del Municipio
Comune litoraneo	1=Comune litoraneo, 0=Comune non litoraneo
Comune Montano	NM=Non montano, T=Totalmente montano, P=Parzialmente montano
Superficie territoriale (kmq)	L'estensione totale del territorio nazionale deriva dalla somma delle misure delle superfici dei comuni italiani al 9 ottobre 2011 (XV° Censimento generale della popolazione)
Grado di urbanizzazione	1=densamente popolato; 2=densità intermedia; 3=scarsamente popolato (rurale)

Tabella 3.4: Descrizione attributi database ISTAT utilizzato nel presente lavoro

messaggio, *provincia*, il quale indica la provincia in cui si è svolto l'intervento e *p_msg* che riporta il messaggio testuale dell'sms inviato. Questi attributi sono stati confrontati con le feature dei database degli interventi *DATA_INTERVENTO* e *ORA_CHIAMATA*, che riportano rispettivamente il giorno e l'ora in cui è avvenuto l'intervento, *DESCRIZIONE* e *DETTAGLIO_TIPOLOGIA*, che riportano una descrizione dell'intervento, *DESC_COMUNE* e *SIGLA_PROVINCIA*, che riportano rispettivamente il comune e la provincia in cui si verifica l'evento. Una volta appurato se per un intervento è stato inviato oppure no un messaggio sms saremo in grado di sapere se tale intervento è rilevante oppure no. Infatti come già detto un evento risulta essere rilevante se per il relativo intervento è stato inviato un sms, *sms_invito* rappresenterà quindi l'attributo contenente la classe del nostro dataset.

Dal momento che il nostro modello implementa l'ultima parte del sistema a cascata in Figura 2.1, andiamo a selezionare fra tutti gli interventi quelli che sono stati precedentemente flaggati dall'operatore umano come *interessanti*, tale flag si trova sotto l'attributo *INTERVENTO_RILEVANTE*, gli altri interventi non saranno più considerati nel proseguimento del nostro lavoro.

Per ogni intervento restante andiamo a calcolare una serie di feature a partire dal database degli interventi e dal database dell'Istat. Tali feature sono:

- *durata_h* calcolata a partire dagli attributi *DATA_CHIAMATA*, *ORA_CHIAMATA* e *DATA_CHIUSURA*;
- *zonaaltimetrica*, *altitudinecentro*, *comunelitoraneo*, *comunemontano*, *superficieterritoriale* ed infine *gradourbanizzazione*, descritte in Tabella 3.4. Tali feature sono state prese dal database dell'Istat e sono state aggiunte facendo riferimento all'attributo *DESC_COMUNE* del database degli interventi;
- *numeroaggiornamenti*, calcolata tramite l'attributo *INTERVENTO*. Infatti dal momento che per ogni aggiornamento dell'operazione dei vigili del fuoco viene generata una nuova riga nel database degli interventi, tramite l'attributo *INTERVENTO*, che definisce l'unità atomica di un'operazione, andiamo a calcolarci il numero di aggiornamenti di un intervento;
- Infine una nuova feature, *tipozona*, è calcolata a partire da *X* e *Y*, rispettivamente longitudine e latitudine del luogo in cui si svolge un'operazione. Questa operazione sarà spiegata successivamente nel Capitolo 4.

Concludendo, abbiamo ottenuto un dataset contenente 270 campioni descritti da tredici feature, rispettivamente: *momentodelgiorno*, *descrizione*, *enti_intervenuti*, *genere primo mezzo inviato*, *zonaaltimetria*, *altitudinecentro*, *comunelitoraneo*, *tipo di comune montano*, *superficie territoriale*, *gradourbanizzazione*, *numeroaggiornamenti*, *durata_h* e infine *tipozona*. La probabilità di ciascuna classe del dataset è leggermente sbilanciata, dal momento che questo presenta 153 campioni etichettati come *non rilevanti* e i restanti 97 come *rilevanti*.

Capitolo 4

Classificazione immagini satellitari

In questo capitolo parleremo del lavoro svolto riguardo la classificazione delle immagini satellitari. Lo scopo di questo lavoro è di aggiungere al dataset finale un nuovo attributo, denominato *tipo_zona*, che rappresenta il tipo di zona in cui è avvenuto l'intervento. I tipi di zona che andremo a considerare sono *zona agricola*, *zona industriale* e *zona urbana*.

Grazie alle coordinate geografiche *X* e *Y*, rispettivamente longitudine e latitudine, acquisite tramite gli attributi del database degli interventi vogliamo assegnare al luogo in questione una delle tre classi possibili. In Figura 4.1 si possono vedere i tre diversi tipi di zone possibili.

L'aggiunta di questo nuovo attributo si basa sulla volontà di avere una conoscenza più approfondita e granulare del territorio su cui si svolge l'intervento rispetto alle informazioni acquisite tramite le statistiche istat descritte sopra. In Figura 4.2 vediamo il diagramma che mostra il procedimento da svolgere per creare il classificatore supervisionato multiclass che sarà utilizzato per la classificazione delle immagini satellitari.

4.1 Raccolta dati

Le immagini satellitari utilizzate per la creazione del dataset impiegato per lo sviluppo del modello di classificazione sono state acquisite tramite l'API *Google Static Maps*.

Abbiamo scelto di acquisire 100 immagini per classe, quindi avendo tre classi in cui etichettare le immagini, abbiamo ottenuto un database composto da 300 immagini. I luoghi sono stati scelti sulla base delle *zone territoriali omogenee*

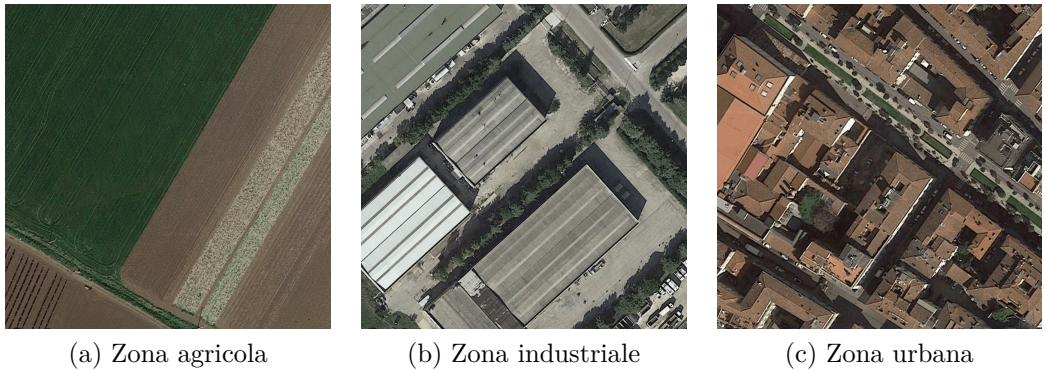


Figura 4.1: Tipi di zone possibili

(*ZTO*) decise dal *piano regolatore generale* emanato a livello comunale. Alla fine della fase di acquisizione delle immagini avremo un database contenente 300 immagini 800×800 *RGB* in formato *png*.

4.2 Pre-elaborazione

Nella fase di pre-elaborazione andremo ad organizzare il dataset per prepararlo alla fase di apprendimento.

Innanzitutto riorganizziamo il database originario delle immagini raccolte in tre nuovi database:

Database 1 Il database 1 contiene tutte e 300 le immagini originali ridimensionate ad una dimensione 250×250 .

Database 2 Contiene tutte e 300 le immagini convertite in toni di grigio, ridimensionate a 250×250 e successivamente private del logo di *Google* presente nella parte inferiore dell'immagine, quindi le immagini finali avranno una dimensione 235×235 .

Database 3 Contiene tutte e 300 le immagini, convertite in scala di grigi e private del logo di *Google*, quindi le immagini finali avranno una dimensione 760×760 .

4.2.1 Feature extraction

Una volta organizzati i tre diversi database contenenti le immagini andiamo ad estrarre da quest'ultime diversi tipi di feature descritte in Sezione 2.6.

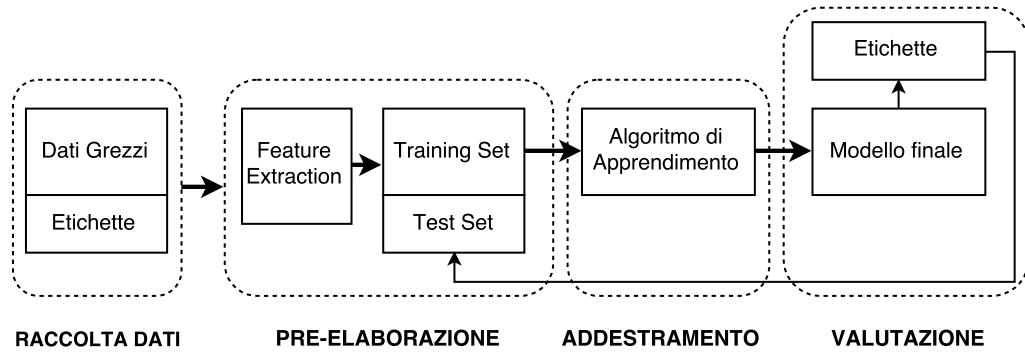


Figura 4.2: Schema a blocchi del procedimento di classificazione delle immagini satellitari

Le diverse feature estratte serviranno successivamente per l’addestramento di altrettanti classificatori che saranno valutati e confrontati fra di loro.

Valori pixel

Dal database 1, contenente immagini *RGB* con una dimensione 250×250 , andiamo ad estrarci delle feature che rappresentano il valore dei pixel contenuti nell’immagine.

Passando alla rappresentazione matriciale di un’immagine, per ogni immagine del database avremo una matrice 3-dimensionale con dimensioni $250 \times 250 \times 3$. Ora andiamo a concatenare ogni riga di questa matrice in un nuovo vettore come visto nell’Algoritmo 1, questo vettore rappresenta il vettore delle feature della singola immagine. Quindi il vettore delle feature estratto è composto dai valori di tutti i pixel che compongono un’immagine.

Facendo questa operazione per tutte e 300 le immagini che compongono il database otteniamo una nuova rappresentazione del database, dove ogni riga rappresenterà i vettori delle feature appena estratti dalle immagini. Il dataset risultante è composto da 300 istanze e 187500 feature e il file CSV relativo ha una dimensione di 190,1 MB. Per ridurre la dimensionalità di questo dataset applichiamo la PCA sull’intero dataset, alla fine otteniamo un dataset descritto

Algorithm 1

```

1: procedure FROMMATRIXTOVECTOR(image)
2:   inizializza vector
3:   for i in range(800) do
4:     for j in range(3) do
5:       appendi image[i,:,j] a vector
6:   return vector

```

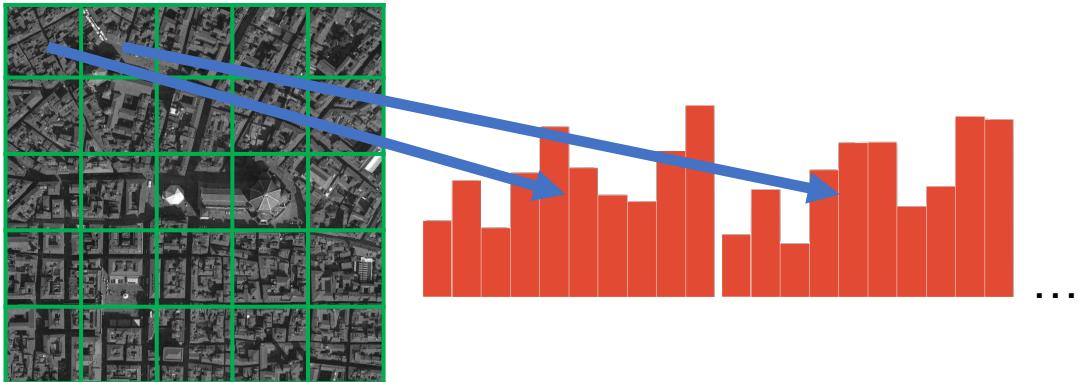


Figura 4.3: Estrazione vettore feature

da 20 feature e con una dimensione di 123 KB.

Local Binary Patterns

Per il database 2, contenente immagini in toni di grigio di dimensione 235×235 , andiamo ad applicare l'operatore LBP discusso nella Sezione 2.6.1.

Ogni immagine del database viene divisa in 25 celle e ad ognuna di queste celle viene applicato l'operatore $LBP_{8,1}^{riu^2}$ con intorni locali $(8, 1)$, quindi come già detto avremo 10 possibili etichette diverse per ogni cella.

Il vettore delle feature per ogni immagine sarà rappresentato dalla concatenazione dei 25 diversi istogrammi ottenuti da ogni cella dell'immagine, questo vettore quindi presenterà 250 feature, 10 per cella. Lo sviluppo del vettore delle feature si può vedere in Figura 4.3.

Descrittori di Haralick

Infine per database 2 andiamo anche ad estrarre i descrittori di Haralick calcolati a partire dalle matrici di co-occorrenza.

Per prima cosa, per ogni immagine del database calcoliamo la matrice \mathbf{P} , data dalla somma delle matrici trovate applicando all'immagine l'operatore GLCM normalizzato parametrizzato volta per volta con $(1, 0^\circ)$, $(1, 45^\circ)$, $(1, 90^\circ)$ e $(1, 135^\circ)$.

Il calcolo della matrice \mathbf{P} è mostrato di seguito:

$$P = \begin{bmatrix} \ddots & & \\ & P_{i,j} & \\ & & \ddots \end{bmatrix}_{(1,0^\circ)} + \begin{bmatrix} \ddots & & \\ & P_{i,j} & \\ & & \ddots \end{bmatrix}_{(1,45^\circ)} + \begin{bmatrix} \ddots & & \\ & P_{i,j} & \\ & & \ddots \end{bmatrix}_{(1,90^\circ)} + \begin{bmatrix} \ddots & & \\ & P_{i,j} & \\ & & \ddots \end{bmatrix}_{(1,135^\circ)}$$

Una volta calcolata la matrice \mathbf{P} per un’immagine andiamo ad estrarre i sei descrittori di Haralick descritti nella Sezione 2.6.2.

Il vettore delle feature di un’immagine sarà rappresentato quindi dai sei descrittori, rispettivamente: contrast, dissimilarity, homogeneity, energy, correlation e ASM.

Alla fine di questa operazione svolta sull’intero database otteniamo un dataset composto da 300 istanze e sei feature.

Ultimo passaggio da fare è di codificare le etichette delle tre classi, *zona agricola*, *zona industriale* e *zona urbana*, come valori interi, questo può essere fatto utilizzando un semplice approccio con mappaggio a dizionario, dove ad ogni classe è associato un valore intero, nel nostro caso:

- Alla classe *zona agricola* è associato il valore 0;
- Alla classe *zona industriale* è associato il valore 1;
- Alla classe *zona urbana* è associato il valore 2;

Concludendo, dai tre database contenenti versioni diverse delle stesse immagini, abbiamo estratto diverse feature, ciascuna riportante informazioni diverse. In questo modo, grazie alle diverse feature estratte, abbiamo creato diverse rappresentazioni dei database. Il prossimo compito sarà quello di addestrare diversi classificatori, utilizzando le feature estratte nella fase di pre-elaborazione, che poi confronteremo fra di loro.

Un riassunto dei dataset di cui abbiamo appena parlato si può vedere in Tabella 4.1.

Prima di procedere alla fase di addestramento del nostro modello andiamo a dividere il dataset in modo casuale tra training set e test set. Il 40% dei campioni faranno parte del test set i restanti del training set.

Database	Numero istanze	Numero feature	Tipo feature
DB-1	300	20	Iistogramma LBP
DB-2	300	250	Iistogramma LBP
DB-2	300	6	Descrittori di Haralick

Tabella 4.1: Riepilogo dei dataset ricavati dalle immagini satellitari

4.3 Addestramento

In questa sezione andiamo ad addestrare il nostro classificatore supervisionato multi-classe in grado di classificare un’immagine satellitare come *zona agricola*, *zona industriale* oppure come *zona urbana*.

La parte di programmazione è stata svolta utilizzando il linguaggio di programmazione *Python* [18], in ambiente *IPython* [15]. Python è uno dei linguaggi di programmazione più utilizzati e noti per l’elaborazione dei dati e gode di una vasta comunità che ha contribuito a sviluppare numerose librerie open source. Le librerie che sono state utilizzate maggiormente per il nostro lavoro sono *Numpy*[24], per la gestione numerica dei dati, e *Scikit-learn*[14], libreria open source che contiene una serie di algoritmi per l’apprendimento automatico.

Come algoritmo di apprendimento abbiamo scelto di utilizzare il Random Forest, della libreria Scikit-learn, addestrato tramite le diverse rappresentazioni dei database, così da confrontare successivamente quali tipi di feature sono più adatte al nostro compito. L’ottimizzazione degli *iperparametri* dei nostri modelli è stata effettuata tramite una *ricerca a griglia*.

Gli iperparametri di un modello sono quei parametri di configurazione del modello stesso. Gli iperparametri di una Random Forest possono essere ad esempio il numero di alberi nella foresta o il criterio di entropia scelto. La ricerca a griglia è una ricerca esaustiva nella quale si specifica una lista di vari valori degli iperparametri del modello per poi valutare le prestazioni del modello per ogni combinazione di questi parametri fino a trovare la combinazione ottimale. La ricerca a griglia è stata fatta utilizzando l’oggetto *GridSearchCV* della libreria Scikit-learn.

Tramite la ricerca a griglia risulta che l’utilizzo di 200 alberi nella Random Forest e dell’Entropia come criterio di split rappresentano la migliore combi-

Tipo Algoritmo	Nome DB	Feature	Acc.	Rec.	Pre.	F1 score	F2 score
Random forest	DB-1	Pixel	0.8750	0.8722	0.8736	0.8721	0.8720
Random forest	DB-2	LBP	0.8750	0.8748	0.8776	0.8746	0.8743
Random forest	DB-2	Descrittori Haralick	0.9250	0.9261	0.9279	0.9267	0.9263

Tabella 4.2: Performance dei diversi classificatori.

Acc.: Accuracy; Rec.: Recall; Pre.: Precision

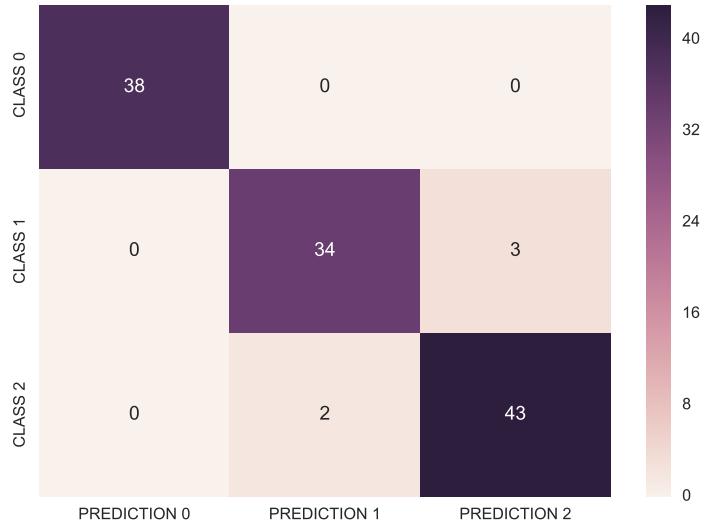


Figura 4.4: Matrice di confusione modello addestrato con dataset GLCM-2

nazione di iperparametri tra quelli testati.

4.4 Valutazione

Le performance dei vari modelli sono riportate in Tabella 4.2, per ogni classificatore le metriche che sono state calcolate sono l'accuracy, il recall, la precision, l'F-score e l'F-2 score. Come si può vedere la Random Forest addestrata con feature estratte dal database 2 possiede delle performance leggermente superiori rispetto agli altri due modelli addestrate con tipi feature diverse. Quindi i descrittori di Haralick, oltre ad avere tempi di calcolo ridotti, si presentano come il tipo di feature migliore, fra i tre proposti, per il nostro compito di classificazione.

Ora cerchiamo di migliorare il nostro modello basato su feature estratte dall'operatore GLCM. Consideriamo un nuovo database, che sarà identificato come il *database 3*, che contiene le 300 immagini originali, convertite in scala di grigi e private del logo di *Google*, le immagini avranno quindi una dimensione di 760×760 . Da questo database andiamo ad estrarre, come abbiamo fatto precedentemente, i descrittori di Haralick calcolati dalle matrici di co-occorrenza dei livelli di grigio.

Tramite questa nuova rappresentazione del database 3 andiamo ad addestrare una Random Forest che sarà successivamente ottimizzata tramite una ricerca a

Tipo Algoritmo	Nome DB	Feature	Acc.	Rec.	Pre.	F1 score	F2 score
Random forest	DB-2	Descrittori Haralick	0.9250	0.9261	0.9279	0.9267	0.9263
Random forest	DB-3	Descrittori Haralick	0.9583	0.9581	0.9597	0.9588	0.9584

Tabella 4.3: Performance dei classificatori basati sui descrittori di Haralick

griglia. In Tabella 4.3 possiamo vedere le performance dei due modelli, basati sui descrittori di Haralick, a confronto. Notiamo subito che le performance di quest’ultimo modello sono leggermente migliorate rispetto alla versione precedente del modello. In Figura 4.4 possiamo vedere la matrice di confusione dell’ultimo modello trovato.

Concludendo, tramite la creazione di questo classificatore saremo capaci, data la latitudine e la longitudine, di etichettare la zona in cui è avvenuto l’intervento come una *zona agricola*, una *zona industriale* oppure una *zona urbana*. Con quest’operazione, come già detto, avremo un’informazione maggiore riguardo al territorio rispetto alle informazioni ricavate dal database rilasciato dall’Istat. L’Algoritmo 2 sarà quindi quello utilizzato per classificare una zona in una delle tre possibili classi.

Algorithm 2

```

1: procedure CLASSIFICAZIONEIMMAGINI(lat, long)
2:   image  $\leftarrow$  getimage(lat, long)
3:   elimina scritta Google da immagine
4:   converti immagine in toni di grigio
5:   vectorfeature  $\leftarrow$  estraidescrittoridiHaralick(image)
6:   prediction  $\leftarrow$  classificatore(feature)
7:   if prediction == 0 then
8:     la zona è agricola
9:   if prediction == 1 then
10:    la zona è industriale
11:   if prediction == 2 then
12:     la zona è urbana
13:   return prediction

```

Capitolo 5

Metodi

Nella prima parte di questo capitolo analizziamo qualche statistica estratta dal nostro dataset, discutendo successivamente di come le varie feature sono distribuite nello spazio e sono correlate fra di loro. Nella parte finale del capitolo esponiamo alcune considerazioni riguardo ai metodi da applicare sulla base delle statistiche fatte precedentemente. Infine descriviamo come applicare i vari approcci descritti nel Capitolo 3 al nostro problema.

5.1 Statistiche dataset

In questa sezione estrarremo delle statistiche interessanti su cui, successivamente, si baseranno le nostre scelte riguardo al modello e agli approcci da utilizzare. Per prima cosa andiamo ad eliminare 20 istanze, poichè queste non presentano l'attributo *tipo zona*, mancando nell'intervento gli attributi *X* e *Y*. Il dataset a nostra disposizione presenta quindi 250 istanze descritte da 13 feaute.

La maggior parte dei descrittori a nostra disposizione sono di tipo categorico, queste sono: *momento del giorno*, *descrizione*, *enti_intervenuti*, *genere primo mezzo inviato*, *zonaaltimetrica*, *comunelitoraneo*, *tipo di comune montano*, *gradourbanizzazione* e *tipozona*. Le restanti *altitudinecentro*, *superficieterritoriale*, *numeroaggiornamenti* e *durata_h* sono di tipo numerico.

	Numero istanze	Numero feature	Feature categoriche	Interventi non rilevanti	Interventi rilevanti
Dataset	250	13	9	153	97

Tabella 5.1: Descrizione dataset

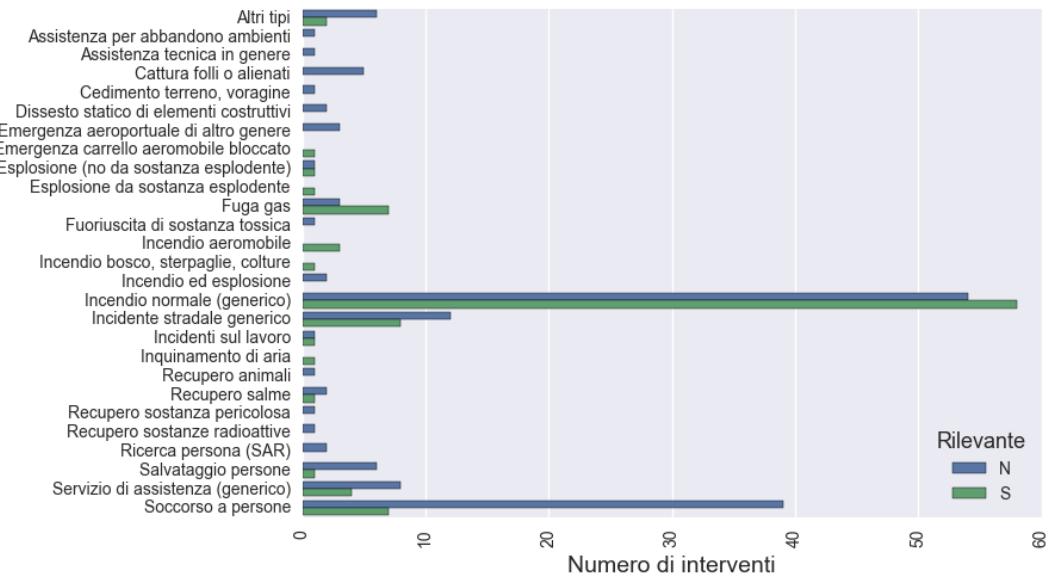


Figura 5.1: Distribuzione degli interventi in base alla tipologia dell’operazione

Il dataset presenta anche delle classi leggermente sbilanciate tra di loro dal momento che gli interventi *non rilevanti* sono circa una volta e mezzo quelli *rilevanti*. In Tabella 5.1 sono sintetizzate le informazioni del nostro dataset.

La Figura 5.1 mostra la distribuzione degli interventi *rilevanti* e *non rilevanti* sulla base del tipo di evento verificatosi. Dalla figura apprendiamo che la maggior parte degli interventi a nostra disposizione riguardano principalmente tre tipologie di eventi: incendi normali, incidenti stradali oppure soccorsi a persone. Dalla figura inoltre si evince che quasi nessuna tipologia è discriminante per classificare un intervento come rilevante. Per alcuni interventi, come ad esempio ‘cattura folli o alienati’, ‘ricerca persona (SAR)’ oppure ‘dissesto stati-

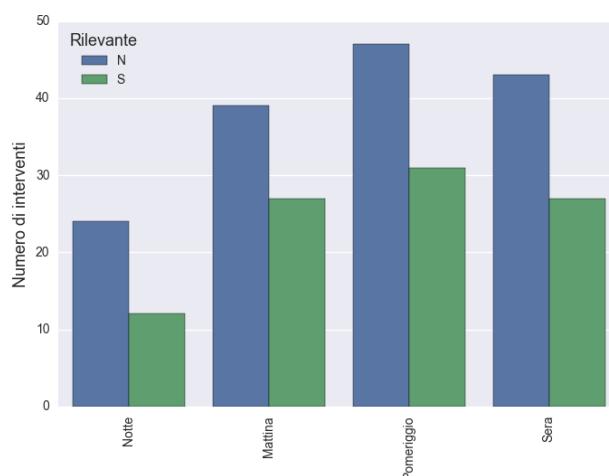


Figura 5.2: Distribuzione degli interventi in base al momento del giorno

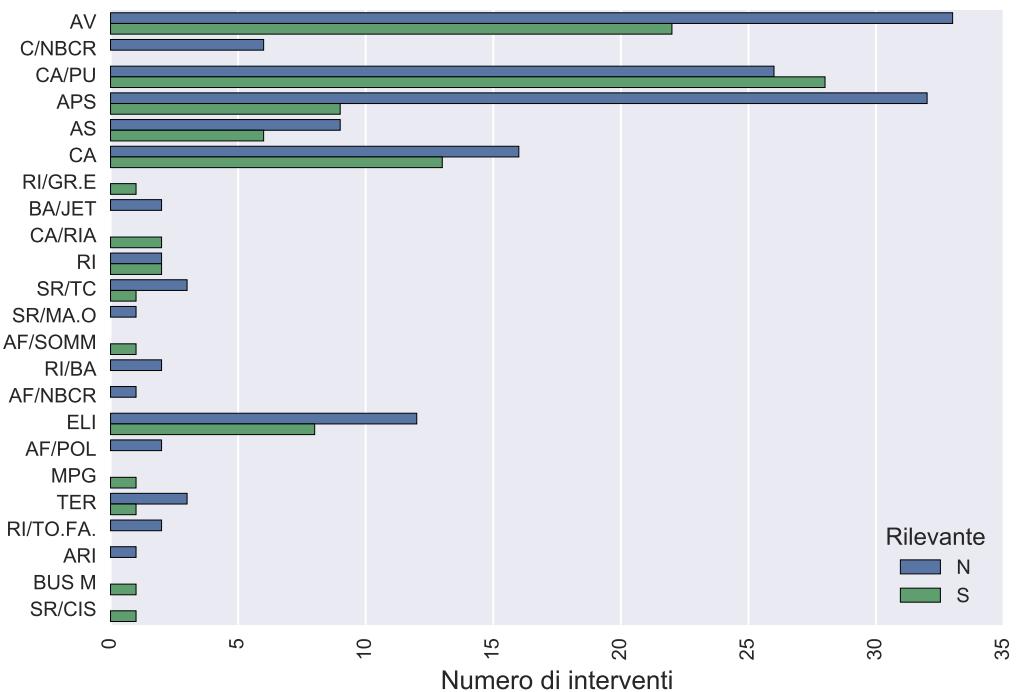


Figura 5.3: Distribuzione degli interventi in base al primo tipo di mezzo viene inviato

co di elementi costruttivi', abbiamo solamente interventi *non rilevanti*, quindi a prima vista potrebbe sembrare che quella tipologia ammetta solamente interventi non rilevanti, ma va tenuto presente che abbiamo a disposizione una piccola quantità di dati per poter dare per assodato tale affermazione. La quasi totalità delle tipologie presentano solamente una decina di campioni. L'unica tipologia che mostra un netto contrasto fra il numero di interventi rilevanti e non rilevanti è il soccorso a persone dove su 46 interventi più del'84% sono non rilevanti.

La Figura 5.2 mostra la distribuzione degli interventi sulla base del momento del giorno, *notte*, *mattina*, *pomeriggio* e *sera*. Si vede che la maggior parte delle operazioni si svolge durante la parte centrale e finale del giorno, ma la distribuzione del tipo di intervento resta pressoché uguale nell'arco della giornata.

La Figura 5.3 mostra la distribuzione degli interventi sulla base della tipologia del primo mezzo mandato sul posto ad occuparsi dell'evento. In Appendice abbiamo l'anagrafica di tutti i generi di mezzo a disposizio dei vigili del fuoco. Da questo grafico si evince che la distribuzione della rilevanza degli interventi è pressoché identica rispetto alla tipologia di mezzo inviato. L'unico tipo di

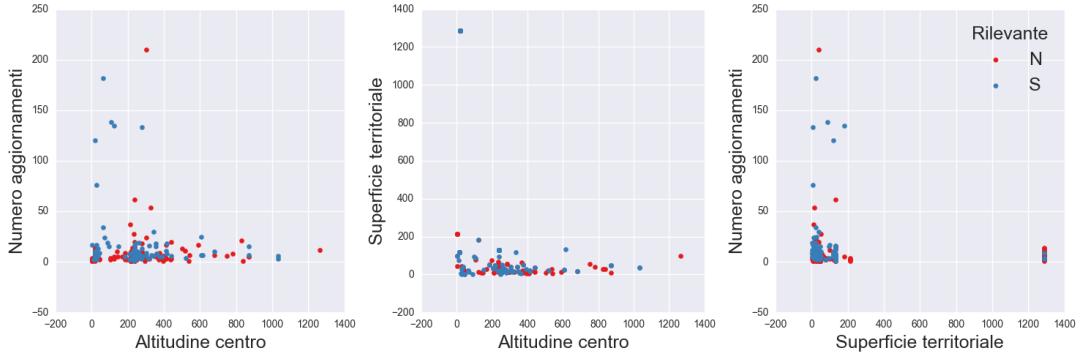


Figura 5.4: Distribuzione nello spazio delle feature numeriche

mezzo, utilizzato almeno cinque volte, che presenta solamente interventi non rilevanti risulta essere *C/N BCR*, inviato in totale 6 volte. Comunque il numero di campioni è troppo ridotto per trarre una regola generale.

Prima di passare alla rappresentazione spaziale delle feature dobbiamo codificare gli attributi categorici in valori interi. Questa codifica servirà anche successivamente durante la fase di addestramento dal momento che gran parte degli estimatori per la classificazione di scikit-learn non ammettono in ingresso feature di tipo categorico. Anche per la Random Forest, che accetta feature di tipo categorico, non esiste un’implementazione in Python che accetti in modo automatico feature di tipo categorico. Comunque tramite uno studio abbiamo appurato che la scelta del valore numerico associato alla feature categorica non introduce alcun bias nella costruzione del Random Forest e nelle prestazioni. La Figura 5.4 mostra come sono distribuite spazialmente le varie feature numeriche. Si vede subito che non esiste un confine decisionale semplice in grado di separare le due classi dal momento che le quest’ultime sono completamente sovrapposte.

5.2 Modello

Sulla base delle considerazioni appena fatte ipotizziamo che il Random Forest, trattato in Sezione 2.7.2, possa essere l’approccio che meglio si adatta al nostro problema. Questa scelta è stata fatta anche in base al fatto che il Random Forest riesce a gestire in modo automatico feature di tipo categorico. La Figura 5.5 mostra la pipeline finale di classificazione: i vari metodi presenti in essa vengono spiegati in dettaglio nel resto del capitolo.

Prima di procedere all’addestramento del classificatore operiamo una feature selection per selezionare le feature più significative con cui addestrare il nostro

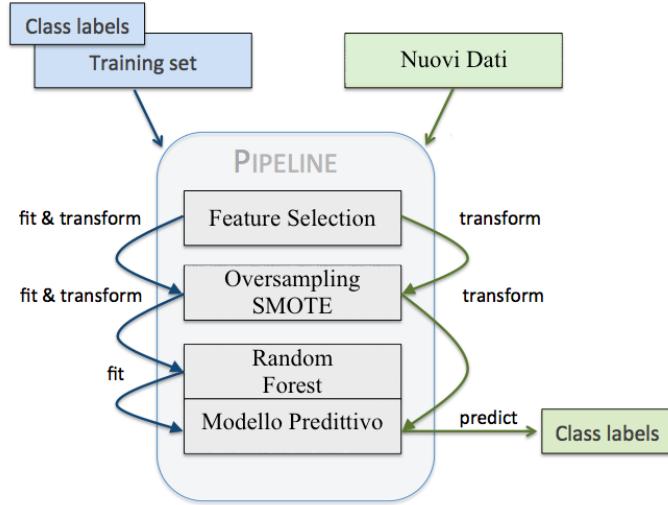


Figura 5.5: Pipeline finale del nostro modello

modello. L’operazione di feature selection è stata svolta applicando il metodo *CBFS* [22], *high performance feature selection algorithm based on feature clearness*, il quale va a selezionare la migliore combinazione di feature in base alla loro capacità nel separare le classi. Operando una feature selection tramite il metodo CBFS andiamo ad eliminare dai dati quella componente di rumore che influenza in maniera negativa le performance globali del nostro classificatore. Dopo una serie di prove su vari algoritmi di apprendimento il Random Forest si è rivelato essere il migliore modello, confermando in questo modo le nostre ipotesi. Tuttavia, durante le varie analisi, si è notato che le varie performance dei nostri algoritmi erano fortemente influenzati dalla scelta del training e test set. Questo problema ipotizziamo possa essere causato dalle ridotte dimensioni del dataset. Per rimediare a questo problema la procedura di addestramento e di valutazione del modello sarà effettuata tramite un metodo di *cross-validation* noto come *Leave-One-Out*.

Il metodo *Leave-One-Out*, *LOO*, rappresenta un tipo di convalida utilizzata solitamente per modelli che si basano su dataset di ridotte dimensioni. Questo metodo opera nel seguente modo: consideriamo di avere un dataset \mathbf{X} composto da n istanze, andiamo a dividere il nostro dataset in $k = n$ parti e durante le n iterazioni addestriamo il nostro modello con $n - 1$ istanze, mentre il campione restante viene utilizzato per il test. Le prestazioni finali del modello sono state calcolate sommando tutte le matrici di confusione ricavate dagli n campioni di prova. Successivamente, dalla matrice di confusione finale abbiamo calcolate tutte le metriche che ci interessavano. In questo modo il modello è testato esattamente una volta su tutti i campioni del dataset \mathbf{X} . Il

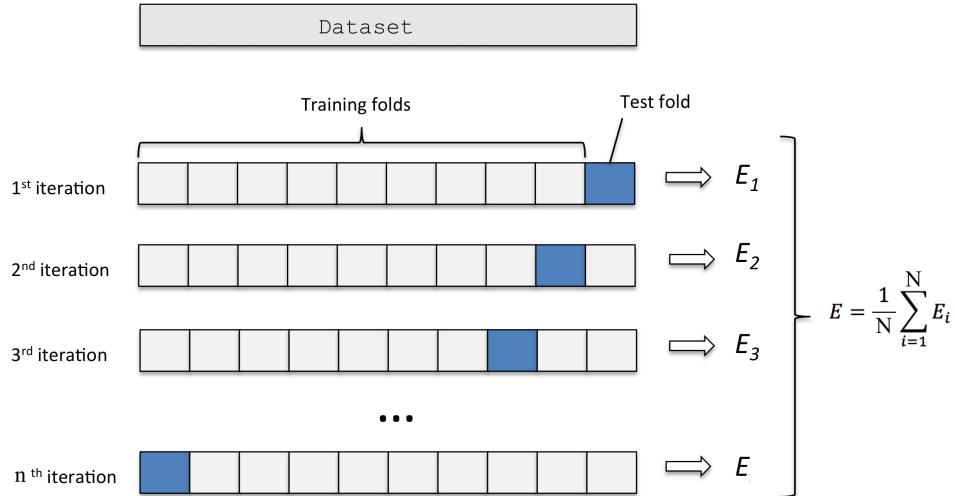


Figura 5.6: Convalida Leave-One-Out

metodo LOO rappresenta un caso speciale della convalida incrociata k -fold in cui il dataset è diviso in k parti uguali. La Figura 5.6 mostra come opera il metodo Leave-One-Out.

Durante le varie analisi è risultato che la classe degli eventi *rilevanti* non veniva classificata ottimamente dal nostro sistema: abbiamo ipotizzato che questo tipo di risultati potrebbe essere causato dal dataset utilizzato per l’addestramento, leggermente sbilanciato verso la classe degli interventi non rilevanti. Il dataset sbilanciato, *unbalanced dataset*, è un problema che si verifica spesso nelle applicazioni reali. Per risolvere questo problema abbiamo deciso di applicare lo SMOTE, *Synthetic Minority Over-sampling Technique*, una tecnica di oversampling utilizzata per equilibrare la probabilità a priori delle classi nel set di addestramento.

Lo *SMOTE* [4] rappresenta una tecnica di oversampling con la quale si va a bilanciare le due classi del dataset sintetizzando nuove istanze della classe minoritaria a partire dai campioni del dataset appartenenti ad essa. L’algoritmo ha un principio di funzionamento abbastanza semplice: per ogni campione appartenente alla classe in minoranza, vengono rilevate i suoi k -vicini campioni appartenenti alla stessa classe, di questi ne vengono selezionati i in maniera random per generare dei campioni sintetici effettuando un join tra questi i vicini e il campione iniziale. Noi utilizzeremo lo SMOTE parametrizzato con $k = 5$ e $i = 5$.

Capitolo 6

Risultati

In questo capitolo presentiamo la valutazione del nostro modello e l'analisi dei vari risultati raccolti.

La Tabella 6.1 mostra i risultati di quattro test diversi identificati dai seguenti acronimi:

- DSC: acronimo che indica che la pipeline finale di classificazione prevede una prima feature selection con il metodo CBFS e una sintesi di nuovi campioni con il metodo SMOTE;
- DS: acronimo che rappresenta una pipeline finale composta dal metodo SMOTE ma priva della feature selection;
- DFS: acronimo che rappresenta una pipeline finale che prevede solamente il metodo CBFS;
- DO: acronimo che indica una pipeline finale priva dei metodi CBFS e SMOTE.

Tutti e quattro i metodi prevedono il Random Forest come algoritmo di classificazione. Dalla Tabella 6.1 si vede che, applicando la feature selection CBFS

Dataset	F ₂ -score	F ₁ -score	Recall	Precision	Accuracy	AUC
DSC	0.5684	0.5869	0.5567	0.6207	0.696	0.7044
DS	0.5682	0.5699	0.5670	0.5729	0.668	0.6909
DFS	0.4936	0.5257	0.4742	0.5897	0.668	0.6812
DO	0.4978	0.5380	0.4742	0.6216	0.684	0.6906

Tabella 6.1: Performance dei vari classificatori.

DO: dataset originale quindi senza SMOTE e CBFS; DFS: dataset con feature selection CBFS; DS: dataset con SMOTE; DSC: dataset con SMOTE e CBFS.

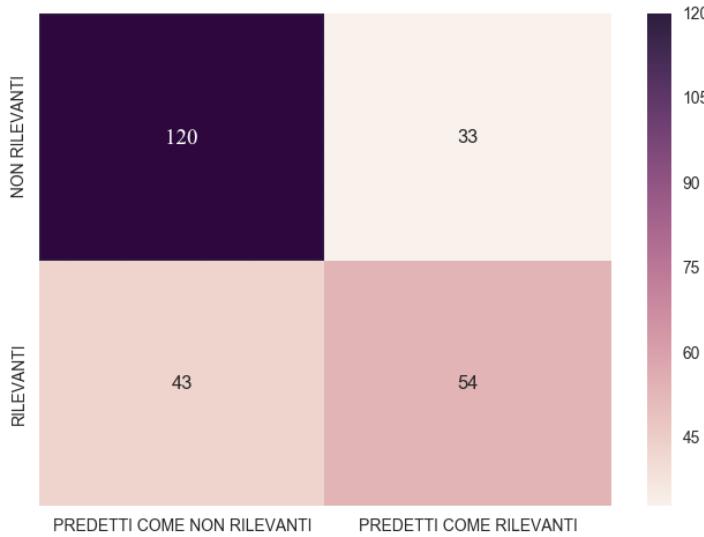


Figura 6.1: Matrice di confusione del nostro modello

e sintetizzando nuovi campioni della classe minoritaria con SMOTE, le performance globali della pipeline di classificazione aumentano. Passando da un classificatore privo del metodo CBFS e SMOTE ad uno con questi due metodi la Recall aumenta di circa il 19%, diminuiscono infatti il numero dei falsi negativi. Infine la pipeline con CBFS e SMOTE è quella che presenta l'AUC maggiore, circa 70%. Quindi da questo momento prendiamo come pipeline di classificazione quella mostrata in Figura 5.5.

Dall'operazione di feature selection otteniamo 8 feature su 13, ovvero *memento del giorno*, *enti_intervenuti*, *genere primo mezzo inviato*, *tipozona*, *altitudinecentro*, *superficie territoriale*, *numeroaggiornamenti* e *durata_h*. I primi quattro descrittori sono di tipo categorico, gli ultimi quattro di tipo numerico. Altra cosa interessante da notare è che l'attributo *tipozona* il cui valore si basa sull'output del classificatore di immagini satellitari, è stato selezionato come una delle feature più rilevanti. La Tabella 6.2 mostra le metriche viste in

Dataset	F ₂ -score	F ₁ -score	Recall	Precision	Accuracy	AUC
DSC	0.5567	0.5730	0.5464	0.6023	0.684	0.6856
DS	0.5498	0.5550	0.5464	0.5638	0.66	0.7073
DFS	0.4329	0.4678	0.4124	0.5405	0.636	0.6626
DO	0.5319	0.5586	0.5155	0.6097	0.684	0.6993

Tabella 6.2: Performance dei vari classificatori senza considerare l'attributo *tipozona*.

DO: dataset originale quindi senza SMOTE e CBFS; DFS: dataset con feature selection CBFS; DS: dataset con SMOTE; DSC: dataset con SMOTE e CBFS.

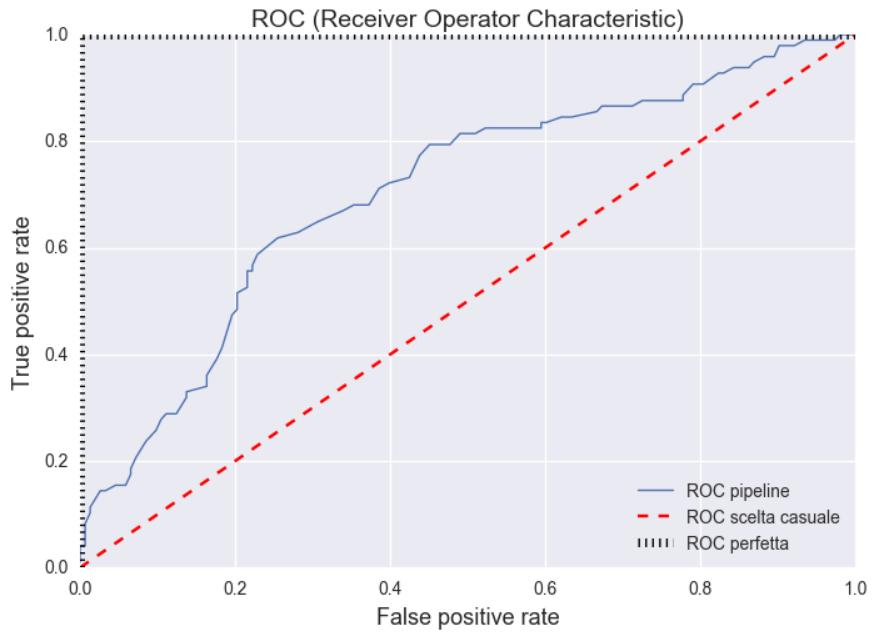


Figura 6.2: Curva ROC del nostro modello

Tabella 6.1 ma senza l'utilizzo dell'attributo *tipozona* durante l'addestramento dei classificatori; notiamo che utilizzando questo attributo le performance del classificatore con SMOTE e feature selection CBFS aumentano leggermente. Infine va notato che 3 delle 8 feature selezionate riguardano la caratterizzazione del territorio su cui è avvenuto l'evento che ha richiesto l'intervento dei vigili del fuoco.

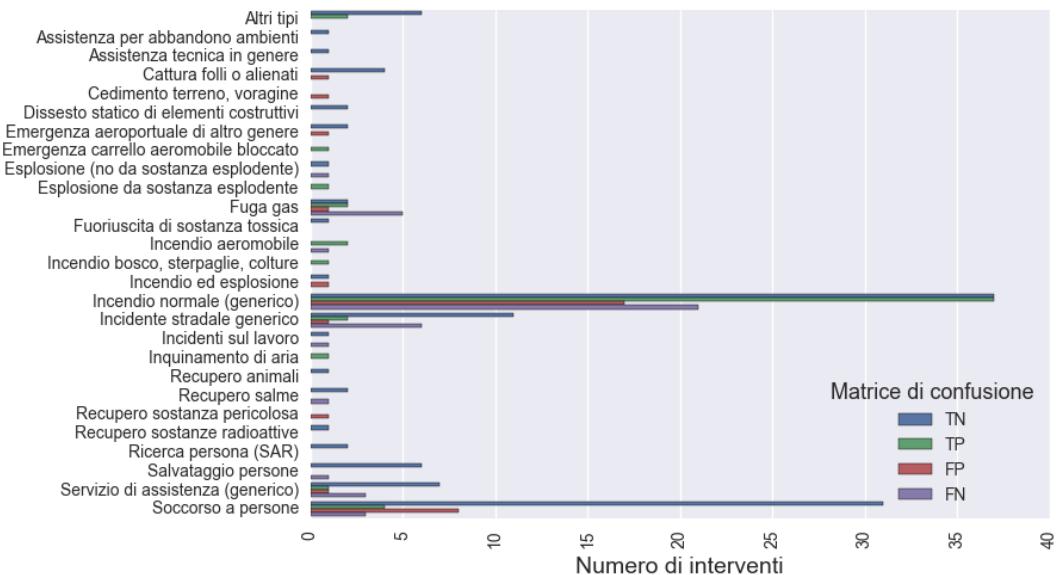


Figura 6.3: Distribuzione della matrice di confusione per tipo d'intervento.
TN: True Negative; TP: True Positive; FP: False Positive; FN: False Negative.

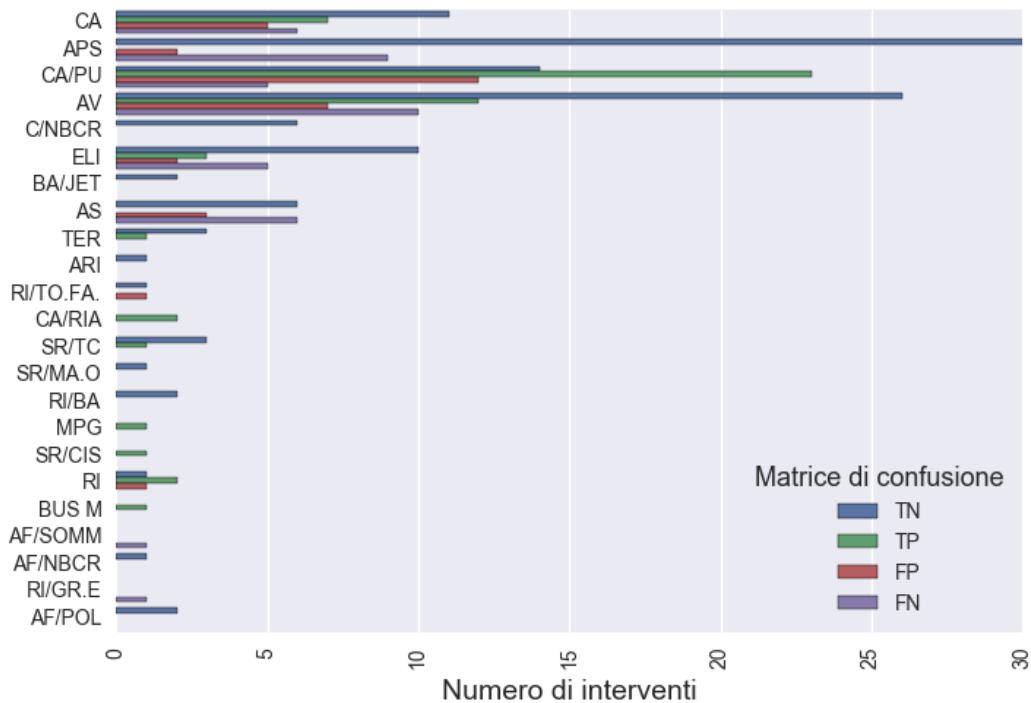


Figura 6.4: Distribuzione delle matrice di confusione per tipo di mezzo.
TN: True Negative; TP: True Positive; FP: False Positive; FN: False Negative.

La Figura 6.1 mostra la matrice di confusione del nostro sistema. Si nota subito che il sistema gestisce bene le istanze appartenenti alla classe degli interventi *non rilevanti* classificando correttamente più del 78% dei campioni. Dall'altra parte il sistema non classifica ottimamente gli eventi *rilevanti*. Abbiamo ipotizzato che questo tipo di risultati potrebbe essere causato dal dataset utilizzato per l'addestramento, leggermente sbilanciato verso la classe degli interventi *non rilevanti* e contenente feature non molto discriminatorie per il nostro tipo di problema.

La Figura 6.2 mostra la curva *ROC*, *Receiver Operator Characteristic*, del nostro modello. I grafici ROC sono strumenti utili per la selezione dei modelli sulla base delle loro prestazioni rispetto ai tassi di falsi positivi (False positive rate) e veri positivi (True positive rate). La diagonale di un grafico ROC può essere interpretata come l'indicazione casuale, quindi i classificatori che rientrano sotto la diagonale sono considerati peggiori rispetto la scelta casuale.

Le Figure 6.3 e 6.4 mostrano la distribuzione della matrice di confusione rispettivamente rispetto la tipologia di intervento e il genere del primo mezzo dei vigili del fuoco inviato a gestire un evento. Dalla prima figura apprendiamo che la tipologia *soccorso a persona* risulta essere quella più correttamente classificata, inoltre sempre da questa figura si evince che la tipologia *incendio*

normale (generico) è quella che presenta il più alto numero di eventi erroneamente classificati. Dalla seconda figura apprendiamo che la per la tipologia di mezzo *APS* la classe degli eventi *rilevanti* è totalmente erroneamente classificata, presentando tutti falsi negativi. Dall'altra parte la tipologia *CA/PU* è quella che presenta la migliore classificazione degli interventi *rilevanti*.

Capitolo 7

Conclusioni

Nel presente lavoro di tesi è stato proposto un sistema che permette il rilevamento automatico degli interventi rilevanti per il Corpo Nazionale dei Vigili del Fuoco. Il sistema da noi sviluppato permetterà ai vigili del fuoco di segnalare, in maniera automatica, gli interventi rilevanti alle autorità competenti.

La soluzione proposta è basata su tecniche e algoritmi del Machine Learning. La pipeline finale di classificazione, riportata in Figura 5.5, prevede una feature selection con il metodo CBFS, il quale seleziona le feature sulla base della loro capacità di separare le classi, e successivamente una sintesi di nuovi campioni della classe minoritaria, con l'algoritmo SMOTE. Sulla base dell'analisi del problema e sul tipo di features si è scelto il Random Forest come algoritmo di apprendimento.

Fra le varie feature estratte dal metodo CBFS e poi utilizzate in fase di addestramento del nostro classificatore è presente l'attributo che serve a descrivere la sede dove ha avuto luogo l'intervento. Per determinare il valore di quest'ultimo descrittore abbiamo costruito un classificatore di immagini satellitari capaci di distinguere tra *zona agricola*, *zona industriale* e *zona urbana*. Le ottime performance ottenute, riportate nel Capitolo 4, e il fatto che, diversamente da altri metodi presenti in letteratura, il nostro classificatore utilizzi dati disponibili pubblicamente, ci porta ad ipotizzare che questo lavoro potrebbe essere portato avanti come lavoro a sé stante. Sviluppi futuri di questo progetto potrebbero riguardare lo studio della distribuzione degli insediamenti urbani di un territorio oppure il confronto di nuove feature per migliorare ulteriormente le performance del classificatore.

Concludendo, i risultati del nostro sistema di rilevamento automatico degli interventi rilevanti mostrano che il nostro approccio riesce a rilevare tali eventi con un tasso di errore raggiunto non trascurabile. Questo sembra esse-

re causato dalla vera natura del problema e dalle feature di tipo categorico utilizzate, queste due cause potrebbero influenzare negativamente il tasso di riconoscimento.

I risultati e le conoscenze raccolte sul problema suggeriscono due strade su cui continuare il nostro progetto:

- Mantenere un sistema simile alla pipeline attuale, andando però ad estrarre e ad analizzare il potere discriminante di ulteriori feature. Queste potrebbero riguardare le condizioni meteorologiche oppure un'analisi dei vari social network per ottenere informazioni riguardo la popolazione e il comportamento umano in un dato luogo;
- Cambiare completamente approccio indirizzandoci verso modelli basati sulle macchine a stati-finiti, descritte brevemente successivamente. Supponiamo che le macchine a stati-finiti riescano meglio a cogliere la dinamicità e la storia dell'intervento per via del grande numero di feature categoriche a nostra disposizione, 7 descrittori su 13, e per il tipo di problema che abbiamo, dove la situazione di un intervento può evolvere nel tempo, passando da *non rilevante* a *rilevante*.

Le *macchine a stati-finiti* (dall'inglese Finite State Machines, *FSM*), anche chiamate *automi a stati finiti* (dall'inglese Finite State Automatas, *FSA*) rappresentano un metodo per la modelizzazione di sistemi il cui output dipende dall'intera storia passata degli input al sistema. Una macchina a stati-finiti può essere utilizzata sia per modellare un sistema esistente sia per modellare un nuovo sistema in grado di risolvere alcuni problemi esistenti. La rappresentazione grafica di un automa a stati-finiti è un *grafo*.

Le macchine a stati-finiti possono modellizzare tutti quei sistemi che presentano le seguenti caratteristiche:

- **Dinamicità:** un sistema può evolvere nel tempo, passando da uno stato all'altro;
- **Discretezza:** un sistema può essere rappresentato utilizzando variabili in ingresso e stati espressi da valori discreti;
- **Simboli finiti:** caratteristica che determina che il numero di simboli di ingresso e di stati sia rappresentabile da un numero finito.

Quindi gli automi a stati-finiti possono essere utilizzati per rappresentare una vasta varietà di sistemi, inclusi:

- Interfacce utente con input digitato, come ad esempio il mouse o la tastiera;
- Conversazioni, nelle quali, per esempio, il significato della parola ‘ciao’ dipenda dalla storia passata delle cose che sono state detto;
- Lo stato di un aereo, incluso i livello di carburante e di ossigeno.

Appendice

In questa appendice riportiamo l'anagrafica di tutti i generi di mezzo a disposizione dei vigili del fuoco.

Codice Mezzo	Descrizione Mezzo
A/AER	AUTOMEZZO SOCCORSO AEROPORTUALE
A/BUS	AUTOBUS
A/SP	AUTOMEZZO SPAZZATRICE
A/TRID	AUTOMEZZO TRIDIMENSIONALE
A/TT	AUTOMEZZO TRATTRICE PER SEMIRIMORCHIO
AA	AUTOMEZZO ANFIBIO
AB	AUTOBOTTE
AB/CAR	AUTOBOTTE CARBURANTE
AB/GPL	AUTOBOTTE PER TRASPORTO GPL
ABP	AUTOBOTTEPOMPA
ABP/AER	AUTOBOTTEPOMPA SERVIZIO AEROPORTUALE
ABP/SC	AUTOBOTTEPOMPA ATTREZZATA SCARRABILE
ACT	AUTOCARRO TRASPORTO
ACT/ATTR	AUTOCARRO SERVIZI ISTITUTO
ACT/BOSC	AUTOCARRO INCENDI BOSCHIVI
ACT/CC	AUTOCARRO CROLLI
ACT/CO2	AUTOCARRO CON APPARECCHIATURA CO2
ACT/EL	AUTOCARRO SERVIZI ELETTRICI
ACT/FURG.SC	AUTOCARRO FURGANATO SCARRABILE
ACT/GRU	AUTOCARRO GRU
ACT/IDR	AUTOCARRO SERVIZI IDRICI
ACT/IG	AUTOCARRO SERVIZI IGIENICI
ACT/ILL	AUTOCARRO ILLUMINAZIONE
ACT/ILL MG	AUTOCARRO ILLUM. MAGAZZINO GENERATORE
ACT/ILL TF	AUTOCARRO ILLUMINAZIONE TORRE FARI
ACT/INC	AUTOMEZZO RECUPERO VEICOLI INCIDENTATI
ACT/LU	AUTOCARRO LUCE - GRUPPO ELETTROGENO
ACT/MEN	AUTOCARRO SERVIZI MENSA
ACT/NBC	AUTOCARRO PER DECON. NUCL. BATTER. CHIM.
ACT/NBCR	AUTOCARRO DECON. NUCL.BATTER.CHIM.RAD.

Codice Mezzo	Descrizione Mezzo
ACT/NUC	AUTOCARRO EMERGENZA NUCLEARE
ACT/OP	AUTOCARRO SEZIONE OPERATIVA
ACT/RIB	AUTOCARRO CASSONE RIBALTABLE
ACT/RU	AUTOCARRO TRASPORTO RU
ACT/SC	AUTOCARRO SCARRABILE
ACT/SCH-CO2	AUTOCARRO CON SCHIUMOGENO CO2
ACT/SMM	AUTOCARRO SATELLITARE MOBILE
ACT/TEN	AUTOCARRO TENDE E CASERMAGGIO
ACT/TRI	AUTOCARRO CASSONE TRIRIBALTABLE
ACT/UFF	AUTOCARRO UFFICIO
AF	AUTOFURGONE CARROZ. CHIUSA
AF/AER	AUTOFURGONE SOCCORSO AEREOPORTUALE
AF/BOSC	AUTOFURGONE INCENDI BOSCHIVI
AF/C	AUTOFURGONE COMBINATO
AF/CC	AUTOFURGONE CARRO COMANDO COL. MOB.
AF/CO	AUTOFURGONE COMANDO
AF/COMBI	AUTOFURGONE COMBI
AF/EL	AUTOFURGONE SERVIZI ELETTRICI
AF/IDR	AUTOFURGONE SERVIZI IDRICI COL. MOB.
AF/MEN	AUTOFURGONE MENSA
AF/NBC	AUTOFURGONE PER DECON. NUCL.BATTER.CHIM.
AF/NBCR	AUTOFURGONE DECON. NUCL.BATTER.CHIM.RAD.
AF/NR	AUTOFURGONE PER NUCLEO RADIOATTIVITA'
AF/NUC	AUTOFURGONE LABORATORIO NUCLEARE
AF/OFF	AUTOFURGONE OFFICINA
AF/OP	AUTOFURGONE OPERATIVO COL. MOB.
AF/P.RAD	AUTOFURGONE PONTE RADIO MOBILE
AF/POL	AUTOFURGONE POLISOCCORSO
AF/RAD	AUTOFURGONE RADIO
AF/RIS	AUTOFURGONE RISTORO COL. MOB.
AF/SAF	AUTOFURGONE SPELEO ALPINO FLUVIALE
AF/SAR	AUTOFURGONE SOCCORSO ACCESSIBILITA' RID.
AF/SC	AUTOFURGONE SCORTA
AF/SMZT	AUTOFURGONE PER NUCLEO SOMMOZZATORI
AF/SOMM	AUTOFURGONE PER NUCLEO SOMMOZZATORI
AF/TRID	AUTOFURGONE TRIDIMENTIONALE
AF/TV	AUTOFURGONE PER RIPRESE TV
AF/UCL	AUTOFURGONE UNITA' CRISI LOCALE
AG	AUTOGRU
AIS	AUTOIDROSCHIUMA
AIS/PB	AUTOIDROSCHIUMA PELINI BARIBBI
AISP	AUTOIDROSCHIUMA POLVERE
AL	AUTOLETTIGA
AP	AUTOPOMPA SENZA SERBATOIO
AP/S.I.EL	AUTOPOMPA SENZA SERB SERV IDR.
APL	AUTOPOMPA LAGUNARE

Codice Mezzo	Descrizione Mezzo
APL/G	AUTOPOMPA LAGUNARE GRANDE
APL/M	AUTOPOMPA LAGUNARE MEDIA
APL/P	AUTOPOMPA LAGUNARE PICCOLA
APS	AUTOPOMPA SERBATOIO
APS/3P	AUTOPOMPA SERBATOIO 3 POSTI
APS/P	AUTOPOMPA SERBATOIO PICCOLA
APS/SR	AUTOPOMPA SERB. STRADA ROTAIA (BIMODALE)
APS/TRID	AUTOPOMPA SERBATOIO TRIDIMENSIONALE
ARI	AUTOMEZZO RAPIDO INTERV. AEROPORTUALE
AS	AUTOSCALA
AS/M	AUTOSCALA CON VOLATA MANUALE
ASA	AUTOMEZZO SOCCORSO AEROPORTUALE
ATOM	ATOMIZZATORE
AV	AUTOVETTURA
AV/9	AUTOVETTURA CON 9 POSTI
AV/AFF. ALVAR	AUTOVETTURA AFFITATA ARVAL
AV/FS	AUTOVETTURA FUORISTRADA
AV/NOLEGGIO	AUTOVETTURA NOLEGGIATA
AVP	AUTOVETTURA POMPA
BA	BARCA
BA/JET	BARCA CON MOTORE IDROJET
BP	BATTELLO PNEUMATICO
BP/EB	BATTELLO PNEUMATICO ENTRO BORDO
BUS G	AUTOBUS GRANDE (OLTRE 20 POSTI)
BUS M	AUTOBUS MEDIO (FINO A 20 POSTI)
BUS P	AUTOBUS PICCOLO (FINO A 9 POSTI)
BUS/AL	AUTOBUS TRASFORMATO IN AUTOLETTIGA
C/NBCR	CARRELLO CON IDROPULITRICE
CA	CAMPAGNOLA
CA/ESK	CAMPAGNOLA CON MODULO ESK
CA/ESK 400	CAMPAGNOLA CON MODULO ESK 400
CA/ESK 600	CAMPAGNOLA CON MODULO ESK 600
CA/FO	CAMPAGNOLA CON FOTOELETTRICA
CA/GRU	CAMPAGNOLA CON GRUETTA
CA/MONTES	CAMPAGNOLA FUORISTRADA (9 POSTI)
CA/NUC	CAMPAGNOLA ATTREZZATA NUCLEARE
CA/PU	CAMPAGNOLA PICK-UP
CA/RIA	CAMPAGNOLA RAPIDO INTERV. AEROPORTUALE
CA/SMZT	CAMPAGNOLA PER NUCLEO SOMMOZZATORI
CC/CARB	CISTERNA TRASPORTO CARBURANTI
CE	CARRELLO ELEVATORE
CEI	CESOIE IDRAULICHE
CEL	CARRELLO ELEVATORE
DEC	DECESPUGLIATORE
ELI	ELICOTTERO

Codice Mezzo	Descrizione Mezzo
ES/MINI	MINI ESCAVATORE
ESC	ESCAVATORE CINGOLATO
ESG	ESCAVATORE GOMMATO
FB	FUORIBORDO
FE	FOTOELETTRICA
FRS/NEVE	FRESA DA NEVE
FS	FS
FS/NEVE	YETI E PRINOTH PER ZONE INNEVATE
GE	GRUPPO ELETTROGENO
GE/B	GRUPPO ELETTROGENO BARELLABILE
GF/B	GRUPPO FARI BARELLABILE
GID/B	GRUPPO IDRAULICO BARELLABILE
HOVERCRAFT	HOVERCRAFT
IAL	IMBARCAZIONE ALLUVIONALE LEGGERA
IDRO	IDROPULITRICE GENERICA
IDRO/RIS	IDROPULITRICE RISCALDATA
LUF	SISTEMA MOBILE DI SUPPORTO ANTINCENDIO
MBP/G	MOTOBARCA POMPA GRANDE
MBP/M	MOTOBARCA POMPA MEDIA
MBP/P	MOTOBARCA POMPA PICCOLA
MBP/RAFF	MOTOBARCA POMPA RAFF
MD	MOTO DISCO
MD/B	MOTODIVARICATORE BARELLABILE
MDA	MOTO D'ACQUA
MDEM	MOTODEMOLITORI
MES	MINIESCAVATORE
MF	MOTOFARO
MNP	MOTONAVE POMPA
MO	MOTOCICLETTA
MOD/BOSC SC	MODULO INCENDI BOSCHIVI SCARRABILE
MOS	MOS MACCHINA OPERATRICE SEMOVENTE
MP/B-INC	MOTOPOMPA BARELLABILE DA INCENDIO
MP/ESA-RI	MOTOPOMPA ESAURIMENTO SU RIMORCHIO
MP/RI	MOTOPOMPA INCENDIO SU RIMORCHIO
MPC	MINIPALA CINGOLATA
MPG	MINIPALA GOMMATA
MS	MOTOSCAFEO
MSG	MOTOSEGA
MT	MOVIMENTATORE TELESCOPICO
MTC	MOTOCARRELLO
MTS	MOTOSLITTA
MTT	MOTOTRONCATRICE
MUL	MULETTO - CARRELLO ELEVATORE
MV/B	MOTOVENTILATORE BARELLABILE
MV/RIS	MOTOVENTILATORE RISCALDATORE
P/B-ESA	MOTOPOMPA BARELLABILE DA ESAURIMENTO

Codice Mezzo	Descrizione Mezzo
POM/IMM	POMPA AD IMMERSIONE
PTR	PIATTAFORMA TELESCOPICA RAGNO
PTTF	PIATTAFORMA
QUA	QUADRICICLO
QUAD	QUADRICICLO
RAS	RASAERBA
RI	RIMORCHIO PER USI VARI
RI/BA	RIMORCHIO PER BARCA
RI/BP	RIMORCHIO PER BP
RI/CANN	RIMORCHIO PER CANNONE LANCIA
RI/CING	RIMORCHIO GIRINO PER CINGOLATI
RI/CONT.CU	RIMORCHIO CONTAINERS CUCINA
RI/CONT.IG	RIMORCHIO CONTAINERS SERVIZI IGIENICI
RI/CU	RIMORCHIO CU
RI/CU G	RIMORCHIO CUCINA GRANDE (800 RAZIONI)
RI/CU M	RIMORCHIO CUCINA MEDIA (300 RAZIONI)
RI/CU P	RIMORCHIO CUCINA PICCOLA (200 RAZIONI)
RI/ELI	RIMORCHIO BIGA PER ELICOTTERI
RI/ESK	RIMORCHIO CON MODULO ESK
RI/ESK 400	RIMORCHIO CON MODULO ESK 400
RI/ESK 600	RIMORCHIO CON MODULO ESK 600
RI/FO	RIMORCHIO PER FOTOELETTRICA
RI/FS	RIMORCHIO PER TERRENI INNEVATI
RI/FSN	RIMORCHIO PER MEZZI DA NEVE
RI/GR.E	RIMORCHIO PER GRUPPO ELETROGENO
RI/MA.O	RIMORCHIO TRASPORTO MACC. OPER.
RI/MCP	RIMORCHIO CON MOTOCOMPRESSORE
RI/MDA	RIMORCHIO PER TRASPORTO MOTO D'ACQUA
RI/MO.ACQ.	RIMORCHIO PER MOTO D'ACQUA
RI/MP	RIMORCHIO PER MOTOPOMPA
RI/MTS	RIMORCHIO PER MOTOSLITTA
RI/NBCR	RIMORCHIO DECON. NUCL.BATTER.CHIM.RAD.
RI/PR	RIMORCHIO TRASPORTO PONTE RADIO
RI/QUAD	RIMORCHIO TRASPORTO QUADRICICLO
RI/S	SCALA AEREA RIMORCHIABILE
RI/SCH	RIMORCHIO PER SCHIUMOGENO
RI/TO.FA.	RIMORCHIO TORRE FARI
RI/VEIC	RIMORCHIO TRASPORTO VEICOLI
RIB/G	RIGID INFLATABLE BOAT GRANDE
RIB/P	RIGID INFLATABLE BOAT PICCOLO
ROUL	ROULOTTE
SBVAD	SERBATOIO BENZINA VERDE ATT. DIDATTICA
SGAD	SERBATOIO GASOLIO ATT. DIDATTICA
SHELTER	CONTAINER SHELTER
SOFF	SOFFIATORI PER RISCALDAMENTO NELLE TENDE

Codice Mezzo	Descrizione Mezzo
SPAZZ	SPAZZATRICE
SR	SR
SR/CIS	SEMIRIMORCHIO CISTERNATO
SR/CON	SEMIRIMORCHIO TRASPORTO CONTAINERS
SR/CP	SEMIRIMORCHIO TRASPORTO CASETTE PIEGH.
SR/CU G	SEMIRIMORCHIO CUCINA DA CAMPO 250 PASTI
SR/CU P	SEMIRIMORCHIO CUCINA DA CAMPO 150 PASTI
SR/ES	SEMIRIMORCHIO TRASPORTO ESCAVATORE
SR/FUR	SEMIRIMORCHIO FURGONATO
SR/MA.O	SEMIRIMORCHIO TRASP. MACC. OPER.
SR/REV	SEMIRIMORCHIO REVISIONE VEICOLI
SR/TC	SEMIRIMORCHIO TRASP. TRATT. CARIC. CING.
SR/TCG	SEMIRIMORCHIO TRASP. TRATT. CARIC. GOM.
TA	TRATTORE APRIPISTA
TCC	TRATTORE CARICATORE CINGOLATO
TCE/G	TERNA CARICATORE ESCAVATORE GOMMATO
TCG	TRATTORE CARICATORE GOMMATO
TER	TERNA
TNKADB	TANICA ADBLUE
TNKDA	TANICA DIESEL AUTOTRAZIONE
TNKDN	TANICA DIESEL NAUTICO
TNKJ	TANICA AVIO
TNKK	TANICA KEROSENE
TNKM	TANICA MISCELA
TNKV	TANICA BENZINA VERDE
TS	TAGLIASIEPI
TT/AGR	TRATTRICE AGRICOLA
ZA	ZATTERA

Bibliografia

- [1] Richard Bellman. *Dynamic Programming*. 1^a ed. Princeton, NJ, USA: Princeton University Press, 1957. URL: <http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false>.
- [2] Jinbo Bi et al. «An improved multi-task learning approach with applications in medical diagnosis». In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2008, pp. 117–132.
- [3] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.
- [4] Nitesh V Chawla et al. «SMOTE: synthetic minority over-sampling technique». In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.
- [5] Richard O Duda, Peter E Hart e David G Stork. *Pattern classification*. Wiley, New York, 1973.
- [6] Roman Eisner et al. «Improving protein function prediction using the hierarchical structure of the gene ontology». In: *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB'05. Proceedings of the 2005 IEEE Symposium on*. IEEE. 2005, pp. 1–10.
- [7] R.A. Fisher. *Dataset Iris*. 1936. URL: <http://archive.ics.uci.edu/ml/datasets/Iris>.
- [8] Robert M Haralick, Karthikeyan Shanmugam et al. «Textural features for image classification». In: *IEEE Transactions on systems, man, and cybernetics* 6 (1973), pp. 610–621.
- [9] Tin Kam Ho. «Random decision forests». In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE. 1995, pp. 278–282.

- [10] Anil K Jain, Robert P. W. Duin e Jianchang Mao. «Statistical pattern recognition: A review». In: *IEEE Transactions on pattern analysis and machine intelligence* 22.1 (2000), pp. 4–37.
- [11] Andrew McCallum. «Multi-label text classification with a mixture model trained by EM». In: *AAAI workshop on Text Learning*. 1999, pp. 1–7.
- [12] Timo Ojala, Matti Pietikäinen e David Harwood. «A comparative study of texture measures with classification based on featured distributions». In: *Pattern recognition* 29.1 (1996), pp. 51–59.
- [13] Timo Ojala, Matti Pietikainen e Topi Maenpaa. «Multiresolution gray-scale and rotation invariant texture classification with local binary patterns». In: *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 971–987.
- [14] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [15] Fernando Pérez e Brian E. Granger. «IPython: a System for Interactive Scientific Computing». In: *Computing in Science and Engineering* 9.3 (mag. 2007), pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: <http://ipython.org>.
- [16] J. Ross Quinlan. «Induction of decision trees». In: *Machine learning* 1.1 (1986), pp. 81–106.
- [17] Md Abdur Rahim et al. «Face recognition using local binary patterns (LBP)». In: *Global Journal of Computer Science and Technology* (2013).
- [18] Guido Rossum. *Python Reference Manual*. Rapp. tecn. Amsterdam, The Netherlands, The Netherlands, 1995.
- [19] A. L. Samuel. «Some Studies in Machine Learning Using the Game of Checkers». In: *IBM J. Res. Dev.* 3.3 (lug. 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: <http://dx.doi.org/10.1147/rd.33.0210>.
- [20] Yutaka Sasaki, Brian Rea e Sophia Ananiadou. «Multi-topic aspects in clinical text classification». In: *Bioinformatics and Biomedicine, 2007. BIBM 2007. IEEE International Conference on*. IEEE. 2007, pp. 62–70.
- [21] Robert E Schapire e Yoram Singer. «BoosTexter: A boosting-based system for text categorization». In: *Machine learning* 39.2-3 (2000), pp. 135–168.

- [22] Minseok Seo e Sejong Oh. «CBFS: High performance feature selection algorithm based on feature clearness». In: *PloS one* 7.7 (2012), e40419.
- [23] Marina Sokolova e Guy Lapalme. «A systematic analysis of performance measures for classification tasks». In: *Information Processing & Management* 45.4 (2009), pp. 427–437.
- [24] Stéfan van der Walt, S. Chris Colbert e Gaël Varoquaux. *The NumPy Array: A Structure for Efficient Numerical Computation*. 2011. DOI: 10.1109/MCSE.2011.37. URL: <http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37>.

Indice analitico

- Accuratezza, 26
- Agente, 21
- Alberi decisionali, *vedi* Decision tree
- Algoritmo greedy, 45
- Angular second moment, *vedi* ASM
- Apprendimento ensemble, 47
- ASM, 41
- Autovalore, 31
- Autovettore, 31
- Bagging, 47
- Big Data, 50
- Bootstrap, 48
- Branch, 43
- Branching factor, 43
- C4.5, 44
- CART, 44
- CBFS, 69
- Clustering, 24
- Componenti principali, 29
- Computer vision, 32
- Contrasto, 40
- Correlazione, 42
- Criterio di stop, 47
- Cross-validation, 69
- Curse of dimensionality, 31
- Dataset, 18
- Dati categorici, 20
- Nominali, 20
- Ordinali, 20
- Dati numerici, 20
- Continui, 20
- Discreti, 20
- Decision tree, 43
- Dissimilarity, 41
- Energia, 41
- Entropia, 45
- Errore di classificazione, 45
- F-score, 26
- Feature, 19
- Grado, 77
- Gray level co-occurrence matrix, 38
- Homogeneity, 41
- ID3, 44
- Impurità di Gini, 45
- Information gain ratio, 45
- Intelligenza artificiale, 18
- Inverse difference moment, *vedi* Homogeneity
- Iperparametri, 62
- Istanze, 19
- K-fold, 70
- Leaf node, 43
- Leave-One-Out, 69

Local binary pattern, 32
Macchine a stati-finiti, 77
Machine Learning, 18
MATLAB, 52
Matrice di co occorrenza dei livelli
di grigio, *vedi* Gray level
co-occurrence matrix
Matrice di confusione, 25
Matrice di covarianza, 30
Matrice di occorrenza, 38
Numpy, 62
Overfitting, 31
Pattern, 32
Precisione, 26
Principal component analysis, 28
Pruning, 47
Python, 62
Random Forest, 47
Recall, 26
Ricerca a griglia, 62
ROC, 74
Root node, 43
Scikit-learn, 62
Sistema di apprendimento debole,
47
Sistema di apprendimento forte, 47
Teorema dei moltiplicatori di
Lagrange, 30
Texture, 32
Trasformata di Karhunen-Loéve,
vedi Principal component
analysis
Unbalanced dataset, 70
Varianza, 29
Voto di maggioranza, 48