Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Triennale in Ingegneria Informatica

Metodi di Machine Learning per l'Analisi Predittiva di Prestazioni di Squadre in Ambito Sportivo

Candidato
Mirco Ceccarelli

Relatore
Prof. Marco Sciandrone

Correlatore
Dott. Matteo Lapucci

Anno Accademico 2018/2019

Indice

Introduzione

Il presente studio costituisce un lavoro di tesi per il corso di laurea triennale in ingegneria Informatica presso l'Università degli studi di Firenze.

La tesi è stata svolta presso il GOL (Global Optimization Laboratory) sotto la supervisione del professore Marco Sciandrone e del dottor Matteo Lapucci, in collaborazione con la Lega Pallavolo Serie A.

L'obiettivo del presente studio è di utilizzare dati relativi alle prestazioni sportive passate dei singoli giocatori della serie A di Pallavolo, per predire a inizio campionato quali saranno le squadre che accederanno alla fase finale del campionato, i Playoff, successiva alla fase di Regular Season. Per questo lavoro sono stati presi in considerazione i dati relativi alle stagioni di Serie A di pallavolo maschile dalla stagione 2001/02 a quella 2017/18.

Nello specifico, sono stati considerati i dati riguardanti le prestazioni di ogni singolo atleta stagione per stagione. Ogni squadra è quindi rappresentata dall'insieme dei giocatori che compongono la rosa ad inizio stagione.

Lo scopo del lavoro è stato quello di identificare modelli di apprendimento supervisionato in grado di predire eventi futuri utilizzando informazioni su eventi passati.

Le tecniche di Machine Learning hanno acquisito grande interesse nel mondo dello sport: permettono infatti di migliorare le prestazioni dei giocatori e delle squadre, riducendo il rischio di infortuni e migliorando ogni

atleta sugli aspetti in cui è meno preparato. Tutto questo è possibile grazie all'utilizzo di un enorme quantità di dati generati dai dispositivi tecnologici impiegati nel monitoraggio dell'attività di ogni atleta. [1]

Sono stati utilizzati diversi modelli come la Regressione Logistica e il Support Vector Machine (SVM). In particolare questi ultimi hanno prodotto buoni risultati di test, nonostante possiamo notare l'incompletezza del dataset utilizzato. C'è infatti totale mancanza di informazioni relative ai giocatori provenienti da campionati diversi da quello della Lega Serie A e quindi non rappresentabili dai dati forniti. Dunque il dataset è potenzialmente migliorabile.

Nel nostro studio risulta essere rilevante sfruttare i dati disponibili per un'analisi predittiva su ogni squadra che compone il dataset, la quale è composta da insieme di giocatori che ne identifica le caratteristiche.

L'elaborato presenta la seguente struttura:

- **Capitolo 1:** si introducono i concetti di Machine Learning e Sport Prediction, insieme agli algoritmi di apprendimento supervisionato successivamente implementati.

- **Capitolo 2:** si descrive l'insieme di operazioni applicate sui dati, volte a definire il dataset finale pronto ad essere impiegato in fase di addestramento.

- **Capitolo 3:** presenta un'analisi dei risultati in termini delle varie metriche relative ai vari modelli predittivi implementati. Tali risultati sono poi messi a confronto tra di essi.

- **Capitolo 4:** contiene le conclusioni relative allo studio e individua un sottoinsieme di possibili futuri lavori.

Capitolo 1

Machine Learning e Sport Prediction

1.1 Machine Learning

Il Machine Learning (in italiano "Apprendimento Automatico") è un ramo dell'Intelligenza Artificiale (AI) che mira a fornire ad un sistema abilità che sono autonomamente apprese dai dati senza che debbano essere esplicitamente programmate. [2]

Idealmente, l'obiettivo principe dell'apprendimento automatico è che una macchina sia in grado di generalizzare a partire dalla propria esperienza, ossia che sia in grado di svolgere ragionamenti induttivi. In questo contesto, per generalizzazione si intende l'abilità di un sistema di risolvere nuove istanze di un problema dopo aver imparato da un certo numero di istanze di esempio, ovvero dopo aver fatto esperienza su un insieme di dati di apprendimento. [3]

C'è dunque una sostanziale differenza tra lo schema classico della concezione di programmazione, dove il programmatore imposta il calcolatore in modo che esegua una funzione f che per ogni input in ingresso fornisca l'out-

put desiderato, e l'approccio del machine learning dove il programmatore imposta il calcolatore in modo che sia esso stesso a determinare quale funzione f (presa da un insieme di funzioni possibili $\mathcal{H}$ detto *Hypothesis Space*, spazio delle ipotesi) usare sui dati.

Negli anni recenti le tecniche di Machine Learning si sono affermate come soluzione di riferimento per molti problemi applicativi, ad esempio:

- Analisi di immagini e video
- Riconoscimento di testo e vocale
- Sistemi di classificazione
- Sistemi di raccomandazione

Di riflesso, sempre più aziende hanno iniziato ad utilizzare tecniche di Machine Learning per migliorare i loro obiettivi aziendali. Alcuni esempi possono essere alcune delle piattaforme di streaming più famose: *Netflix* (raccomandazione dei film), *Spotify* (raccomandazione musica) o *Youtube* (raccomandazione dei video), ma anche *Facebook* (social network) o *Amazon* (piattaforma commerciale). [4]

La crescente efficacia di questa tipologia di algoritmi è in larga parte dovuta alla quantità di dati in continuo aumento e alla contemporanea disponibilità di computer con potenza di calcolo sempre più significativa. Grazie a questo secondo fattore in particolare, si può ottenere una maggiore rapidità nell'addestramento dei modelli di machine learning in presenza di dataset di dimensioni elevate. [5]

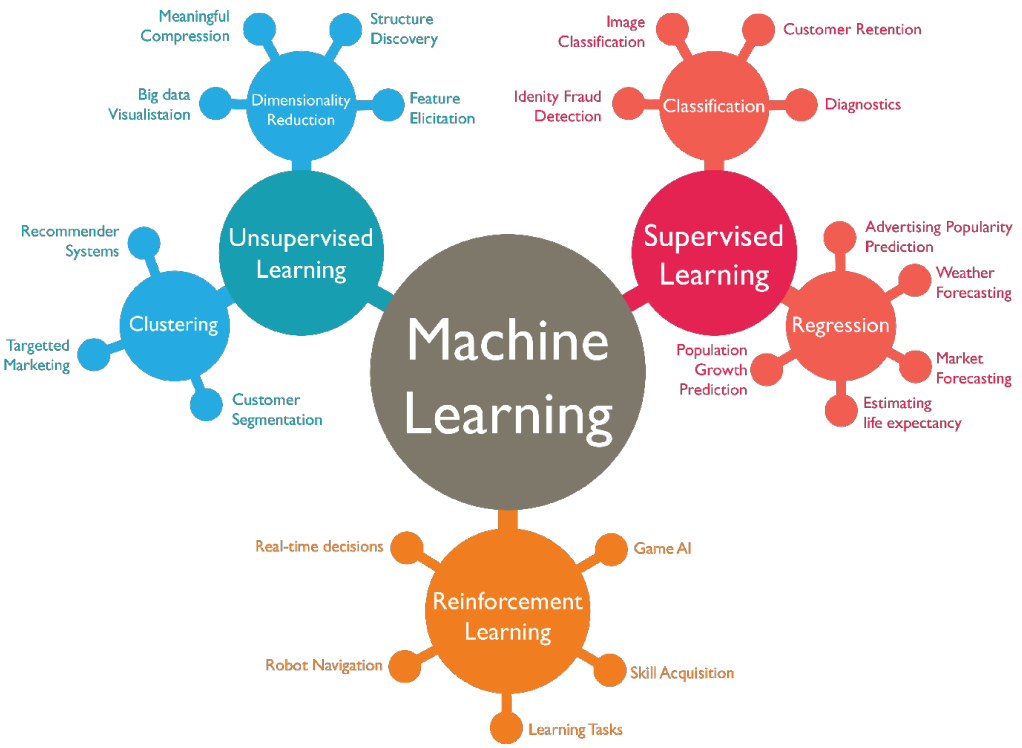

Figura 1.1: Schema riassuntivo delle tecniche di Machine Learning [6]

1.2 Modelli predittivi nello sport

Come detto, gli algoritmi di machine learning possono essere utilizzati in vari campi. Sicuramente un ambito che può trarre beneficio dall'utilizzo dei modelli predittivi di machine learning è quello relativo allo sport. In particolare, modelli predittivi efficaci possono risultare utili per: [7]

- Predire l'esito di un incontro tra due squadre
- Richiedere la definizione di tipologie particolari di allenamento per ogni atleta al fine di incrementare le prestazioni individuali e quindi migliorare i risultati della squadra di appartenenza

- Trovare le migliori strategie per affrontare un avversario e vincere una partita.

Con l'evoluzione e la diffusione della tecnologia nello sport si possono avere quantità di dati sempre maggiore tramite l'utilizzo ad esempio di *wearable trackers* o *motion cameras*. Questi dati possono essere sfruttati per migliorare le prestazioni sportive degli atleti e della società sportiva stessa, dunque è importante non solo raccoglierli, ma anche filtrarli e processarli. [8]

Tali informazioni non vengono solamente utilizzate dalle società sportive per formulare strategie che portino alla vittoria, ma sono ampiamente utilizzate anche in ambito delle scommesse sportive dove i vari bookmakers investono grosse cifre nello studio di modelli di predizione più accurati possibili. [7]

Questi modelli dipendono da molti fattori [9]:

- I risultati dei precedenti incontri
- Gli indicatori delle prestazioni individuali dei giocatori
- Le informazioni relative alle squadre avversarie

Il lavoro di tesi si sviluppa in questo contesto.

1.3 Stato dell'arte

Adottare le tecniche di intelligenza artificiale nell'ambito sportivo non è più una novità, è diventato anzi fondamentale per poter raggiungere i massimi risultati sia individuali che di squadra.

Nel nostro studio analizzeremo la Pallavolo, uno sport caratterizzato da un insieme di azioni che si ripetono in modo sistematico. I dati statistici

relativi a questo sport sono quindi particolarmente adatti per essere utilizzati in analisi predittive e statistiche.

Nella letteratura specialistica di questo ambito è possibile trovare vari articoli in cui vengono analizzati i vari fattori e le varie azioni che caratterizzano la pallavolo.

Ad esempio:

- In [6] si analizzano un insieme di parametri legati alle possibili azioni durante la partita, con l'obiettivo di individuare quelli più significativi per il successo di una squadra in termini della classifica finale.

- In [10] si ricercano indicatori per la performance in grado di predire correttamente la posizione della squadra in una competizione di alto livello nella categoria maschile e femminile della pallavolo. Il risultato di questa analisi porta ad affermare che il numero di *muri-punto* e il numero di *ace per set* influenzano in modo considerevole la classifica finale.

- In [11] è oggetto di studio la partecipazione del libero e la sua influenza nelle fasi di attacco e difesa. Viene mostrato che la fase difensiva è la chiave per la vittoria della gara e che le squadre che hanno una difesa migliore, possono attaccare con maggiore efficienza.

- In [12] si definiscono due valori che sono in grado di definire con più accuratezza le performance di una squadra: SER (*Server Efficency Ratio*) e AER (*Attack Efficency Ratio*).

- In [13] si studia l'evoluzione del gioco, in cui si sottolinea la propensione della maggioranza delle squadre a migliorare la fase di ricezione, che porta vantaggi anche nella fase successiva di attacco. In questa di-

rezione diventa importante l'azione della battuta, vista come elemento di complessità per la fase di ricezione successiva.

- In [14] vengono valutate le abilità dei singoli giocatori per capire quale di queste portano al successo; tra i fattori considerati, i risultati mostrano che gli *ace*, gli *errori in ricezione* e gli *attacchi murati* sono fattori importanti in grado di individuare il vincitore della partita.

Gli articoli citati aiutano dunque a dare un'idea di quali siano i parametri più importanti e decisivi per stabilire la squadra che risulterà vincitrice.

1.4 Classificazione delle tecniche di Machine Learning

Le tecniche di Machine Learning possono essere classificate in tre macro categorie a seconda della natura del "segnale" utilizzato per l'apprendimento o del "feedback" disponibile al sistema di apprendimento. [15] Queste categorie sono:

- *Supervised Learning (Apprendimento Supervisionato)*
- *Unsupervised learning (Apprendimento Non Supervisionato)*
- *Reinforcement Learning (Apprendimento per Rinforzo)*

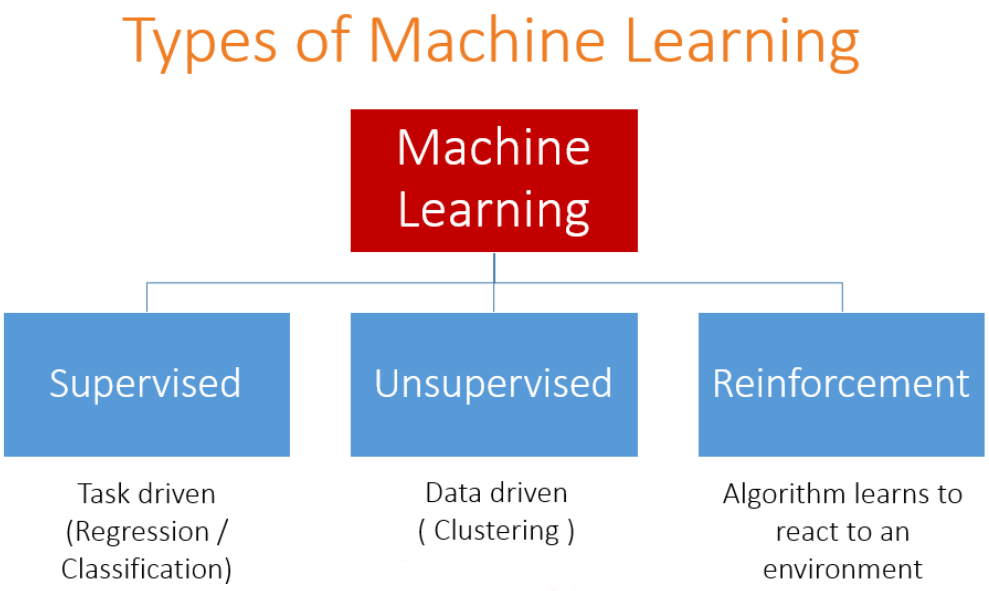

Figura 1.2: Macro categorie del Machine Learning

Di seguito si elencano le principali caratteristiche di queste tre macro categorie, soffermandoci con maggiore attenzione sulla categoria di interesse per questo studio.

1.4.1 Supervised Learning

Nella macrocategoria delle tecniche di Supervised Learning, c'è una distinzione forte tra input (appartenenti ad un insieme $\mathcal{X}$ detto *Spazio degli Input*) e output (appartenenti ad un insieme $\mathcal{Y}$ detto *Spazio degli Output*); l'addestramento di un modello di apprendimento supervisionato consiste nel definire una funzione $\bar{f}$ che mappi punti in X verso punti in Y, approssimando il vero fenomeno f che ha prodotto i dati. L'addestramento è svolto utilizzando dati etichettati, ossia coppie $(x, y) \in X \times Y$ tali che $f(x) = y$. La funzione ottenuta $\bar{f}$ può quindi essere utilizzata su nuovi dati in input mai visti prima.

Più precisamente, il sistema è in possesso di una serie di dati caratterizzati da un insieme di features (detto *Dataset* e indicato con la lettera $\mathcal{D}$), sulla base dei quali deve definire dei pattern rappresentativi ricercando un modello matematico in grado di indicare la classe di appartenenza di nuovi dati; questi a loro volta devono rispecchiare l'insieme di features dei dati di training.

La funzione $\bar{f}$ sopracitata viene dunque presa da un insieme di funzioni possibili detto $\mathcal{H}$ (*Hypothesis Space*), la scelta di questo $\mathcal{H}$ è uno dei problemi chiave dell'apprendimento. Si possono inoltre presentare due problematiche:

- **Underfitting**: i dati sono insufficienti per identificare una funzione che sia in grado di generalizzare da $\mathcal{D}$ su tutto $\mathcal{X}$. Il modello si comporta male sia nell'addestramento sia nella fase di test.

- **Overfitting**: la funzione è estremamente precisa sui dati del train set, ma al di fuori di quelli commette errori significativi. Il modello si comporta estremamente bene sul training set, ma scarsamente sul test set.

Esempi di campi applicativi di questo apprendimento comprendono task come *speech recognition*, *spam filtering*, *information retrieval* oppure *image recognition*. [2]

L'insieme di problemi risolvibili utilizzando tecniche di Supervised Learning sono compiti di:

- *Classificazione*, dove gli output sono costituiti da due o più classi e il sistema di apprendimento deve produrre un modello che assegni gli input non ancora visti a una o più di queste. In altre parole, il modello è addestrato a dare risposte con valore categorico (es.: in un task di classificazione binaria la riposta può essere "sì" o "no")

- *Regressione*, dove l'output del modello è continuo. In altre parole, la risposta è un valore numerico reale.

1.4.2 Unsupervised Learning

Algoritmi di apprendimento non supervisionato vengono utilizzati quando non si hanno informazioni relative alla classe di appartenenza dei dati a disposizione. Non avendo alcuna informazione sui dati in input, il modello deve trovare una qualche sorta di schemi sui dati.

Al contrario dell'apprendimento supervisionato, durante l'apprendimento vengono forniti all'algoritmo solo esempi non annotati, in quanto le classi non sono note a priori ma devono essere apprese automaticamente.

Esempi di campi applicativi di questi modelli di apprendimento comprendono task come *system diagnosis* o *social network analysis*.

L'insieme di problemi risolvibili utilizzando tecniche di Unsupervised Learning comprende task di:

- *Clustering*, dove un insieme di punti in input deve essere diviso in gruppi. Diversamente da quanto accade per la classificazione, i gruppi non sono noti a priori.

- *Anomaly Detection*

Algoritmi di questa classe sono spesso utilizzati come fase di pre-processing alla quale segue un apprendimento supervisionato. [2]

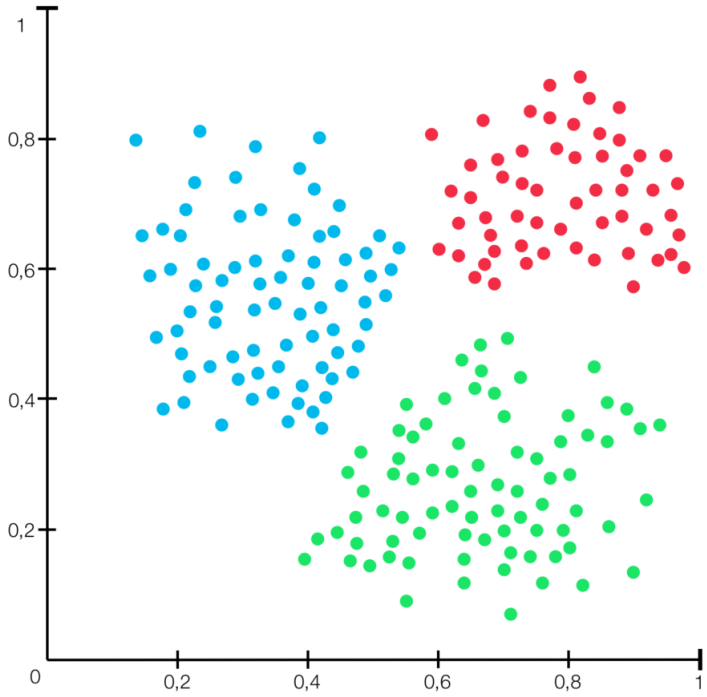

Figura 1.3: Definizione cluster per un insieme di punti con algoritmo di apprendimento non supervisionato

1.4.3 Reinforcement Learning

Questi tipi di algoritmi puntano a definire sistemi in grado di apprendere ed adattarsi alle mutazioni dell'ambiente in cui sono immersi, attraverso la distribuzione di una "ricompensa", detta rinforzo, che consente la valutazione delle loro prestazioni. [2]

La supervisione dunque è fornita in un secondo momento (viene data informazione sulle conseguenze delle decisioni già prese). Si noti che qui, a differenza degli altri due casi precedenti, è forte la caratterizzazione temporale dell'input.

In questa tesi abbiamo affrontato il problema in esame con approcci di tipo supervisionato. I dati in possesso riguardano i giocatori che hanno preso parte alle ultime stagioni della Serie A di pallavolo maschile. Questi dati, se usati in modo opportuno, possono essere sfruttati per definire un modello predittivo in grado di indicare, con una percentuale alta di accuratezza, quali sono le squadre che accederanno alla fase di Playoff del Campionato prima che questa inizi a svolgersi.

1.5 Modelli di apprendimento supervisionato

Di seguito verrà elencato un insieme di metodi che permettono di definire un modello predittivo tramite un processo di supervisione e vedremo come questi di differenziano tra loro.

Nella fase di addestramento di un modello è necessario suddividere il dataset $\mathcal{D}$ di esempi etichettati in due parti:

- **Training Set**: esempi utilizzati per addestrare il sistema.
- **Test Set**: dati utilizzati esclusivamente per valutare la bontà del modello, simulando il contesto applicativo.

Uno schema classico prevede di suddividere il dataset in 80% per addestrare e 20% per testare; questa suddivisione però può essere variata secondo le esigenze del problema come anche noi abbiamo fatto nello studio di questa tesi.

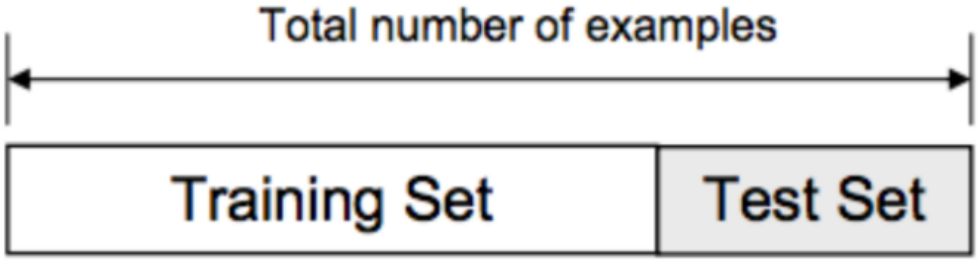

Figura 1.4: Suddivisione dataset per l'apprendimento supervisionato

I problemi di classificazione si possono suddividere in due categorie:

- *Classificazione Binaria*: quando il target può assumere due valori (sì,no)
- *Classificazione Multiclasse*: quando il target può assumere un numero finito di valori interi numerici.

Nel nostro studio sarà una classificazione di tipo *binaria*, in quanto la previsione sarà fatta sulla partecipazione o meno ai Playoff di una squadra al termine della regular season.

In questo lavoro, sono stati testati i seguenti modelli:

- **Regressione Logistica**
- **Support Vector Machine (SVM)**

Nel resto di questa sezione andremo a descrivere i principi matematici che garantiscono il funzionamento di questi modelli.

1.5.1 Regressione Logistica

La Regressione Logistica è un modello di apprendimento supervisionato che può essere utilizzato per risolvere task di classificazione. Prende in input un insieme di esempi caratterizzati da una serie di features (attributi) e classificati utilizzando un'etichetta; essi costituiscono il training set dell'algoritmo e portano alla definizione di un modello matematico il quale avrà il compito di assegnare l'etichetta opportuna all'arrivo di nuovi dati. Il modello prescelto dovrà essere testato su un insieme di dati classificati a priori ma di cui si ignora la label, che costituiranno il test set.

La regressione logistica permette di generare un risultato che, di fatto, rappresenta una probabilità che un dato valore di ingresso appartenga a una determinata classe. Nei problemi di regressione logistica binomiale, la probabilità che l'output appartenga ad una classe sarà P, mentre che appartenga all'altra classe 1-P (dove P è un numero compreso tra 0 e 1 perché esprime una probabilità). La regressione logistica binomiale lavora bene in tutti quei casi in cui la variabile che stiamo cercando di predire è binaria, cioè può assumere solamente due valori: il valore 1 che rappresenta la classe positiva, o il valore 0 che rappresenta la classe negativa.

Come tutte le analisi di regressione, la regressione logistica è un'analisi predittiva che viene utilizzata per misurare la relazione tra la variabile dipendente (ossia ciò che vogliamo prevedere) e l'una o più variabili indipendenti (le nostre caratteristiche), stimando delle probabilità tramite una funzione logistica. Queste probabilità vengono successivamente trasformate in valori binari (che assumono valori compresi tra 0 e 1) per poter effettivamente fare una previsione. Successivamente si assegna il risultato della previsione alla classe di appartenenza, in base alla vicinanza o meno alla classe stessa. Se ad esempio otteniamo un valore 0,8 dopo aver applicato la funzione logisti-

ca, diremo che l'input ha generato una classe positiva e verrà assegnato alla classe 1 (viceversa se avesse ottenuto valore $< 0{,}5$).

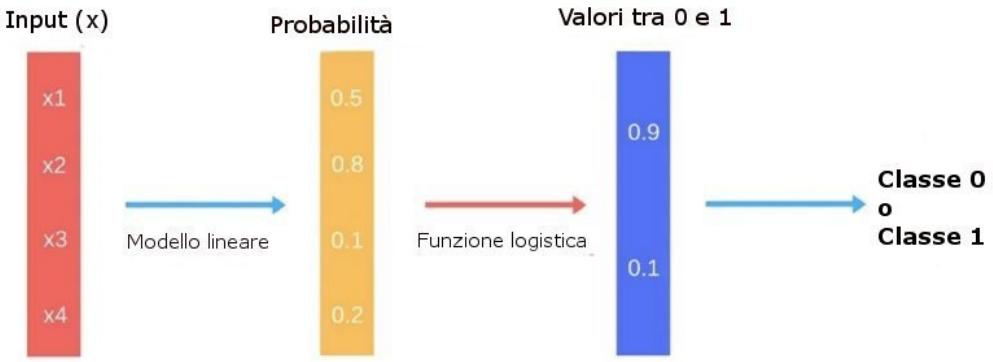

Figura 1.5: Funzionamento della Regressione Logistica

Dunque la regressione logistica è una trasformazione della regressione lineare, utilizzata per problemi di classificazione.

La regressione lineare è definita dalla funzione $z : \mathbb{R}^d \to \mathbb{R}$

$$z = < w,x > + b$$

Dobbiamo quindi applicare una trasformazione a z in modo che il suo output sia 0 o 1.

Tale trasformazione, che porta alla regressione logistica, avviene grazie alla *funzione logistica* $\phi(z)$. [16]

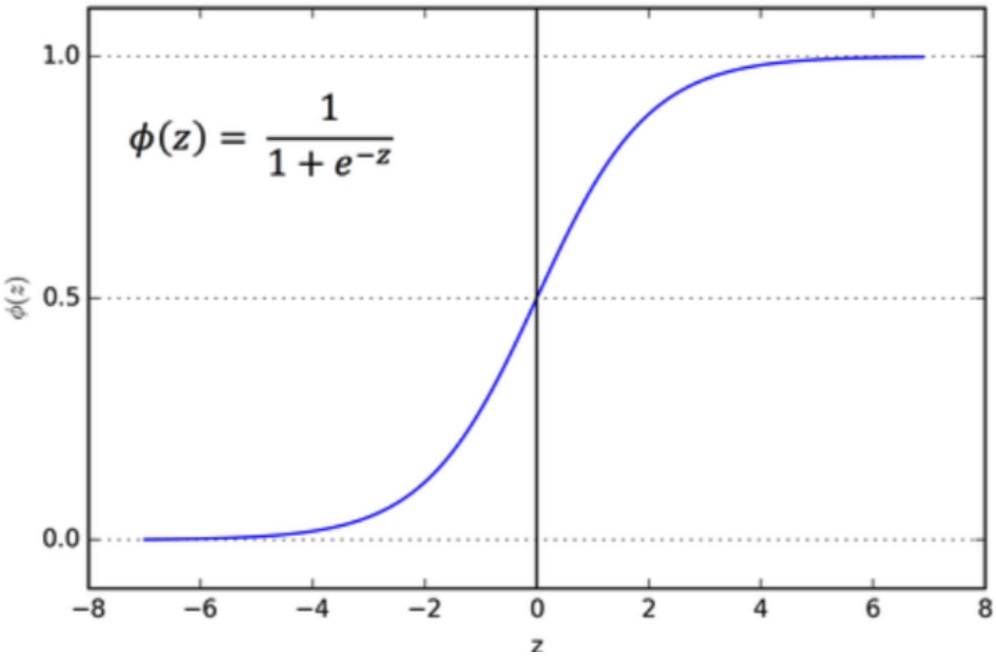

Figura 1.6: $\phi(z) : \mathbb{R} \rightarrow [0,1]$ trasforma un numero reale $\mathbb{R}$ in un numero fra 0 e 1.

Il **processo di trasformazione** è il seguente:

- Calcolo $z = < w,x > + b \rightarrow$ ottengo un valore in $\mathbb{R}$
- Calcolo $\phi(z) = \frac{1}{1+e^{-z}} \rightarrow$ ottengo un numero reale in [0,1]

$$\Downarrow$$

Ottengo la probabilità di appartenere alla classe 1 e calcolo:

$$classe(x) = \begin{cases} 0, & \text{se } \phi(z) < 0.5 \\ 1, & \text{altrimenti} \end{cases} \tag{1.1}$$

$$\Downarrow$$

Ottengo l'etichetta della classe

Dunque la **Regressione Logistica** può essere rappresentata dalla seguente equazione:

$$\bullet\, y = \frac{\mathrm{e}^{-z}}{1+\mathrm{e}^{-z}} \quad con \quad z =< w, x > +b \tag{1.2}$$

Dove:

- x valori di input
- w coefficienti del modello lineare
- y valore da predire

In generale i valori di input x vengono combinati linearmente usando i pesi o coefficienti w per prevedere un valore di output y.

I coefficienti w dell'algoritmo di regressione logistica vengono stimati tramite il metodo di massima verosimiglianza $\ell(w)$ e da questo il problema di ottimizzazione derivante è il seguente:

$$\hat{w} = \arg\max_{w} \ell(w) = \arg\min_{w} \sum_{i=1}^{n} \log\left(1 + e^{-y_i w \cdot x_i}\right) + \lambda \|w\|^2, \tag{1.3}$$

dove $\lambda\|w\|^2$ è un termine di regolarizzazione.

Questo metodo prevede di ottenere quella serie di coefficienti di regressione per i quali la probabilità di ottenere i dati che abbiamo osservato è massima. Essa, infatti, permette di cercare valori per i coefficienti che minimizzano l'errore nelle probabilità previste dal modello rispetto a quelle nei dati.

1.5.2 Support Vector Machine (SVM)

Anche in questo caso ci troviamo di fronte a un algoritmo di apprendimento supervisionato che può essere utilizzato per risolvere task di classificazione.

Il training set dell'algoritmo è caratterizzato da una serie di features (attributi) classificate utilizzando un'etichetta. L'obiettivo è la definizione di un modello matematico che dovrà assegnare l'etichetta opportuna all'arrivo di nuovi dati.

Quella delle SVM è una famiglia di classificatori importante per vari aspetti. Innanzitutto non utilizzano valori di probabilità: si possono derivare direttamente attraverso una formulazione geometrica e l'ottimizzazione. Consentono poi di definire classificatori (o regressori) non lineari, attraverso l'uso dei kernel.

- **Modello Lineare**

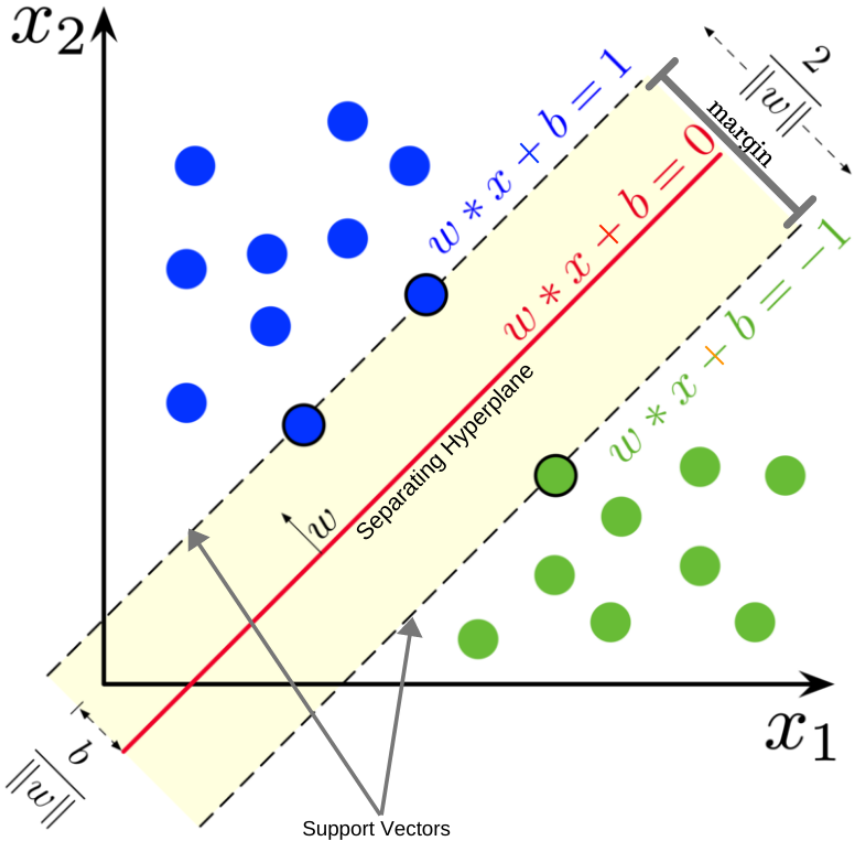

Figura 1.7: Iperpiano ottimo per la separazione dei punti in SVM (Modello Lineare)

Assumiamo che $\mathcal{D} = \{ (x^{(i)}, y^{(i)}), i = 1,...,n\}$ sia il nostro dataset e, dato che nel nostro studio di tesi siamo nel caso binario, $y^{(i)} \in \{-1,1\}$. L'idea alla base di questo classificatore è la ricerca dell'iperpiano ottimo di separazione $(w \cdot x + b = 0)$, cioè quello che massimizza la distanza $(\frac{w \cdot x + b}{\|w\|})$ tra i due insiemi di punti $(w \cdot x + b \geq 1 \ \ per \ \ y = +1 \ \ \& \ \ w \cdot x + b \leq 1 \ \ per \ \ y = -1$, queste due disequazioni quando vale l'uguaglianza identificano i *confini del margine*).

Se il vettore dei pesi è indicato da w e $\|w\|$ è la norma di questo vettore (ossia la sua lunghezza) allora è facile vedere che la dimensione del *margine massimo* è: $\frac{2}{\|w\|}$. [17]

Il problema di ottimizzazione di questo modello è il seguente:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ & y_i(w \cdot x_i + b) \geq 1 \quad \forall i \end{aligned} \tag{1.4}$$

- **Modello Non Lineare**

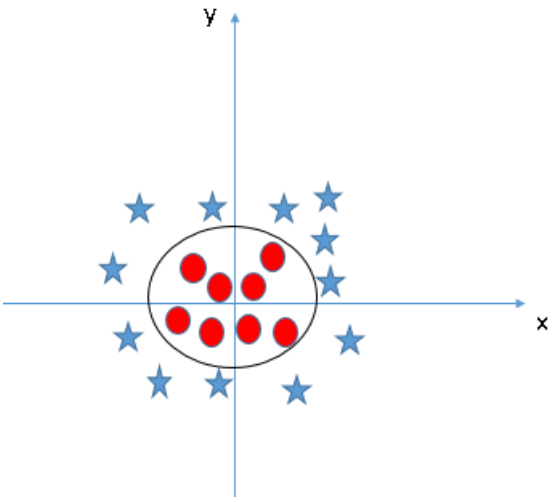

Figura 1.8: Classificazione dati non lineari in SVM (Modello Non Lineare)

Gli algoritmi SVM possono essere lineari, quando i dati sono linearmente separabili, oppure non lineari, quando serve una funzione non lineare per separarli; per questo si utilizzano kernel che presentano funzioni di attivazione

non lineari. Per fare ciò, dal modello lineare di SVM, si devono rilassare i requisiti di separabilità, ottenendo una formulazione del problema di ottimizzazione come segue:

$$\begin{aligned} \min_{w,b,\xi} \quad & \|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ & y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \end{aligned} \tag{1.5}$$

dove ξ_i sono le *variabili di Slack* e $C > 0$ è un *iperparametro* che indica una penalità.

Il problema precedente ha la seguente formulazione duale:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \phi(x_i) \cdot \phi(x_j)\lambda_i\lambda_j - \sum_{i=1}^{n}\lambda_i \\ & \sum_{i=1}^{n}\lambda_i y_i = 0 \\ & 0 \leq \lambda_i \leq C \end{aligned} \tag{1.6}$$

Questa formulazione permette di utilizzare le funzioni kernel, le quali consentono di sostituire il prodotto scalare tra due esempi con un'altra funzione che rappresenta il prodotto scalare in uno spazio a dimensione maggiore in cui gli esempi siano linearmente separabili.

In generale una funzione $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ è un kernel se soddisfa la seguente proprietà: $k(x, y) = < \phi(x), \phi(y) > \quad \forall x, y \in \mathbb{R}^n$ per una funzione $\phi : \mathbb{R}^n \to \mathcal{H}$, dove $\mathcal{H}$ è uno *spazio euclideo*, ovvero uno spazio lineare in cui è definito un prodotto scalare. Il punto chiave dei metodi kernel è che non serve conoscere la trasformazione ϕ, ma basta conoscere la funzione k. [18]

Esistono varie funzioni kernel, tra queste per il nostro studio abbiamo considerato le seguenti:

- **Kernel Lineare** : $k(x_i, y_i) = x_i \cdot y_i$
- **Kernel Gaussiano** di raggio σ $(\sigma > 0)$: $k(x_i, y_i) = \mathrm{e}^{\frac{-\|x_i - y_i\|^2}{2\sigma^2}}$

Capitolo 2

Descrizione del Dataset e Analisi dei Dati

2.1 Obiettivo e Mezzi

Il presente studio si propone di definire un modello predittivo riguardante la valutazione delle squadre della Lega Serie A Pallavolo Maschile, basandosi sulle statistiche delle prestazioni dei giocatori che le compongono relative alle stagioni precedenti. Il modello non sfrutta dati legati alla squadra intesa come organizzazione, ma si limita a valutare gli atleti, quindi le squadre vengono definite in base all'insieme di giocatori che fanno parte della rosa a inizio campionato. Questo modello dovrebbe idealmente essere in grado di predire a inizio stagione quali sono le squadre che accederanno alla fase finale del campionato, i Playoff, cercando di massimizzare l'accuratezza della predizione.

Per implementare i metodi di pre processamento è stato utilizzato *Pandas*, una libreria software scritta in linguaggio di programmazione Python per la manipolazione e l'analisi dei dati. [19]

2.2 Descrizione Dataset

Per questo studio di tesi la Lega Pallavolo Serie A ha fornito due versioni del dataset a sua disposizione: **Dataset Large** e **Dataset Small**

• Il **Dataset Large** è composto in maniera tale che ogni riga rappresenti una squadra partecipante a una particolare stagione con l'insieme dei giocatori, cioè:

- **centrali** (indicati con la lettera **C**): 1 - 6
- **liberi** (indicati con la lettera **L**): 1 - 3
- **palleggiatori** (indicati con la lettera **P**): 1 - 5
- **schiacciatori** (indicati con la lettera **S**): 1 - 11

Stagione	Squadra	C1_bat_pos	C1_bat_neg	C1_ric_pos	C1_ric_neg	C1_att_pos	C1_att_neg	C1_mur_pos	C1_mur_neg	C2_bat_pos	C2_bat_neg	C2_ric_pos	C2_ric_neg	C2_att_pos	C2_att_neg	C2_mur_pos	C2_mur_neg
2001	Asystel Milar	0,30547961	0,63391433	0,56363636	0,37575758	0,89831257	0,04108137	0,83237438	0,10701956	0,30157068	0,49842932	0,76	0,04	0,75663957	0,04336043	0,68125	0,11875
2001	Borgocanale	0,25617829	0,52563989	0,70615836	0,07565982	0,71858289	0,06323529	0,67520661	0,10661157	0,22314593	0,32836922	0,27575758	0,27575758	0,53030303	0,02121212	0,44890768	0,10260747
2001	Bossini Sangi	0,25760426	0,2939109	0,47272727	0,07878788	0,50325758	0,04825758	0,45642633	0,09508882	0,15954378	0,36772895	0,43939394	0,08787879	0,48095823	0,0463145	0,42683983	0,1004329
2001	Casa Modeni	0,42949826	0,41292598	0,72754821	0,11487603	0,78988596	0,05253829	0,77817155	0,0642527	0,18103109	0,51593861	0,6040404	0,09292929	0,67141414	0,02555556	0,58974359	0,10722611
2001	Icom Latina	0,1904	0,42778182	0,44958678	0,16859504	0,57744891	0,04073291	0,58138528	0,03679654	0	0	0	0	0	0	0	0
2001	Itas Diatec Tr	0,20633609	0,44214876	0,45934343	0,18914141	0,61003713	0,03844772	0,61399097	0,03449387	0,15551583	0,4808478	0,35353535	0,28282828	0,59659091	0,03977273	0,55099778	0,08536585
2001	Lube Banca M	0,32521948	0,39599264	0,49583333	0,22537879	0,67254136	0,04867076	0,66273546	0,05847666	0,20860578	0,48836392	0,54627355	0,15069615	0,66311688	0,03385281	0,62106211	0,07590759
2001	Maxicono Pa	0,25803595	0,45105496	0,60779221	0,1012987	0,67127273	0,03781818	0,67177033	0,03732057	0,16208598	0,39548978	0,37171717	0,18585859	0,52392894	0,03364681	0,50570825	0,05186751
2001	Noicom Bret	0,35906895	0,52577953	0,7679817	0,11686678	0,82350395	0,06134454	0,66060606	0,22424242	0,34146763	0,46459298	0,55416667	0,25189394	0,75852878	0,04753183	0,69513483	0,11092577
2001	Roma Volley	0,05362096	0,12213662	0,04393939	0,13181818	0,16034024	0,01541733	0,16477273	0,01098485	0,0171123	0,05561497	0	0	0,06993007	0,0027972	0,03636364	0,03636364
2001	Sempre Volle	0,27426376	0,4893726	0,62479339	0,13884298	0,72668622	0,03695015	0,61740812	0,14622824	0,07461853	0,26477541	0,31111111	0,02828283	0,31361718	0,02577675	0,24393939	0,09545455
2001	Sira Cucine A	0,19585492	0,40414508	0,45882353	0,14117647	0,57324841	0,02675159	0,53478261	0,06521739	0	0	0	0	0	0	0	0
2001	Sisley Trevisc	0,30123737	0,61391414	0,72156177	0,19358974	0,86189916	0,05325235	0,82858313	0,08656839	0,27199123	0,3583118	0,51774892	0,11255411	0,58336557	0,04693746	0,55913978	0,07116325
2001	Yahoo! Italia	0,21630094	0,38369906	0,56	0,04	0,56451613	0,03548387	0,50689655	0,09310345	0,20519481	0,27359307	0,35909091	0,11969697	0,45325253	0,02553535	0,40218182	0,07660606
2002	Asystel Milar	0,19708322	0,35443193	0,55151515	0	0,50797448	0,04354067	0,50679771	0,04471744	0,10883519	0,35177087	0,46060606	0	0,44247196	0,0181341	0,41262626	0,0479798
2002	Bossini Gabe	0,28614216	0,43506996	0,60733652	0,1138756	0,68140496	0,03980716	0,65514689	0,06606523	0,23768454	0,45928516	0,60984848	0,08712121	0,63320906	0,06376063	0,62727273	0,06969697
2002	Canadiens Vc	0,09746698	0,29041181	0,38787879	0	0,35878788	0,02909091	0,34909091	0,03878788	0	0	0	0	0	0	0	0
2002	Copra Venta	0,22961467	0,44917321	0,5312253	0,14756258	0,64927536	0,02951252	0,66223208	0,0165558	0,21387941	0,35581756	0,34487403	0,22482294	0,51182707	0,0578699	0,4661157	0,10358127
2002	Edilbasso & f	0,16655844	0,40919913	0,41125541	0,16450216	0,53246753	0,04329004	0,47562582	0,10013175	0,17211389	0,39758308	0,39440559	0,17529138	0,522441	0,04725597	0,5026738	0,06702317
2002	Estense Carif	0,41683954	0,3225544	0,64885591	0,09053803	0,67464475	0,06474919	0,64041995	0,09897399	0,17410468	0,3046832	0,44098884	0,03779904	0,46010347	0,01868441	0,42245989	0,05632799
2002	Icom Latina	0,36294504	0,39463072	0,62582345	0,13175231	0,72843823	0,02913753	0,69097569	0,06660007	0,14104129	0,43471629	0,4030303	0,17272727	0,54936869	0,02638889	0,5241972	0,05156038
2002	Itas Grundig	0,29381818	0,31830303	0	0	0,5276907	0,08443051	0,56839827	0,04372294	0,14931437	0,39614017	0,35573123	0,18972332	0,51724138	0,02821317	0,47603306	0,06942149

Figura 2.1: Frammento del Dataset Large

• Il **Dataset Small** contiene invece un sottoinsieme dei giocatori della rosa, scelti in base al maggior numero di set giocati nella stagione precedente:

- **centrali** (indicati con la lettera **C**): 1 - 4
- **liberi** (indicati con la lettera **L**): 1 - 2
- **palleggiatori** (indicati con la lettera **P**): 1 - 2

- **schiacciatori** (indicati con la lettera **S**): 1 - 5

Entrambi i dataset hanno inoltre altre features che indicano: la *Stagione*, la *Squadra*, i *Playoff* e la *Posizione* di arrivo a fine campionato.

L'informazione relativa al *Playoff* sarà a tutti gli effetti quella che dovrà individuare il tipo di classificazione necessaria per la predizione, perchè dovrà essere espressa nel seguente modo:

- **1**: se la squadra accede ai Playoff in quella stagione
- **0**: se la squadra non accede ai Playoff in quella stagione

Per ogni giocatore ci sono **4** possibili **azioni di gioco**:

- **battuta** (abbreviata in "*bat_*")
- **ricezione** (abbreviata in "*ric_*")
- **attacco** (abbreviata in "*att_*")
- **muro** (abbreviata in "*mur_*")

Tali azioni, giudicate da un osservatore dotato di conoscenze specifiche sulla pallavolo, possono essere:

- **positive** (abbreviata in "*pos_*")
- **negative** (abbreviata in "*neg_*")

Un'azione **positiva** è composta da: ***azione_pos=azione_vin+azione_piu*** (esempio bat_pos=bat_vin+bat_piu)

Un'azione **negativa** è composta da: ***azione_neg=azione_err+azione_meno*** (esempio bat_neg=bat_err+bat_meno)

Il calcolo di ognuna delle azioni(*battuta, ricezione, attacco e muro*) viene effettuato nel seguente modo:

⋆ **vin** = $\frac{Azioni\ vinte}{Numero\ totale\ azioni}$

⋆ **piu** = $\frac{Azioni\ piu}{Numero\ totale\ azioni}$

⋆ **err** = $\frac{Azioni\ errate}{Numero\ totale\ azioni}$

⋆ **meno** = $\frac{Azioni\ meno}{Numero\ totale\ azioni}$

Dove:

⋆ **vin**: numero di azioni che portano a guadagnare un punto.

⋆ **piu**: numero di azioni positive.

⋆ **err**: numero di azioni che portano alla perdita di un punto.

⋆ **meno**: numero di azioni sbagliate.

2.3 Analisi dei Dati

Tutti i valori ottenuti per le features descritte nella sezione precedente, sono numeri appartenenti all'intervallo [0,1]. Utilizzando questi numeri però il modello ottenuto sarà ragionevolmente scorretto, perché le statistiche non sono pesate e in questo modo non viene data la giusta importanza ai giocatori con il numero maggiore di set.

Definiamo dunque per ogni statistica un fattore (chiamato **factor**) che sia in grado di individuare l'importanza del giocatore considerato. Questo fattore viene calcolato nel seguente modo:

•**factor** = $\frac{Set\ giocati\ nella\ stagione\ precedente}{Numero\ massimo\ di\ set\ nel\ dataset}$

Moltiplicando **factor** per l'azione considerata si otterranno dei nuovi valori che saranno pesati sulla quantità di set giocati dall'atleta durante l'intera stagione. Questo nuovo valore è inserito poi nell'apposita casella nei due nostri dataset.

I giocatori che presentano un numero di set giocati pari a zero per l'intero periodo temporale del dataset vengono eliminati dalla tabella poichè presentano un peso zero nei risultati della squadra.

Nel **Dataset Small** vengono quindi selezionati i giocatori in base al numero di set giocati nella stagione precedente. Se per esempio una squadra presenta 11 schiacciatori nella propria rosa a inizio stagione, di questi saranno selezionati per il Dataset Small i primi 5 con *factor* maggiore.

Nei due Dataset Large e Small compaiono come features per ogni azione solo **azione_pos** e **azione_neg** (relative a ogni giocatore). Tutti gli altri valori (come: azione_vin, azione_piu, azione_err, azione_meno, numero set, factor, ecc...) che servono per calcolare questi due valori sopracitati, non sono stati considerati singolarmente nel presente studio.

Per questa tesi è stato preso in considerazione solo il **Dataset Small** che, avendo un numero più ristretto di giocatori, consente un'analisi più agevole e possibilmente può produrre anche risultati migliori. Prima di essere utilizzato questo dataset è stato sottoposto a una ulteriore fase di pre processamento.

2.4 Pre processamento

Questa fase è di fondamentale importanza perchè si vogliono utilizzare i dati a disposizione per definire un dataset finale che possa essere considerato appropriato per la fase di predizione successiva.

Questa fase di pre processing è consistita di tre passi:

- Il primo passo è quello di scegliere, per il dataset finale, solo le azioni che vengono eseguite più spesso per ogni ruolo. La scelta è stata fatta come segue:
  - Per i **centrali** vengono considerate solo le azioni di **battuta, attacco e muro**
  - Per i **liberi** viene considerata solo l'azione di **ricezione**
  - Per i **palleggiatori** vengono considerate solo le azioni di **battuta e attacco**
  - Per gli **schiacciatori** vengono considerate solo le azioni di **battuta, attacco, ricezione e muro**
- Il secondo passo è quello di fare per ogni squadra e in ogni stagione la media di ogni azione (sia positiva che negativa) per ogni ruolo, in modo da compattare i dati relativi a una singola statistica.
- Il terzo passo è quello di riportare in questo dataset finale le features relative alla *Stagione*, al *Nome Squadra*, ai *Playoff* e alla *Posizione* in classifica a fine anno, presenti già nel Dataset Small.

Il **Dataset Finale** sarà dunque il dataset su cui saranno eseguite tutte le predizioni e sul quale saranno applicati i vari modelli di apprendimento supervisionato, citati nel capitolo precedente.

Tale Dataset Finale appare come segue:

Stagione	Squadra	C_bat_pos	C_bat_neg	C_att_pos	C_att_neg	C_mur_pos	C_mur_neg	L_ric_pos	L_ric_neg	...	Playoff	Posizione
2001	Asystel Milano	0.2395186225	0.4771480400000005	0.6720229575000001	0.04464370925000004	0.64548762375	0.071179043	0.33447279350000003	0.1261332675		1	5
2001	Borgocanale Taranto	0.12398125300000001	0.22298844425	0.324253565	0.02271613175	0.28102857324999997	0.052304760000000006	0.0	0.0		0	10
2001	Bossini Sangemini Montichiari	0.10428700825000001	0.1654099615	0.24605395175	0.023643018	0.22081653975	0.048880429749999996	0.252811516	0.0805218175		1	7
2001	Casa Modena Salumi	0.24602183600000002	0.4085236185	0.6200925107499999	0.03445294375	0.57820467925	0.07634077524999999	0.30665402	0.102436889		1	4
2001	Icom Latina	0.0476	0.1069454545	0.14436222699999998	0.010183227750000001	0.14534632025	0.009199134250000001	0.0	0.0		0	11
2001	Itas Diatec Trentino	0.12144738025	0.2967344377499997	0.39474791875	0.02343389925	0.37149797300000004	0.0466838455	0.245335876	0.07890654799999999		0	9
2001	Lube Banca Marche Macerata	0.1653306145	0.29679059775	0.43555055374999996	0.0265706585	0.41586917824999997	0.046252033750000004	0.2916325605	0.0750341065		1	1
2001	Maxicono Parma	0.1220415	0.33250395450000003	0.42406284625	0.0304826085	0.43224843300000004	0.0222970215	0.3345001355	0.1230756225		1	3
2001	Noicom Brebanca Cuneo	0.1897480255	0.26782773175	0.42830675500000004	0.029269002500000002	0.363826999	0.09374875875	0.2115606935	0.08843930650000001		1	6
2001	Roma Volley	0.0176833135	0.04443789824999996	0.05756757875	0.0045536335	0.050284090749999996	0.011837120999999999	0.103375068	0.0602612955		0	14
2001	Sempre Volley Padova	0.08722057249999998	0.18853700325	0.26007585050000004	0.0156817255	0.2153368795	0.06042069625	0.321145153	0.097036665		0	13
2001	Sira Cucine Ancona	0.0489637305	0.1010362695	0.143312102	0.006687898	0.13369565225	0.01630434775	0.0	0.0		0	12
2001	Sisley Treviso	0.143307151	0.24305648525	0.36131618375	0.025047452749999997	0.3469307285	0.039432908	0.34586978549999997	0.12988779		1	2
2001	Yahoo! Italia Volley Ferrara	0.12974068049999998	0.23692598599999998	0.34413913325	0.0225275335	0.31454231975	0.052124347	0.299020849	0.094918545		1	8
2002	Asystel Milano	0.10590615525	0.254699905	0.33263686400000003	0.027969196749999998	0.32693655425	0.0336695065	0.2441354725	0.095258467		1	4
2002	Bossini Gabeca Montichiari	0.16058050000000001	0.31669222725	0.43910805075000003	0.03816467675	0.4304135165	0.04685921075	0.27209302350000003	0.0824524315		0	10
2002	Canadiens Verona	0.02436674425	0.0726029525	0.08969696975	0.007272727249999995	0.08727272725	0.00969696975	0.0	0.0		0	13
2002	Copra Ventaglio Piacenza	0.121872396	0.23267305850000003	0.32798604675	0.0265594075	0.310369773	0.044175681499999994	0.0	0.0		0	12
2002	Edilbasso & Partner Padova	0.0846680825	0.201695554	0.263727133	0.022636503250000002	0.244574905	0.041788731249999995	0.0795454545	0.0265151515		0	11
2002	Estense Carife Ferrara	0.14773605475	0.16741546024999998	0.29429311725	0.020858398	0.2710229905	0.04412852475	0.31619097949999997	0.0807787175		1	5
2002	Icom Latina	0.22487218575	0.38421872325	0.58090132	0.028189589	0.54816042325	0.060930486000000006	0.173142934	0.0510994905		1	6
2002	Itas Grundig Trentino	0.16842449925000003	0.287636107	0.41467348	0.041387125749999996	0.41932154749999995	0.036739058500000005	0.330549975	0.078540934		1	7
2002	Kerakoll Modena	0.17776082825	0.44042098999999996	0.59108746375	0.027094354499999997	0.5467784640000001	0.071403354	0.376909421	0.0927875485		1	2
2002	Lube Banca Marche Macerata	0.13504679275	0.26495320725	0.37501398625000004	0.0249860135	0.33981003425	0.060189965750000005	0.2522086065	0.10536715099999999		1	3

Figura 2.2: Frammento del Dataset Finale

Capitolo 3

Analisi dei Risultati

3.1 Baseline

Prima di implementare i vari modelli predittivi citati nei capitoli precedenti e osservare i risultati ottenuti, è stata eseguita un'analisi di base per capire quali risultati aspettarsi di ottenere e possibilmente migliorare tramite il presente lavoro.

Sostanzialmente si è voluto osservare quante squadre entrate ai Playoff in una determinata stagione, fossero riuscite a rientrare ai Playoff anche nella stagione successiva.

Il risultato ottenuto è che il **72%** delle squadre che partecipa in una stagione ai Playoff, era riuscita ad accedervi anche nella stagione precedente.

3.2 Considerazioni Preliminari

Per implementare i modelli di apprendimento automatico è stata utilizzata *Scikit-learn*, una libreria open source per il linguaggio di programmazione

Python, la quale contiene algoritmi di classificazione, regressione e clustering (raggruppamento). [20]

3.2.1 Metriche Utilizzate

I risultati dei vari modelli di apprendimento automatico implementati in questo studio di tesi sono espressi tramite varie classificazioni statistiche (dette anche Metriche), ottenute grazie alla così detta *Matrice di Confusione* o *Tabella di Errata Classificazione*. Quest'ultima è composta nella maniera seguente: ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. L'elemento sulla riga i e sulla colonna j è il numero di casi in cui il classificatore ha classificato la classe "vera" i come classe j.

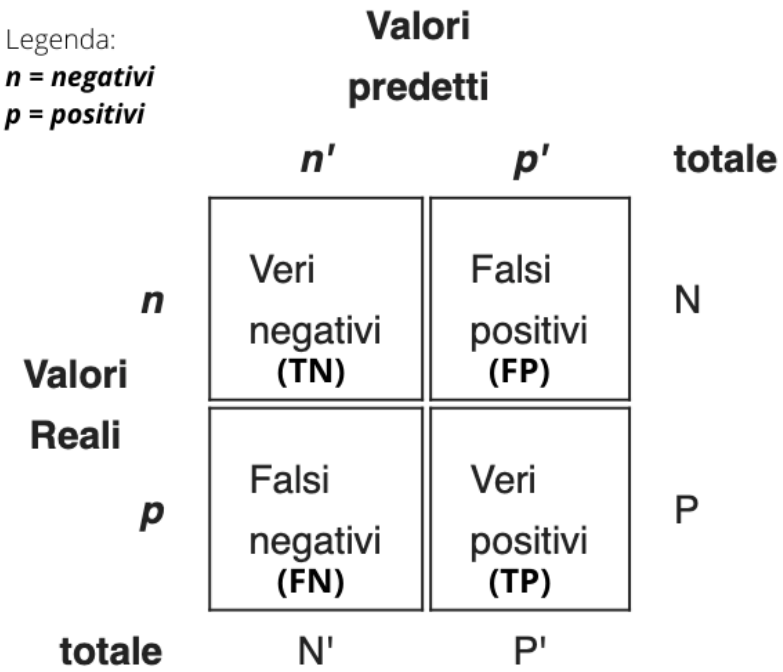

Figura 3.1: Matrice di Confusione per la classificazione delle istanze

La terminologia della Matrice di Confusione:

- **TN** (True Negative) sono quei valori che erano negativi nei valori reali e sono stati predetti correttamente con valore negativo.
- **TP** (True Positive) sono quei valori che erano positivi nei valori reali e sono stati predetti correttamente con valore positivo.
- **FP** (False Positive) sono quei valori che erano negativi nei valori reali e sono stati predetti, sbagliando, con valore positivo.
- **FN** (False Negative) sono quei valori che erano positivi nei valori reali e stati predetti, sbagliando, con valore negativo.
- **N** (N = TN + FP) sono il numero dei casi realmente negativi nei dati.
- **P** (P = FN + TP) sono il numero dei casi realmente positivi nei dati.

Tramite questi valori si possono ricavare le varie metriche, in particolare sono di nostro interesse: *Accuracy*, *Balanced_ Accuracy*, *Precision*, *Recall* e *F1_ score*.

- **Accuracy(ACC)** = $\frac{TP+TN}{P+N}$ è il grado di corrispondenza del dato predetto con il dato reale di riferimento.
- **Balanced_ Accuracy(BACC)** = $\frac{\frac{TP}{P}+\frac{TN}{N}}{2}$ svolge lo stesso ruolo dell'Accuracy, però viene utilizzata sopratutto quando il *test set* non è bilanciato (bilanciato significa stesso numero di prove in ogni classe)
- **Precision(PREC)** = $\frac{TP}{TP+FP}$ può essere vista come una misura di esattezza o fedeltà (esempio: un valore di Precision di 1.0 per la classe C significa che ogni oggetto che è stato etichettato come appartenente alla classe C vi appartiene davvero, ma non dice niente sul numero di elementi della classe C che non sono stati etichettati correttamente).

- **Recall(REC)** $= \frac{TP}{TP+FN}$ può essere vista come una misura di completezza (esempio: un valore di Recall pari ad 1.0 significa che ogni oggetto della classe C è stato etichettato come appartenente ad essa, ma non dice niente sul numero di elementi etichettati non correttamente con C.

- **F1_score(F1)** $= 2 \cdot \frac{PREC \cdot REC}{PREC+REC}$ è la media armonica tra Precision e Recall. Nella classificazione binaria è una misura di accuratezza di un test.

Oltre a queste metriche sopracitate, per questa tesi è stata introdotta da noi anche un'ulteriore metrica, chiamata *Errore*, relativa a ogni i-esima squadra:

$$\textbf{Errore(Squadra i)} = \begin{cases} 0, & \text{se } y^{(i)} = \hat{y}^{(i)} \\ pos^{(i)} - nSquadrePlayoff, & \text{se } y^{(i)} = 0 \ e \ \hat{y}^{(i)} = 1 \\ (nSquadrePlayoff + 1) - pos^{(i)}, & \text{se } y^{(i)} = 1 \ e \ \hat{y}^{(i)} = 0 \end{cases} \tag{3.1}$$

dove:

- ⋆ $y^{(i)}$ indica se realmente la squadra i-esima è entrata(1) o no(0) ai playoff.

- ⋆ $\hat{y}^{(i)}$ indica se nella predizione la squadra i-esima è entrata(1) o no(0) ai playoff.

- ⋆ $pos^{(i)}$ indica la posizione in classifica a fine campionato della squadra i-esima.

- ⋆ $nSquadrePlayoff$ indica il numero complessivo di squadre che in una determinata stagione entrano ai playoff.

3.3 Tuning del Modello di Regressione Logistica

Per questo modello di apprendimento sono disponibili vari parametri su cui è possibile agire per ottenere un migliore risultato in termini di accuratezza. Quelli su cui è stata posta maggiore attenzione sono:

- **C** è l'inverso della forza di regolarizzazione cioè $\frac{1}{\lambda}$. Un valore più piccolo specifica una regolarizzazione più forte. Nel nostro caso prende valore $C \in [2^{-8}, 2^{-7}, ..., 2^{7}]$.
- **solver** indica l'algoritmo da usare nel problema di ottimizzazione. Nel nostro caso è "lbfgs".
- **max_iter** indica il numero massimo di iterazioni affinché il solver converga. Nel nostro caso gli è stato assegnato valore 200.
- Gli altri parametri sono presi con i loro valori di default.

3.4 Tuning del Modello di Support Vector Machine Lineare

Con questo modello di apprendimento supervisionato i parametri su è stato deciso di agire sono i seguenti:

- **C** è il parametro di penalità del termine di errore. Nel nostro caso prende valore $C \in [2^{-8}, 2^{-7}, ..., 2^{7}]$.
- **max_iter** è il numero massimo di iterazioni che devono essere eseguite. Nel nostro caso gli è stato assegnato valore 20000.
- Gli altri parametri sono presi con i loro valori di default.

3.5 Tuning del Modello di Support Vector Machine Non Lineare

Anche il modello non lineare del Support Vector Machine presenta dei parametri a cui è stato assegnato un valore, questi sono:

- **C** è il parametro di penalità del termine di errore. Nel nostro caso prende valore $C \in [2^{-8}, 2^{-7}, ..., 2^{7}]$.
- **kernel** specifica il tipo di kernel che deve essere usato nell'algoritmo. Può essere "linear", "poly", "rbf" e "sigmoid". Nel nostro caso ha valore "rbf" ovvero kernel gaussiano.
- **gamma γ** è un coefficiente del kernel per i tipi "rbf", che corrisponde ad $\frac{1}{2\sigma^2}$. I valori possibili per questa variabile sono $\gamma \in \{\gamma_0 \cdot 2^i \mid i = -8, ..., 7\}$ dove $\gamma_0 = \frac{1}{numeroFeatures} = \frac{1}{20}$.
- Gli altri parametri sono presi con i loro valori di default.

3.6 Analisi dei Risultati per la Regressione Logistica

Prima di procedere alla fase predittiva per questo modello di apprendimento supervisionato, è necessario scomporre l'insieme dei dati in *training set* e *test set*. Questa partizione vine effettuata secondo un criterio temporale, quindi le stagioni dal 2001/02 a 2014/15 andranno a definire il *training set*, mentre le restanti stagioni dal 2015/16 fino al 2017/18 saranno utilizzate in fase di testing.

In questo modello è stato scelto di produrre i risultati in termine della metrica di *F1_score*.

La scelta del parametro **C**, affinché renda un risultato migliore di F1_score in fase di testing, viene fatta attraverso la tecnica della *Cross-Validation* (*Convalida Incrociata*). In *scikit-learn* viene eseguita tramite la funzione "cross_val_score()", la quale prende in ingresso il valore di ritorno di un'altra funzione chiamata "StratifiedShuffleSplit()", che è un unione della *StratifiedKFold* e *ShuffleSplit* e che consiste nella suddivisione del training set in altre due parti chiamate ancora una volta *training set* e *test set*, questa volta divise in modo non temporale, ma assegnando al primo l'80% del training set iniziale e al secondo il 20% del training set iniziale. Tale funzione "StratifiedShuffleSplit()" prende in ingresso anche un valore n che indica il numero di rimescolamento e le iterazioni di suddivisione da eseguire sul nostro training set iniziale. Nel nostro caso assume valore 10.

Quindi una volta scelto il parametro **C** che rende il migliore risultato di *F1_score* tramite la Cross-Validation, riutilizzo tale valore (chiamato **C_max**) per riaddestrare l'intero dataset.

Per ottenere un risultato più preciso, tutti questi passaggi sopraelencati vengono eseguiti per 50 iterazioni e del valore di *F1_score* ne viene fatta la media.

Il risultato di *F1_score* totale dunque è il seguente:

Classificatore	**F1_score**	**Media C_max su 50 iterazioni**
Regressione Logistica	75,9%	$30{,}04 \approx 2^5$

Tabella 3.1: Risultato per la Regressione Logistica

3.7 Analisi dei Risultati per SVM (Modello Lineare)

Essendo il modello lineare dell'SVM, il kernel che è stato utilizzato è dunque di tipo lineare ($k(x_i, y_i) = x_i \cdot y_i$).

Anche per questo modello di apprendimento la suddivisione dei dati è eseguita secondo un criterio temporale, dove le stagioni dal 2001/02 a 2014/15 andranno a definire il *training set*, mentre le restanti stagioni dal 2015/16 fino al 2017/18 saranno utilizzate in fase di testing.

I risultati anche in questo caso sono espressi in termine della classificazione statistica dell'*F1_score*.

La scelta del parametro **C** con cui addestrare tutto il dataset, avviene in maniera analoga al caso della Regressione Logistica, ovvero tramite l'operazione di *Cross-Validation* applicata alla parte *training set* del dataset.

Quindi una volta scelto il parametro **C** che rende il migliore risultato in termini di *F1_score* con la Cross-Validation, riutilizzo tale valore (chiamato **C_max**) per riaddestrare l'intero dataset.

Come nel caso della Regressione Logistica, si eseguono 50 iterazioni del codice, così da ottenere un valore di *F1_score* più preciso, facendone la media.

Il risultato di *F1_score* totale dunque è il seguente:

Classificatore	**Tipo di kernel**	**F1_score**	**Media C_max su 50 iterazioni**
Support Vector Machine (SVM)	linear	73,3%	16,885 $\approx 2^4$

Tabella 3.2: Risultato per SVM modello lineare

3.8 Analisi dei Risultati per SVM (Modello Non Lineare)

Per questo modello non lineare del Support Vector Machine è stato utilizzato un kernel di tipo gaussiano ($k(x_i, y_i) = \mathrm{e}^{\frac{-\|x_i - y_i\|^2}{2\sigma^2}}$). Sono state eseguite diverse implementazioni di questo modello, andando a perfezionare il risultato finale.

• Prima implementazione:

In questo caso il dataset viene suddiviso come i modelli precedenti per quanto riguarda il *training set* (stagioni dal 2001/02 al 2014/15) e il *test set* (stagioni dal 2015/16 al 2017/18). Per adesso verrà valutato solo *F1_ score*, poi nelle implementazioni successive di questo modello, saranno mostrati i risultati anche di altre metriche.

Oltre al solito parametro **C**, nel caso non lineare di SVM, c'è anche un altro parametro **gamma** $\boldsymbol{\gamma}$. La coppia migliore di questi due valori con cui addestrare tutto il dataset, viene scelta sempre tramite la procedura di *Cross-Validation* applicata alla parte del *training set* del dataset. Dunque questa procedura restituisce la coppia **C** e **gamma** (chiamati **C_max** e **gamma_max**) che ha prodotto il migliore risultato in termine di *F1_ score*. Come nei casi precedenti, si eseguono 50 iterazioni del codice così da ottenere un valore di *F1_ score* più preciso, facendone la media.

Classificatore	**Tipo di kernel**	**F1_score**
Support Vector Machine (SVM)	gaussiano (o radial basis function "rbf")	82,5%

Tabella 3.3: Risultato per SVM modello non lineare, prima implementazione

• Seconda implementazione:

In questa seconda implementazione viene cambiata la suddivisione del dataset tra *training set* e *test set*. Viene eseguito infatti un ciclo dove ad ogni iterazione si scorre un anno diverso. Dunque l'anno alla i-esima iterazione andrà a comporre il *test set*, mentre tutti gli altri anni restanti andranno a formare il *training set*.

Anche in questo caso si considera come metrica solo la *F1_score*. Rimane anche invariato il metodo per selezionare la migliore coppia di **C** e **gamma**. Vengono eseguite sempre 50 iterazioni di tutto il codice.

Il risultato di *F1_score* totale dunque è il seguente:

Classificatore	**Tipo di kernel**	**F1_score**
Support Vector Machine (SVM)	gaussiano (o radial basis function "rbf")	80,9%

Tabella 3.4: Risultato per SVM modello non lineare, seconda implementazione

• Terza implementazione:

Questa terza implementazione è sostanzialmente uguale alla precedente, l'unica differenza è stata apportata al numero di iterazioni del codice, che da 50 iterazioni è stato portato ad una sola iterazione.

Dunque il risultato di *F1_score*, ottenuto facendone la media tra il risultato di tutti gli anni, variando di anno in anno il *test set*, è il seguente:

Classificatore	**Tipo di kernel**	**F1_score**
Support Vector Machine (SVM)	gaussiano (o radial basis function "rbf")	81,4%

Tabella 3.5: Risultato per SVM modello non lineare, terza implementazione

• Quarta implementazione:

Per tutte queste implementazioni finora elencate però sorge un problema ovvero quello che non c'è un vincolo sul numero di squadre che devono accedere ai playoff in una determinata stagione. Questo comporta che il modello può predirne un numero superiore o inferiore rispetto alle squadre che poi realmente sono entrate ai playoff. Tale problematica si risolve in due passi:

⋆ Viene introdotta una variabile che indichi il numero di squadre che entrano ai playoff per ogni diversa stagione (*nSquadrePlayoff*).

⋆ Viene usata come funzione predittiva, al posto di "predict( )", "predict_proba( )", la quale invece che rendere come output valori 0 o 1 della colonna PredictedPlayoff, rende il risultato in termini di probabilità. Quindi si decide di mettere a 1 la colonna PredictedPlayoff alle n squadre che hanno la probabilità più alta. Dove n = *nSquadrePlayoff* e varia da stagione a stagione.

Per quanto riguarda la suddivisione del dataset è identica alla seconda e terza implementazione. La coppia di parametri **C** e **gamma** vengono scelti sempre in ugual modo e si esegue anche in questo caso solo una iterazione del codice.

In questa implementazione i risultati però non vengono dati solo in termine di *F1_score* come nei punti precedenti, ma vengono generate anche altre metriche come: *Accuracy, Balanced_Accuracy, Precision* e *Recall*. Si introduce anche un'ulteriore classificazione statistica: *Errore*, creata appositamente per questo studio di tesi.

I risultati di queste metriche è dunque il seguente:

	Support Vector Machine (SVM) con kernel gaussiano (o radial basis function "rbf")
F1_score	81%
Accuracy	76,6%
Balanced_Accuracy	73,4%
Precision	81%
Recall	81%
Errore	0,65

Tabella 3.6: Risultato per SVM modello non lineare, quarta implementazione

Si può notare che un *Errore* pari a 0,65 indica che il modello commette nella maggior parte dei casi errori sulle squadre che si qualificano sulla soglia dei playoff. Supponiamo infatti che le squadre che entrino ai playoff siano 8; è facile comprendere che tra la squadra che è arrivata ottava e la squadra che è stata esclusa dai playoff arrivando nona, non ci siano enormi differenze statistiche, ma semplicemente sono differenziate da fattori non prevedibili all'interno della stagione, quali, per esempio, un infortunio di un giocatore importante o un esordiente particolarmente forte o anche uno scontro diretto deciso sul filo di lana.

Un esempio di output per un test in una determinata stagione con questo modello di apprendimento supervisionato è:

```
TEST SU STAGIONE: 2008

                          Squadra  Stagione  PlayoffProbability  PredictionPlayoff  RealPlayoff  Posizione  Errore
Acqua Paradiso Gabeca Montichiari      2008            0.785409                  1            1          6       0
              Antonveneta Padova       2008            0.382497                  0            0         14       0
         Bre Banca Lannutti Cuneo      2008            0.875561                  1            1          3       0
      Copra Nordmeccanica Piacenza     2008            0.827118                  1            1          5       0
              Framasil Pineto          2008            0.101979                  0            0         12       0
            Itas Diatec Trentino       2008            0.878957                  1            1          2       0
      Lube Banca Marche Macerata       2008            0.878342                  1            1          1       0
              Marmi Lanza Verona       2008            0.705287                  0            0          9       0
       RPA-LuigiBacchi.it Perugia      2008            0.818195                  1            1          4       0
                 Sisley Treviso        2008            0.861351                  1            1          8       0
         Stamplast Martina Franca      2008            0.257254                  0            0         11       0
      Tonno Callipo Vibo Valentia      2008            0.702552                  0            1          7       2
              Trenkwalder Modena       2008            0.839978                  1            0         10       2
                      Yoga Forl        2008            0.757318                  0            0         13       0

Numero Squadre Playoff= 8
Numero Squadre Stagione=14

f1 del C_max e gamma_max=0.875
accuracy del C_max e gamma_max=0.8571428571428571
balanced_accuracy del C_max e gamma_max=0.8541666666666667
precision del C_max e gamma_max=0.875
recall del C_max e gamma_max=0.875
errore medio in questo anno=0.2857142857142857

C_max e gamma_max=[(0.5, 6.4)]
```

Figura 3.2: Esempio di output per il test sull'anno 2008 con questa implementazione

3.9 Confronto tra i Risultati

Come è stato possibile vedere nelle sezioni precedenti il modello di apprendimento che ha reso il migliore risultato in termine di *F1_score*, è stato il Support Vector Machine non lineare, con kernel gaussiano, il quale ha dato un notevole risultato di *F1_score* nell'intorno dell'**81%**.

Questo è un notevole miglioramento rispetto agli altri modelli implementati, infatti la Regressione Logistica produceva un valore di *F1_score* pari a **75%** e anche lo stesso Support Vector Machine lineare non riusciva a dare un risultato così importante, dato che anche esso si fermava in un intorno del **74%**.

Di seguito è presentata una tabella che riassume i risultati ottenuti:

	F1_score
Regressione Logistica	75,9%
SVM kernel lineare	73,3%
SVM kernel gaussiano 1ª implementazione	82,5%
SVM kernel gaussiano 2ª implementazione	80,9%
SVM kernel gaussiano 3ª implementazione	81,4%
SVM kernel gaussiano 4ª implementazione	81%

Tabella 3.7: Confronto tra i risultati dei vari modelli predittivi

Capitolo 4

Conclusioni

Il presente studio si proponeva di definire un modello predittivo con la capacità di prevedere a inizio stagione del campionato di Serie A di pallavolo maschile le squadre partecipanti alla fase finale del campionato, i Playoff. Per questo tipo di classificazione sono stati utilizzati dati relativi ai singoli giocatori delle precedenti stagioni. Con i dati forniti dalla Lega, si voleva definire un dataset che potesse essere utilizzato dai modelli di classificazione.

Le tabelle forniteci quindi sono state filtrate e trasformate affinché si ottenesse il dataset desiderato.

La fase successiva prevedeva l'implementazione di metodi di classificazione che fossero in grado di eseguire una classificazione binaria tra squadre che accedono ai Playoff e squadre che invece non riescono a raggiungere tale risultato.

Sono stati utilizzati metodi di apprendimento supervisionato come la Regressione Logistica, Support Vector Machine Lineare e Support Vector Machine Non Lineare.

Maggiore interesse è stato dato al modello dell'SVM Non Lineare con kernel *gaussiano*, il quale presentava un insieme di parametri maggiore ri-

spetto ai precedenti e ha permesso dunque di classificare con un buon valore di *F1_score* di circa **0.81**.

Analizzando bene il dataset, si può però notare l'incompletezza dei dati disponibili. Ci sarebbe infatti la necessità di integrare informazioni su atleti provenienti da campionati differenti dalla serie A italiana. Nella maggior parte dei casi questi giocatori "esordienti" sono di fondamentale importanza per il proprio team e considerarli con peso nullo va a penalizzare irrevocabilmente la classificazione finale, motivo per cui quasi tutti gli errori si manifestano assegnando alla casella PredictedPlayoff 0 a squadre che invece avevano la casella Playoff corretta con valore pari a 1.

Gli sviluppi futuri possono comprendere lavori di raffinamento dei modelli proposti in questo studio, con lo scopo di renderli più robusti; il dataset definito in questa tesi può essere reso completo o integrato con dati relativi alle squadre nel loro complesso, oltre a comprendere anche i giocatori provenienti da altri campionati, portando così a notevoli miglioramenti di classificazione. Una sfida interessante e di maggior rilievo potrebbe essere la trasposizione di questo studio in altri sport.

Bibliografia

[1] OLSPS Analytics. *How machine learning in sports analytics is disrupting the game.* Luglio 2018. `http://www.bizcommunity.com`.

[2] Ethem Alpaydin. *Introduction to Machine Learning.* 2004.

[3] Christopher M.Bishop. *Pattern Recognition and Machine Learning.* 2006.

[4] Luis Pedro Coelho - Willi Richert. *Building Machine Learning Systems with Python.* Luglio 2013.

[5] Subash Thota. *The Evolution of Machine Learning.* Synectics, Published Online.

[6] Drikos S. *A longitudinal study of the success factors in high-level male volleyball. Journal of Physical Activity, Nutrition and Rehabilitation*, 2018.

[7] Georgie L. *Machine Learning in sport betting.* Ottobre 2018. Published Online.

[8] Shourjya Sanyal. *How are Wearables Changing Athlete Performance Monitoring?. Forbes*, Novembre 2018.

[9] Rory P. Bunker - Fadi Thabtah. *A machine learning framework for sport result prediction.* Novembre 2018.

[10] Vladimir Pridal. *Analysis of relation between team placing in tournament and selected indicators of playing performance in top-level volleyball.* Settembre 2018. Published Online.

[11] João P.V. y Moreno M.P. Rentero L. *Analysis of the influence of the libero in different phases of game in volleyball.* 2015.

[12] Alexandros Laios Yiannis Laios Sotiris Drikos, Panagiotis Kountouris. *Correlates of Team Performance in Volleyball.* 2009.

[13] Kountouris Panagiotis Laios Yiannis. *Evolution in men's volleyball skills and tactics as evidenced in the Athens 2004 Olympic Games. International Journal of Performance Analysis in Sports*, 2005.

[14] Kountouris Panagiotis Alexandros Laios. *Receiving and serving team efficency in Volleyball in relation to team rotation. International Journal of Performance Analysis in Sports*, 2011.

[15] Peter Russell, Stuart; Norvig. *Artificial Intelligence: A Modern Approach.* 2003.

[16] David G. Kleinbaum - Mitchell Klein. *ntroduction to Logistic Regression.* 2010.

[17] Sunil Ray. *Understanding Support Vector Machine algorithm for examples.* Settembre 2017. `http://www.analyticsvidhya.com`.

[18] Nello Cristianini - John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning*, 2000.

[19] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* Dicembre 2011.

[20] Raul Garreta - Guillermo Moncecchi. *Learning Scikit-learn: Machine Learning in Python.* 2013.