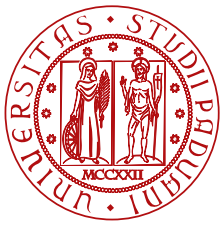

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Previsione di risultati calcistici grazie al Machine Learning

Relatore

Prof. Chiariotti Federico

Laureando

Coron Alessandro

Anno Accademico 2023-2034

A papà, sei sempre con me

Sommario

Il presente documento tratta l'uso di moderni strumenti di Machine Learning per provare a prevedere al meglio il risultato di incontri calcistici, prendendo nello specifico il campionato italiano. Questo è possibile grazie ad un vasto dataset che contiene risultati, goal, classifiche e via discorrendo, di tutte le partite disputate negli ultimi 6 anni di Serie A, opportunamente modificato ed elaborato per consentire ai modelli di ML di perfezionarsi al meglio.

Partendo dal presupposto che nel calcio nulla è certo e che ogni partita ha la sua storia, l'obiettivo è di ottenere un risultato quanto più accettabile possibile.

Attraverso l'utilizzo di diverse tecniche di ML, quali Support Vector Machines, Random Forests e Gradient Boosting, siamo in grado di trovare una correlazione tra i dati disponibili prima dell'evento e l'esito della partita, al fine di poter prevedere quest'ultimo col solo utilizzo dei dati disponibili precedentemente. Questi ultimi vengono sia presi così come sono, sia vengono elaborati in modo da ottenere ulteriori indici riguardanti l'incontro, ad esempio si può intuire la forma delle squadre in base alle prestazioni ottenute negli ultimi match, tarandola a sua volta sulla forma delle squadre affrontate.

Una buona parte del lavoro riguarda infatti capire quali features hanno più influenza sull'esito finale e come crearne altre in grado di rendere più efficace il modello allenato su di esse.

A loro volta, andranno poi trovati i parametri adatti ad ogni modello volti a perfezionare il risultato finale.

Indice

Elenco delle figure

Glossario

DL Il deep learning è un sottoinsieme di machine learning che utilizza reti neurali multilivello, chiamate reti neurali profonde, per simulare il complesso potere decisionale del cervello umano. Richiede generalmente più lavoro, maggiore quantità di dati e maggiore potenza di calcolo rispetto al ML.. 3

feature Una feature è una proprietà individuale e misurabile di un fenomeno osservato.[1] La scelta di caratteristiche discriminanti, ad alto contenuto informativo e indipendenti fra loro è un passo cruciale per ottenere un efficiente algoritmo di riconoscimento di pattern, classificazione e regressione. . 6

gradiente Il gradiente di una funzione in un punto fornisce direzione e verso nei quali la funzione cresce più rapidamente. Nel verso opposto al gradiente avviene la massima decrescenza. 11

ML Il termine Machine Learning descrive un processo attraverso il quale i computer sono capaci di apprendere e migliorare le proprie prestazioni senza essere esplicitamente programmati per farlo. Questo significa che, invece di seguire istruzioni statiche, i sistemi di machine learning analizzano i dati e, tramite algoritmi, apprendono da essi per fare previsioni o prendere decisioni basate su nuove informazioni.. 5

Capitolo 1

Introduzione

Il calcio è sicuramente lo sport più famoso, apprezzato e probabilmente anche praticato in Italia, seguito da milioni di tifosi che si dilettano anche nel tentativo di prevederne i vari esiti. Tolto l'intuito e la conoscenza personale, esiste un metodo più pratico e schematico per tale scopo? Chiaramente la risposta è no, o meglio, pronosticare correttamente ogni risultato è impossibile: troppe le variabili in campo o i singoli eventi imprevedibili che possono influenzare l'andamento di una partita, primo tra tutti forse il fattore umano. Tuttavia rimane che dando un attento sguardo a statistiche e dati di match trascorsi, qualche tipo di correlazione tra questi e il risultato la si può trovare. Entra dunque in gioco il ramo dell'intelligenza artificiale chiamato *Machine Learning*, trattato nel Capitolo 2, in grado di scovare *pattern* impercettibili all'uomo, analizzando migliaia di dati grazie a diverse tecniche.

Questi algoritmi, esposti nel prossimo capitolo, sono molto potenti, ma devono avere a disposizione la giusta quantità di dati, e soprattutto questi devono essere significativi e accuratamente elaborati; di questo si tratterà nel Capitolo 3, dove partendo da dati grezzi e semplici e combinandoli tra loro si cercherà di creare dei nuovi attributi in grado di rappresentare le chance di vittoria dell'una o dell'altra squadra.

In questa serie di esperimenti verranno utilizzati solo dati relativi a partite passate e non relativi alla valutazione singola della squadra, come altri studi simili hanno fatto (vedi prossima sezione), causa irreperibilità di questi ultimi. Va anche detto che non esistono, o per lo meno non gratuiti, dataset che contengano statistiche meno generali e più precise, quali possesso palla, metodo di realizzazione dei goal, abilità nel contropiede e via dicendo, che ritornerebbero un risultato più preciso. Nel Capitolo 3 si può trovare la descrizione delle statistiche utilizzate e di quelle create. Infine nei Capitoli 4 e 5 vi

saranno i risultati e le conclusioni anche in relazione a lavori già esistenti.

1.1 Stato dell'arte

L'argomento trattato in questa tesi risulta molto gettonato e nel web sono presenti diversi lavori simili, principalmente su campionati di altre nazioni. Tale interesse è suscitato anche dalle scommesse sportive, tanto che diversi studi valutano i risultati non tanto sulla precisione della previsione, cioè se vince squadra di casa, ospite o pareggio, ma sul profitto che quella previsione avrebbe portato se fosse stata giocata, in relazione alle quote date dalle case di betting.

La previsione di esiti sportivi è naturalmente applicata a una moltitudine di sport, dalla Formula 1 al basket; sport tra di loro troppo diversi, pertanto ci limiteremo a studi riguardanti il solo calcio.
La maggior parte di questi raggiungono una precisione tra il 60% e il 80% sull'esito finale. Uno dei primi studi a riguardo, fatto da Shin e Gasparyan [1], è incentrato sul campionato spagnolo e va a mostrare la differenza di performance nell'utilizzo del solo valore delle squadre, tramite punteggi assegnati nel videogioco FIFA 15 di EA Sports [2], e dati reali, basati sulle prestazioni medie delle squadre negli scontri precedenti. Sorprendentemente il loro modello più performante è una SVM (vedi Capitolo 2) che ha ottenuto il 79% di precisione con i dati del videogame e il 73% con quelli reali.

Prasetio e Harlili [3] hanno applicato la *logistic regression* su dati da loro elaborati indicanti l'offensività e la difensività della squadra di casa e ospite, ottenendo con questi soli 4 indici una precisione del 69.5%. Va però detto che sono andati a prevedere solo la vittoria di una delle squadre e non il pareggio, risultato più imprevedibili che in ogni studio ha ottenuto il valore di precisione più basso.

In opposizione al precedente, Snyder [4] ha condotto diversi esperimenti sul campionato inglese, usando però un ben più alto numero di feature con le più disparate informazioni: capacità dello stadio, distanza percorsa dalla squadra fuori casa per arrivare alla partita, soldi spesi sul calciomercato, soldi spesi in stipendi e altre statistiche non viste in altri studi. Nonostante ciò, la precisione massima raggiunta del 51% evidenzia come non sia tanto la quantità di attributi a disposizione per allenare il modello, quanto la qualità ad aumentare le performance.

Chen [5], come altri, ha usato i dati del videogioco FIFA 19 [2] per prevedere i risultati de La Liga, prima serie spagnola; ha però voluto mettere a confronto gli algoritmi

di ML quali *Random Forests* (2.3) e *SVM* (2.2) con quelli di Deep Learning, ovvero le reti neurali, strutture molto più complesse che richiedono un lavoro maggiore dovuto al perfezionamento della struttura stessa in relazione ai dati specifici. Queste ultime hanno ottenuto il miglior risultato, con il 57% di precisione, mentre gli altri due sono arrivati intorno all'54%.

Infine, Taspinar et al. [6] hanno condotto lo studio più simile a quello trattato in questa tesi: il campionato in questione è proprio la Serie A e alcuni dei dati utilizzati sono simili a quelli esposti nel Capitolo 3, con la differenza che le *feature* utilizzate sono molto più dettagliate e sono in grado di rappresentare perfettamente le partite giocate. Non è necessario elencarle tutte per denotarne la specificità ma ne bastano alcune: vittorie in duelli aerei, dribbling riusciti, recuperi palla, contrasti, passaggi chiave. Insomma dati incredibilmente mirati che hanno portato a una soprendente precisione del 89.63%.

Risulta interessante anche questo articolo [7] che tratta dell'influenza del fattore casa sulle partite, ovvero l'apporto che può dare il sostegno dei propri tifosi. Generalmente la squadra di casa vince con una percentuale del 45%, contro il 30% della squadra ospite e un 25% di pareggi, ma si è visto che durante il periodo di pandemia da COVID-19, dove non era permesso ai tifosi di essere allo stadio, il dato su vittorie in casa e fuori è passato rispettivamente a 40% e 35%.

Capitolo 2

Machine Learning

Il termine Machine Learning fa riferimento al rilevamento automatico di patterns significativi nei dati. Già da decenni siamo circondati da tecnologie basate sul ML: motori di ricerca che imparano a proporci i risultati migliori, programmi anti-spam che imparano a filtrare le nostre mail, e le transizioni online sono protette da software che imparano a riconoscere le frodi.
Una caratteristica comune di tutte queste applicazioni è che, in contrasto col tradizionale uso dei computer, un umano non può fornire una specifica esplicita e dettagliata di come questi compiti dovrebbero essere eseguiti [8].

L'idea fondamentale alla base del machine learning riguarda l'esistenza di una relazione matematica tra una combinazione di dati di *input* e di *output*. Il modello di *machine learning* non conosce questa relazione in anticipo, ma può dedurla se dispone di un numero sufficiente di dati. Questo significa che ogni algoritmo viene costruito sulla base di una funzione matematica modificabile. È possibile capirne il funzionamento in questo modo:

1. Noi "addestriamo" l'algoritmo fornendogli le seguenti combinazioni di input/output $(i, o) : (2, 10), (5, 19), (9, 31)$

2. Secondo i calcoli dell'algoritmo, la relazione tra input e output è: $o = 3i + 4$

3. In seguito, forniamo un input di 8 e chiediamo di predire l'output. L'algoritmo è capace di stabilire automaticamente che l'output è pari a 28.

Nonostante questa sia una comprensione di base, il machine learning si basa sul principio secondo cui tutti i punti dati possono essere collegati matematicamente dai sistemi informatici purché questi dispongano di quantità di dati e potenza di calcolo sufficienti per elaborare tali dati. Di conseguenza, la precisione dell'output è direttamente correlata

al valore dell'input fornito[9].

L'esempio riportato sopra può risultare banale, tanto che basterebbe tracciare una retta in un grafico bidimensionale per arrivare a una relazione valida. Pensiamo ora al caso in cui invece di una singola feature in input, ne avessimo addirittura qualche decina: risulterebbe ora impossibile rappresentare una funzione, che evidenzi una relazione tra questi input e un output, in un grafico comprensibile all'occhio umano; è dunque questo il vero potenziale degli algoritmi di ML.

I due principali compiti svolti da questi algoritmi sono classificazione e regressione:

- classificazione quando gli input portano ad un output il cui dominio è composto da un numero finito di classi;

- regressione quando l'output può assumere un qualsiasi valore numerico.

Ci sono poi due tipi principali di apprendimento per queste due *task*: si parla di *supervised learning* quando l'algoritmo ha a disposizione gli output, in questo caso *label*, alle quali associare ciò che viene dato in input. Si pensi ad esempio a diverse immagini di veicoli, ognuna con una label corrispondente alla tipologia del veicolo stesso; mentre c'è poi l'*unsupervised learning*, dove l'algoritmo cerca *pattern* nei soli input, avendo solo quelli a disposizione.

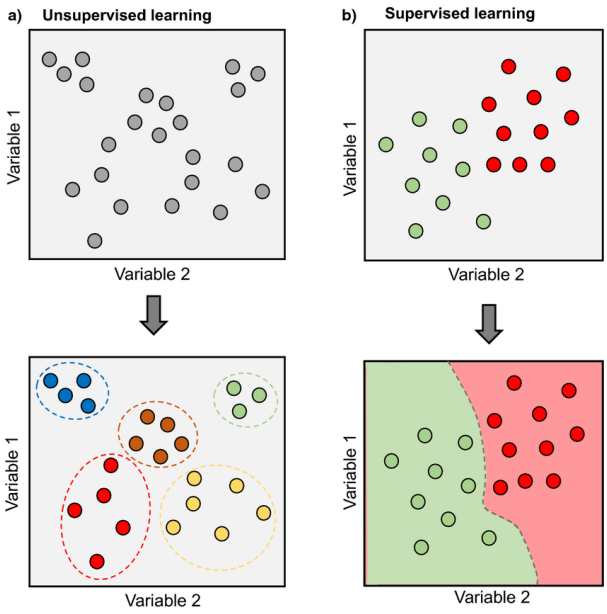

Figura 2.1: Rappresentazione grafica unsupervised e supervised learning. Immagine presa da [10]

Nel Capitolo 4, per risolvere il problema di classificazione esposto nel Capitolo 1, verrano applicati i principali algoritmi di *supervised learning*, questi ultimi esposti nelle sezioni successive.

2.1 Supervised learning

Una parte degli algoritmi di *supervised learning* si basa su un approccio cosiddetto geometrico: dato un *training set* di N coppie input-output

$$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$$

dove ogni coppia è stata estratta da una distribuzione congiunta $D_{X,Y}$, l'obiettivo è trovare un'ipotesi $h*$, estratta da uno spazio di ipotesi H, tale che

$$h* = argmax_{y \in Y} D(y|x)$$

cioè che si avvicini il più possibile a y_i, che viene chiamata *ground truth*.

La vera misura della qualità di un'ipotesi non sta nella precisione con cui apprende l'andamento del *training set*, ma su come si comporta con dati mai visti prima, contenuti in una porzione del dataset generalmente più ridotta, detto *test set*. L'abilità di un modello di apprendere *patterns* compatibili con dati mai visti è detta generalizzare; qualora ottenesse buoni risultati solo sul *training set* e non sul *test set* si parla di *overfitting*, mentre se non riuscisse a trovare una funzione che raggiunga un livello di errore accettabile si parla di *underfitting*. L'obiettivo dell'addestramento del modello è dunque quello di minimizzare entrambi questi fenomeni, e questo compito prende il nome di *Bias-Variance tradeoff* (per approfondimento si rimanda a [8]).

Introduciamo ora il concetto di *decision tree*: è un predittore in grado di associare a dei determinati valori degli attributi in input una label in output, partendo da un nodo radice e arrivando a una foglia; il percorso dalla radice alla foglia esamina i possibili valori degli attributi, prendendo questi uno per volta.

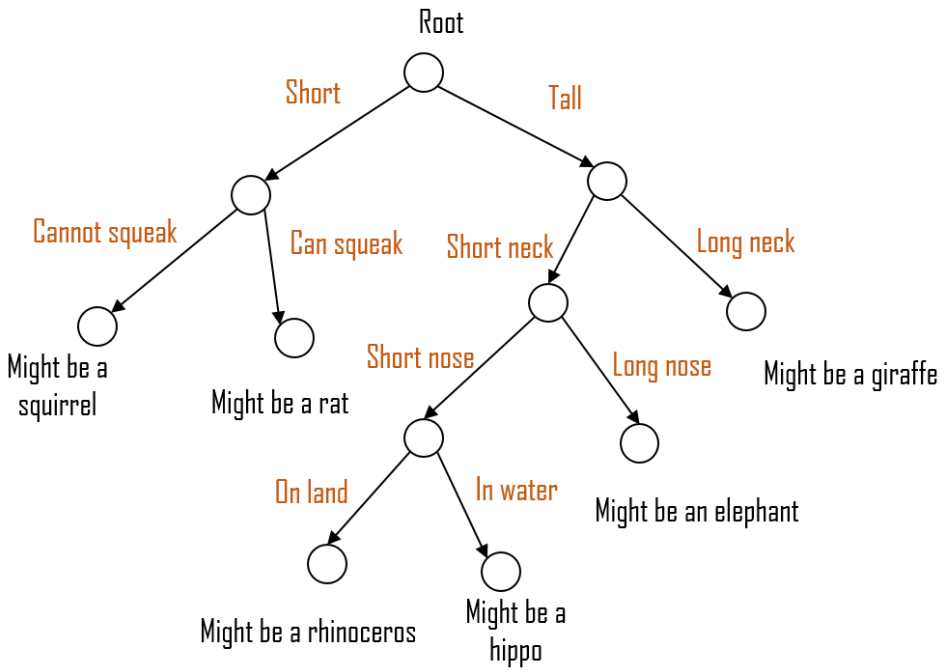

Figura 2.2: Esempio di decision tree sulla classificazione di animali. Immagine presa da [11]

Questo albero risulta facilmente rappresentabile e dunque comprensibile con poche *feature* che assumono pochi valori discreti, e va via via complicandosi all'aumentare di queste o del loro dominio, o in caso di dominio continuo.

Questi due approcci sono le fondamenta su cui si basano gli algoritmi che verranno ora esposti e usati nel Capitolo 4.

2.2 SVMs

Le *Support Vector Machines* vanno a creare un massimo margine separatore fra i dati, vale a dire un iperpiano lineare che suddivide i dati nelle classi di appartenenza ed è collocato il più possibile distante da i dati di ogni classe.

2.2.1 Kernel trick

Chiaramente nel caso di dati linearmente separabili il compito risulta abbastanza semplice, ma qualora i dati non lo fossero, le SVMs sfruttano il cosiddetto *kernel trick*: lo spazio del dominio delle *feature* viene aumentato fino a N dimensioni, andando a rimappare i valori di queste su nuove funzioni, vedi Fig. 2.3, in modo che i dati diventino ora linearmente separabili.

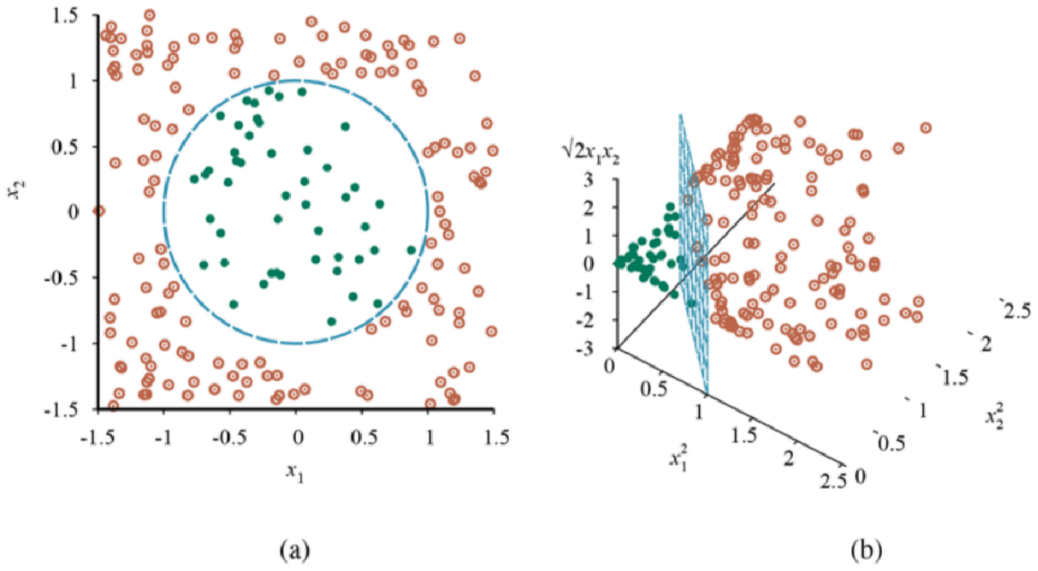

Figura 2.3: Esempio preso da [12], dove i due attributi in (a) vengono rimappati usando le funzioni scritte sugli assi di (b).

Le *kernel functions*, definite come $K(x_j, x_k)$ dove x_j, x_k sono i vettori rappresentanti le *feature*, più comuni sono:

- lineare $K(x_j, x_k) = x_j \cdot x_k$
- polinomiale $K(x_j, x_k) = (1 + x_j \cdot x_k)^d$, d indica l'ordine del polinomio
- gaussiana $K(x_j, x_k) = e^{-\gamma|x_j - x_k|^2}$, γ indica l'ampiezza della curvatura

Per l'applicazione pratica di questo algoritmo, utilizziamo la libreria Scikit-Learn, la cui classe *SVC* richiede il perfezionamento di due parametri principali, C e γ (nel solo caso del kernel gaussiano), oltre alla scelta del kernel da utilizzare.
C è un parametro di regolarizzazione, utile quando ci sono dati con molto rumore, nei quali pochi esempi meno in linea con gli altri porterebbero all'*underfitting*; C permette a questi esempi di eccedere nella parte opposta dell'iperpiano separatore a favore di un miglior margine complessivo.

2.3 Random Forests

L'idea alla base del modello *Random Forests* sta nel selezionare a caso delle *feature* su cui andare a creare un *decision tree*, e ripetendo questo procedimento si crea una foresta

nella quale ogni albero considera un insieme diverso di attributi. Per problemi di regressione si prende poi la media delle predizioni di ogni albero, mentre per la classificazione si sceglie la classe più scelta tra gli alberi.

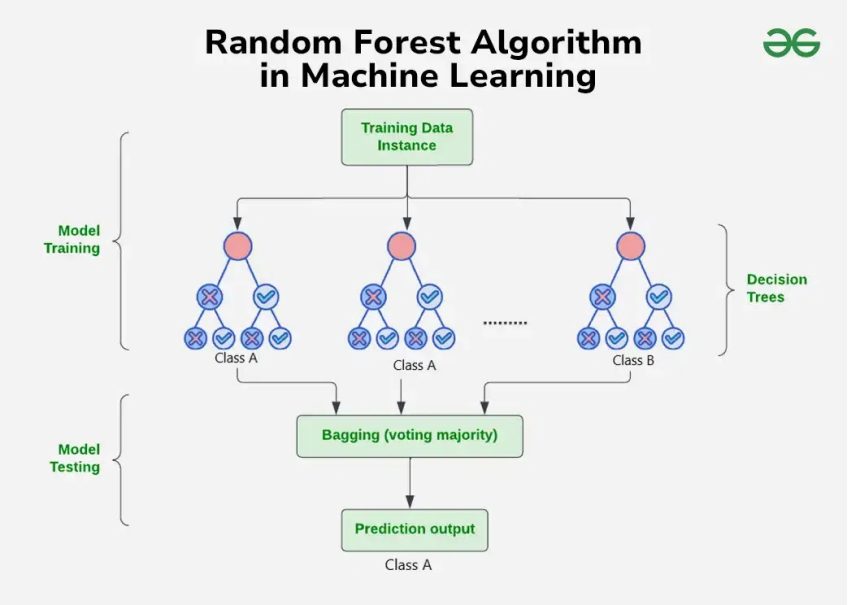

Figura 2.4: Rappresentazione grafica dell'algoritmo. Immagine presa da [13]

Anche per questo modello, utilizziamo la classe *RandomForestClassifier* dalla libreria Scikit-Learn. I parametri in questo caso sono *n_estimators*, che indica il numero di alberi che verranno creati, *max_features*, il massimo numero di *feature* da considerare quando si divide un nodo, *min_samples_split* che indica il numero minimo di campioni per dividere un nodo interno e *min_samplesleaf*, che denota il minimo numero di campioni che una foglia deve avere.

2.4 Gradient Boosting

Per spiegare come funziona l'algoritmo di *Gradient Boosting* è necessario introdurre prima quello di *Boosting*: quest'ultimo genera una prima ipotesi h_1 sul *training set* e va poi a dare un peso minore agli esempi classificati correttamente e uno maggiore a quelli incorretti. Qui si va a creare una seconda ipotesi h_2 che darà più importanza agli esempi classificati incorrettamente e così via fino a un numero K di ipotesi.
Alla fine anche le ipotesi ricevono un peso, in base alla performance ottenuta sul loro specifico *training set*, e il risultato finale non è altro che una media pesata delle predizioni di ogni ipotesi.

Quando si parla di *Gradient Boosting*, si orienta la nuova ipotesi, in questo caso un *decision tree*, verso la direzione che minimizza una certa funzione di *loss*, basandosi sull'andamento del gradiente tra la risposta corretta e quella data dall'ipotesi precedente. In pratica cerca gli errori dell'albero precedente e prova a minimizzarli in quello succesivo.

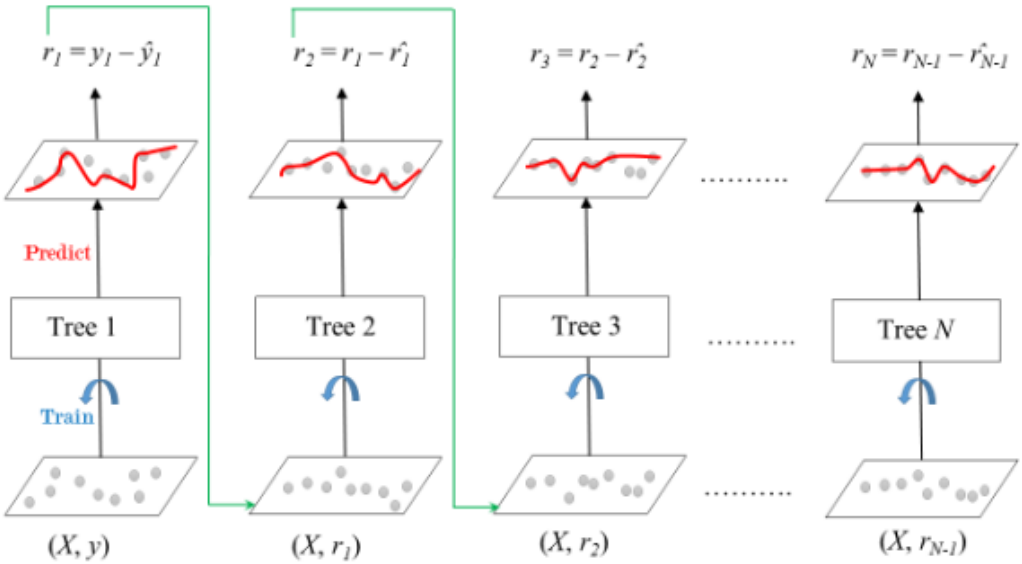

Figura 2.5: Rappresentazione del funzionamento dell'algoritmo. Immagine presa da [14]

Per utilizzare questo algoritmo adoperiamo la classe fornita dalla libreria XGBoost, che necessita il perfezionamento dei seguenti parametri: *eta* rappresenta di quanto vengono aggiornati i pesi dati alle *feature* a ogni nuovo albero, per prevenire l'*overfitting*; γ indica la minima riduzione di *loss* per dividere un nodo e *min_child_weight*, ovvero il valore minimo che deve avere un nodo affinchè possa essere un nodo figlio, altrimenti non viene diviso.

Capitolo 3

Dataset utilizzato

Al fine di ottenere un buon risultato, è necessario avere un dataset di partite quanto più ampio possibile, in modo che i modelli abbiano quanto più materiale su cui allenarsi. Nel web è possibile trovare gratuitamente i dati relativi a tutte le partite di un campionato, in questo caso della Serie A; verranno presi in considerazione gli ultimi 10 anni del campionato italiano, dalla stagione 2013/2014 alla 2023/2024. I dati, sottoforma di file csv, sono presi da [15] e contengono le informazioni riportate nella seguente tabella:

Sigla variabile	**Descrizione**
Date	Data del match
HomeTeam	Nome squadra di casa
AwayTeam	Nome squadra fuori casa
FTHG	Goal della squadra di casa
FTAG	Goal della squadra fuori casa
FTR	Indica l'esito finale (H, D, A)
HS	Tiri della squadra di casa
AS	Tiri della squadra fuori casa
HST	Tiri in porta della squadra di casa
AST	Tiri in porta della squadra fuori casa
HC	Calci d'angolo della squadra di casa
AC	Calci d'angolo della squadra fuori casa

Tabella 3.1: Dati utilizzati per la creazione delle *feature*

Ci sono poi a disposizione altre statistiche che non sono riportate perchè non utilizzate, quali le quote date da diverse case di betting al match in questione. Dai dati in tabella, che verrano indicati con le sigle presenti, si procede a creare *feature* più o meno elaborate, indicate col nome completo, che verranno poi selezionate o scartate in base al grado di correlazione tra queste e il risultato finale. Nelle sezioni successive tutte le statistiche indicate come medie non fanno altro che prendere un singolo anno e andare a sommare tutti i valori relativi a quella statistica nelle giornate precedenti e dividerle per il numero della giornata. Ad esempio, il numero di goal della Juventus all'ottava giornata è

$$\frac{\sum_{n=1}^{8} Goals_{Juventus}^{giornata_n}}{8}$$

Essendo dunque dati annuali, nella creazione del *training set* sono state rimosse le prime 5 giornate per ogni stagione, in quanto avrebbero rischiato di deviare il vero andamento di una squadra, che si evidenzia dopo diverse partite.
Nonostante la rimozione di queste 5 giornate, che per 10 partite a giornata sono 50 partite, il dataset utilizzato conta un numero di 3641 record, sufficiente al nostro scopo.
Vengono ora divise quelle scelte tra semplici e più complesse.

3.1 Feature semplici

Partendo dai risultati delle partite, si calcola quanti punti hanno ottenuto entrambe le squadre fino a quel punto (3 vittoria, 1 pareggio, 0 sconfitta), e si normalizza il punteggio sulla zbase dei punti massimi ottenibili a quella giornata, ad esempio alla quinta giornata si dividono i punti per 15, in modo che durante tutto il campionato i punti assumano un valore tra 0 e 1.
La prima *feature* è dunque

$$PointsDiff = HomePoints - AwayPoints$$

positiva se la squadra di casa ha più punti, negativa viceversa. Le successive 9 sono molto simili e rappresentano la differenza su statistiche medie delle partite.
Corner rappresenta la differenza sui calci d'angolo medi a partita di entrambe le squadre

$$Corner = HomeCorner - AwayCorner$$

AverageGoal è la differenza tra i goal medi a partita della squadra di casa e di quella ospite

$$AvgGoals = HomeAvgGoals - AwayAvgGoals$$

mentre *GoalDiff* è la differenza della differenza reti delle squadre

$$GoalDiff = (HomeGoals - HomeGoalReceived) - (AwayGoals - AwayGoalsReceived)$$

Le prossime 4 indicano differenze sulle statistiche relative ai tiri:
differenza dei tiri medi a partita effettuati dalle due squadre

$$Shots = HomeShots - AwayShots$$

differenza dei tiri in porta medi a partita effettuati dalle due squadre

$$ShotsOnTarget = HomeSOT - AwaySOT$$

differenza dei tiri medi a partita subiti dalle due squadre

$$ShotReceived = HomeSR - AwaySR$$

differenza dei tiri in porta medi a partita subiti dalle due squadre

$$ShotsOnTargetReceived = HomeSOTR - AwaySOTR$$

Infine vi è la differenza tra i tiri che fa la squadra di casa e quelli che riceve quella ospite e viceversa

$$HomeShotsDifference = HomeShots - AwayShotsR$$

$$AwayShotsDifference = AwayShots - HomeShotsR$$

3.2 Feature complesse

Gli attributi successivi sono più elaborati dei precedenti e vedremo in seguito se risulteranno più efficienti. Partiamo da *HomeForm* e *AwayForm*, che stanno ad indicare la forma della squadra: in pratica dopo ogni giornata una squadra aumenta questo valore se vince e lo diminuisce se perde, di un ammontare proporzionale alla forma dell'altra squadra, cioè una squadra con valore più basso guadagna molto se vince contro una con valore alto, e quest'ultima perde lo stesso ammontare. La frazione di punteggio che va aggiunto a una e tolto all'altra è $\frac{1}{3}$, ovvero 0.33.
In caso di pareggio, è comunque la squadra con valore minore a "rubare" punti all'altra squadra. Questa è l'unica *feature* che deriva da altri studi, anche perchè sono pochi quelli forniscono le formule per gli attributi da loro creati. Lo studio in questione è [16]. Da queste due, si ricava la loro differenza

$$FormDiff = HomeForm - AwayForm$$

C'è poi *HomeInfluence*, che tiene conto del variare della performance delle squadre tra le partite che giocano in casa e quelle fuori casa: in pratica prende il totale dei punti che ha la squadra e con questi divide il totale dei punti fatti nello stadio di casa. Questo è l'*HomeFactor* e il suo opposto è l'*AwayFactor*, banalmente $1 - HomeFactor$. Dunque

$$HomeInfluence = HomeFactor_{HomeTeam} - AwayFactor_{AwayTeam}$$

La *feature Realization* indica la percentuale di goal fatti mediamente a partita da una squadra sul totale di tiri fatti in media

$$Realization = \frac{HomeGoals}{HomeShots} - \frac{AwayGoals}{AwayShots}$$

Con *GoalkeeperEfficiency* si intende la percentuale di tiri in porta subiti *SOTR* da una squadra che non si sono trasformati in goal *GR*, la maggior parte delle volte appunto grazie al portiere.

$$GKEfficiency = (1 - \frac{HomeGR}{HomeSOTR}) - (1 - \frac{AwayGR}{AwaySOTR})$$

Vi è infine *Chance* che è una combinazione delle due precedenti e come altre *feature* che rappresentano entrambe le squadre in un unico valore, un numero positivo indica la squadra di casa come favorita, viceversa il contrario

$$Chance = (HomeR + HomeGKE) - (AwayR + AwayGKE)$$

3.3 Selezione feature

Il prossimo passo è quello di selezionare quali usare. In realtà solo le *SVMs* hanno bisogno di questo processo, in quanto *Random Forests* e *Gradient Boosting* selezionano autonomamente gli alberi con quelle più rilevanti; andremo dunque a utilizzare la funzione *get_score()* della libreria *XGBoost*, che ritorna un valore tanto più alto quanto nella creazione della foresta una certa *feature* è risultata importante. Si può scegliere in base a quale criterio dare importanza: in questo caso utiliziamo il peso dato a ognuna, ovvero quante volte è stata usata per creare uno split nei nodi degli alberi; infatti l'algoritmo considera queste funzioni discriminanti al fine di separare le osservazioni nelle diverse classi.

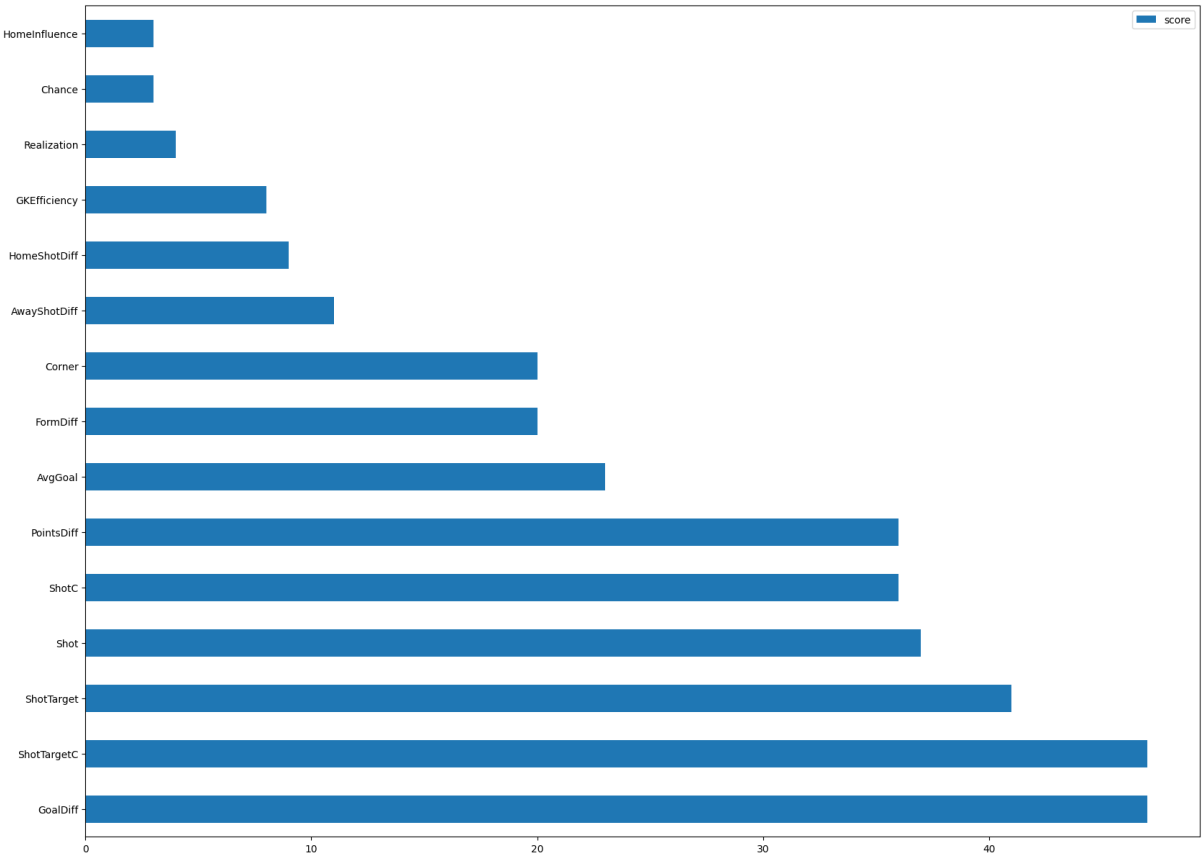

Figura 3.1: Importanza delle varie feature

Nella 3.1 viene riportata l'importanza dei vari attributi. Solitamente per questo processo si utilizzano determinati coefficienti di correlazione che sfruttano la varianza dei dati per determinare una dipendenza con il risultato da predirre; di questi ne esistono diversi, da applicare alla giusta composizione dei dati. Nel caso specifico di questa tesi, con attributi numerici e valore da predirre categorico e non binario, e soprattutto con relazione tra dati e label non lineare, non si è riusciti a trovare un coefficiente in grado di stimare al meglio l'importanza da attribuire, pertanto si è optato per la selezione degli alberi.

Da varie prove, gli attributi che portano il modello SVM a una maggior precisione sono i 9 più influenti del grafico 3.1, ovvero

Sigla variabile	**Descrizione**	**Peso**
GoalDiff	Differenza della differenza reti	47
ShotTargetC	Differenza dei tiri medi in porta subiti	47
ShotTarget	Differenza dei tiri medi in porta	41
Shot	Differenza dei tiri medi	37
ShotC	Differenza dei tiri medi subiti	36
PointDiff	Differenza dei punti in classifica	36
AvgGoal	Differenza dei goal medi	23
FormDiff	Differenza della forma	20
Corner	Differenza dei calci d'angolo medi	20

Tabella 3.2: Dati selezionati per la SVM

Capitolo 4

Esperimenti

In questo capitolo vengono riportati i risultati ottenuti utilizzando gli algoritmi descritti nel Capitolo 2 e i dati dal Capitolo 3.
Il primo passo sta nel cercare i parametri migliori da utilizzare nelle classi fornite dalle librerie.

4.1 Ottimizzazione iperparametri

Questo passaggio risulta molto semplice grazie alla classe *GridSearchCV* fornita dalla libreria *Scikit-Learn*; è sufficiente definire una griglia con all'interno i vari valori che ogni parametro deve assumere e la classe andrà ad allenare un modello per ogni combinazione. Al variare del numero di valori per ogni parametro che si decide di testare, del numero dei parametri e della grandezza del dataset, questo processo risulta più o meno lungo o necessità di più o meno potenza di calcolo. In questo caso, per un variare di 3/4 valori per 3/4 parametri su un dataset di 3641 record, non è stato necessario l'ausilio di macchine più potenti di un normale computer portatile.

La classe valuta la combinazione di valori migliori utilizzando la *K-fold Cross Validation*.

4.1.1 K-fold Cross validation

La *K-fold cross validation* è una delle tecniche di validazione più usata in ML. Questa serve a ridurre il rischio di *overfitting*, il problema derivante dal modello troppo impostato sul *training set* che non è in grado di generalizzare su dati mai visti.
Per fare questo, il *training set* viene diviso in diverse porzioni a seconda del valore di K, e a

turno viene allenato su tutte queste meno una, la quale verrà utilizzata come validazione per testarne la precisione. Al turno seguente, la validazione è effettuata su un'altra porzione e l'allenamento sulle restanti *K-1* e così via. Alla fine il valore di precisione o di qualsiasi altro metro di valutazione, vale la media dei singoli punteggi. Generalmente K assume valore 5 o 10, nel nostro caso avrà valore 5.

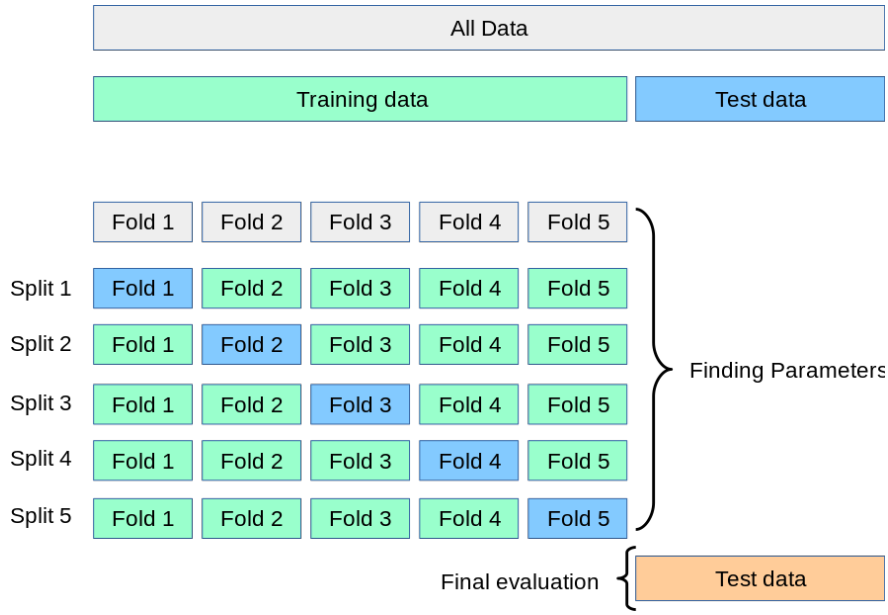

Figura 4.1: Cross validation per parameter tuning. Immagine presa da [17]

La classe *GridSearchCV* valuta ad una ad una le combinazione dei parametri inseriti nella griglia e per decretare quale sia il migliore utilizza appunto la *cross validation*.

Viene dunque eseguita la procedura sopra indicata su ognuno dei 3 algoritmi che utilizzeremo. Il grafico seguente riporta solo i valori per C e γ, che sono i più importanti.

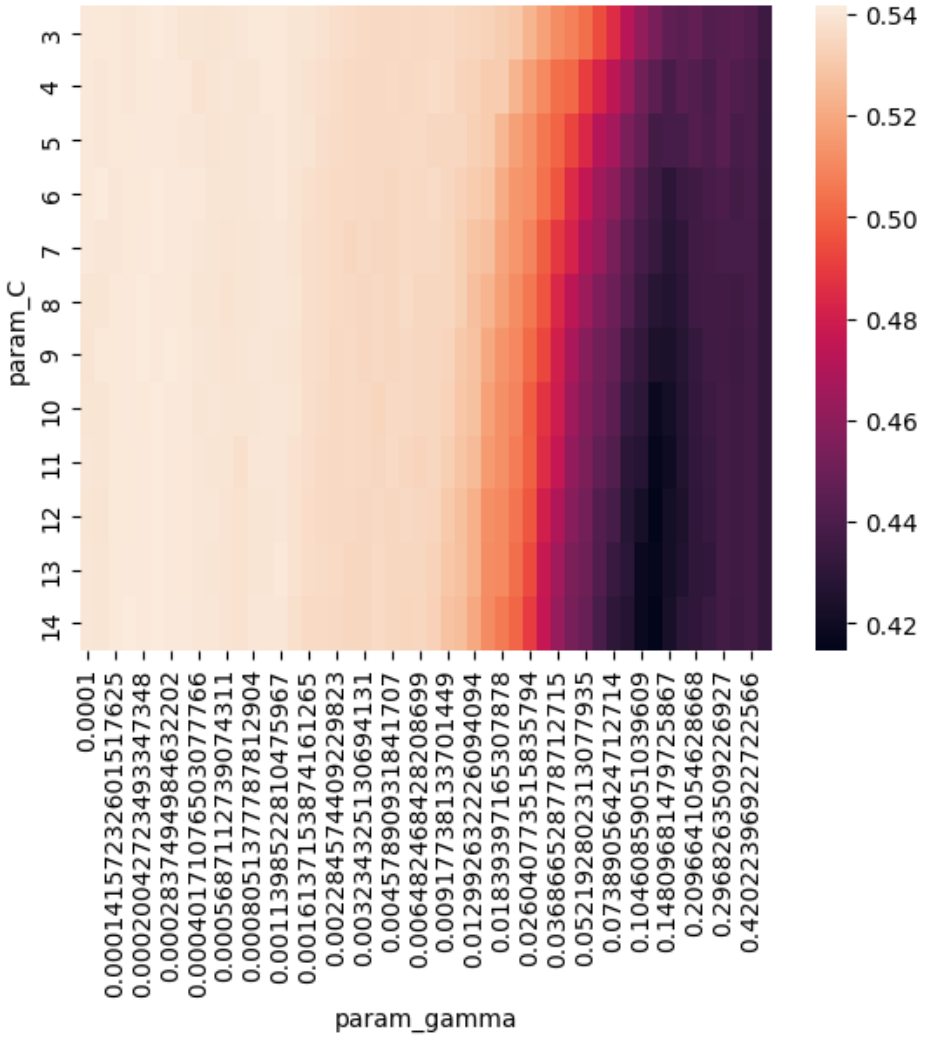

Figura 4.2: Punteggio su valori di γ e C

Per la *SVM* si ottengono i seguenti parametri:

Parametro	Valore
C	9
γ	0.000283749498

Tabella 4.1: Valori selezionati per *SVM*

Il grafico successivo mostra i valori ottenuti per *max_features* e *n_estimators*

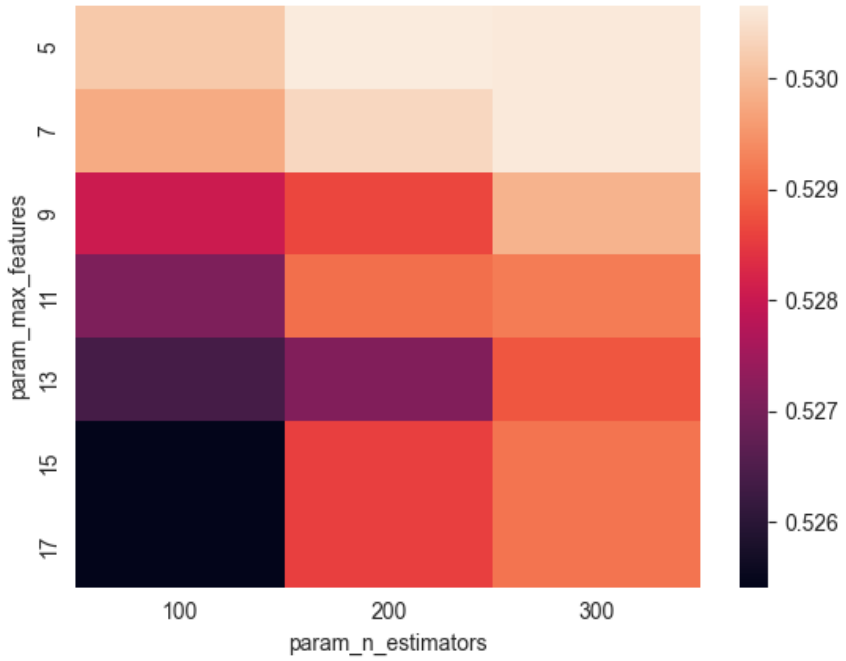

Figura 4.3: Punteggio su valori di *max features* e *n estimators*

I valori ottenuti per *Random Forest* sono dunque:

Parametro	Valore
max_features	11
n_estimators	300
max_depth	3
min_samples_leaf	1
min_samples_split	2

Tabella 4.2: Valori selezionati per *Random Forest*

E infine per *Gradient Boosting* è evidenziato l'andamento per γ e *eta*

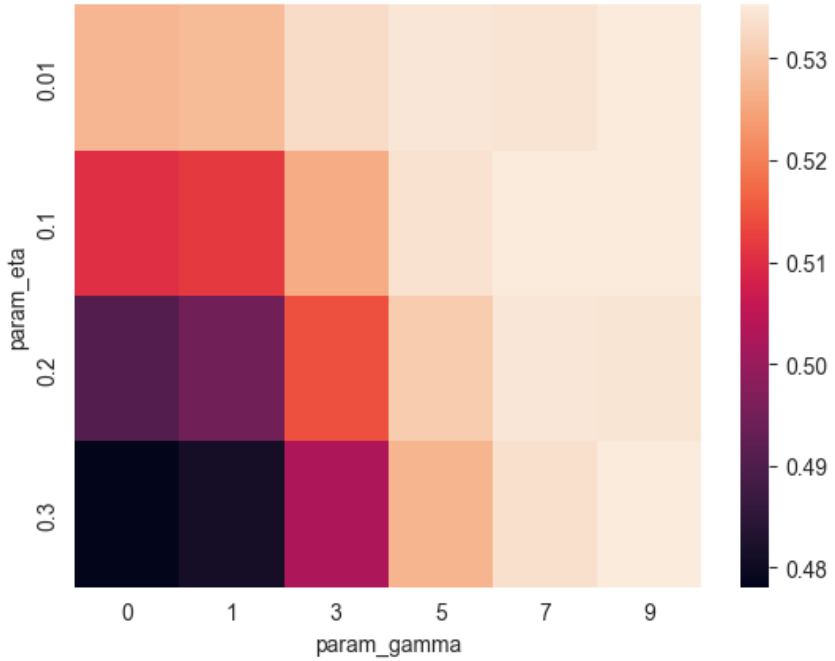

Figura 4.4: Punteggio su valori di γ e *eta*

I valori selezionati sono dunque:

Parametro	Valore
eta	0.3
γ	9
subsample	0.5
max_depth	3
min_child_weight	0

Tabella 4.3: Valori selezionati per *Gradient Boosting*

4.2 Training e testing

Ottenuti questi parametri si procede a allenare i modelli per vederne precisione e differenze di performance sui vari esiti disponibili.

Il modello migliore, ma non per distacco, risulta essere *Gradient Boosting* che ottiene una precisione del 54.32%. Di seguito viene riportata la *confusion matrix*, grafico molto utile per comprendere quali classi ha predetto in maniera corretta e quali no, mostrando nella diagonale principale quelle corrette e sulle altre caselle i valori con cui le ha scambiate.

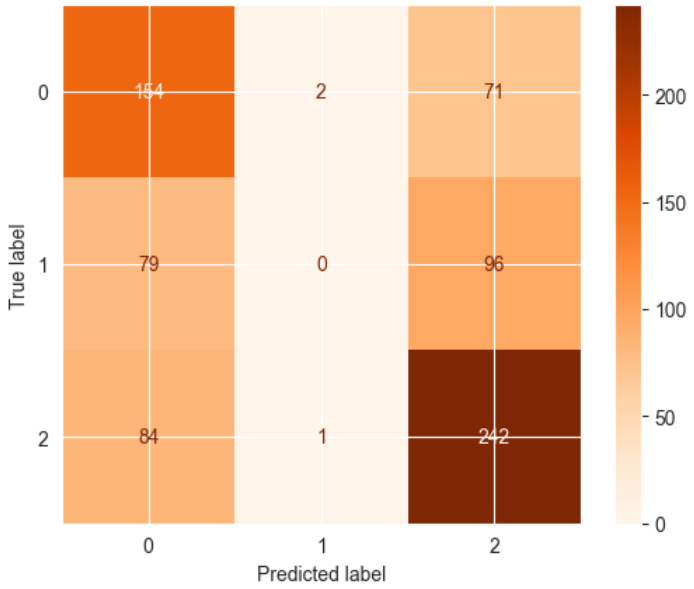

Figura 4.5: *Confusion matrix* del modello *Gradient Boosting*

Il modello *Random Forest* ottiene invece una precisione del 54.18%, col seguente punteggio sui singoli esiti.

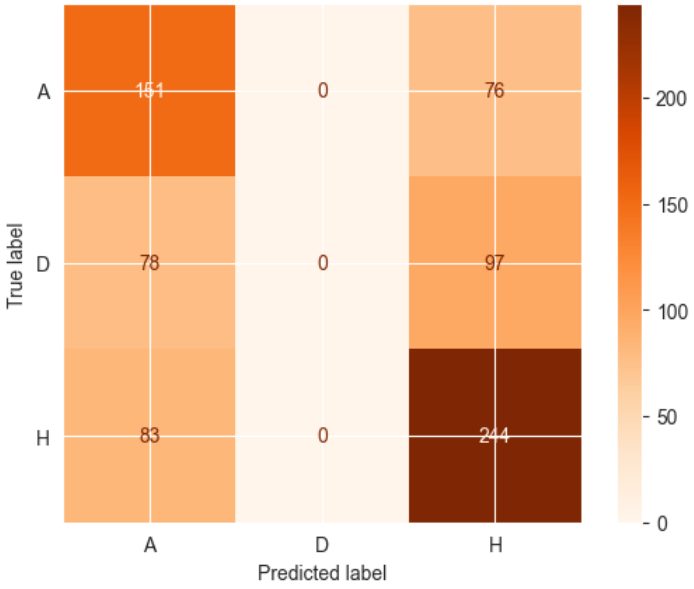

Figura 4.6: *Confusion matrix* del modello *Random Forest*

Infine *SVM* arriva al 54.16% di precisione.

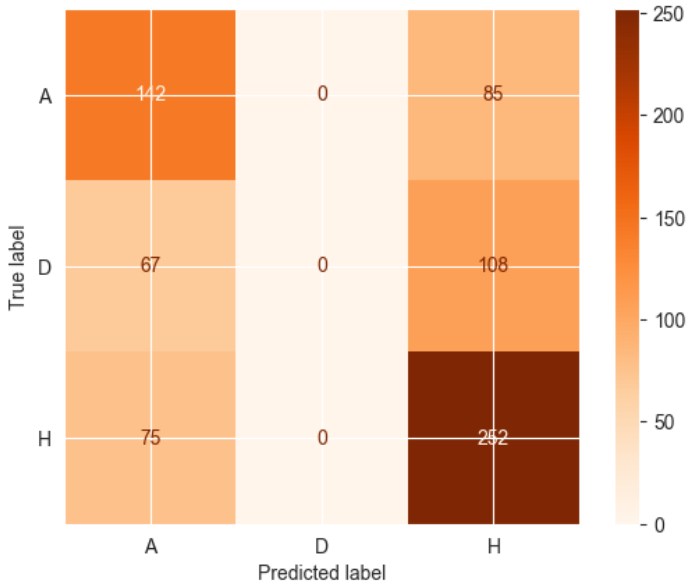

Figura 4.7: *Confusion matrix* del modello *SVM*

Vengono qui di seguito esposte altre metriche per valutare la performance di un modello, quali *recall* e *F1 score*; mentre la precisione indica il numero di predizioni giuste sul numero di volte che una classe è stata predetta

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

il *recall* indica la percentuale di esiti corretti riguardanti una classe

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

e il *F1 score* rappresenta un compromesso tra le due metriche precedenti

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

I punteggi ottenuti dai vari modelli sono:

Modello	Precision	Recall	F1 score
SVM	54.16	69.8	60.42
Random Forest	54.18	70.56	60.8
Gradient Boosting	54.32	70.92	61.19

Tabella 4.4: Valori delle metriche sui vari modelli

Salta subito all'occhio che mentre l'esito *H*, cioè la vittoria della squadra di casa, risulta essere quello che viene predetto meglio, seguito dalla vittoria fuori casa, il pareggio non solo non conta predizioni corrette in nessuno dei modelli, ma viene azzardato erroneamente solo 3 volte nel *Gradient Boosting*.
Dai risultati di studi simili era risaputo che il pareggio fosse l'esito più difficile da prevedere, ma in questo caso sembra quasi non essere preso in considerazione. Proviamo ora a riallenare i modelli tenendo come parametri in ognuno quelli di default: mentre il *SVM* continua a non predirre nessun pareggio (comportamento ritrovato anche in altri studi), *Random Forest* e *Gradient Boosting* si comportano in modo differente, come visibile nelle successive matrici.

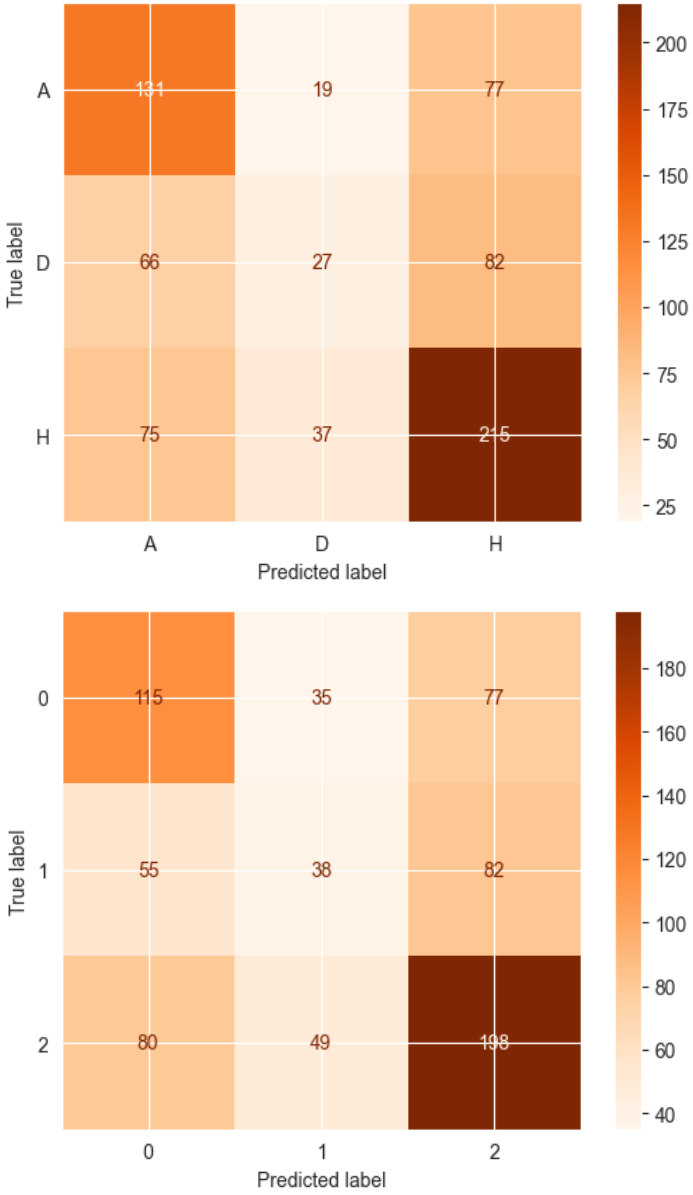

Figura 4.8: *Confusion matrix* di *Random Forest* (sopra) e *Gradient Boosting* (sotto)

Si nota che con i valori predefiniti questi due modelli sono in grado di prevedere

alcuni pareggi; tuttavia non raggiunge una buona percentuale di precisione su tale esito, tanto che sono più quelli predetti erroneamente che quelli corretti, questo a conferma della difficoltà della sua previsione. Sembra dunque che nel processo di *tuning* dei parametri i valori che tendono ad evitare il pareggio in quanto esito meno frequente e più difficilmente pronosticabile, riescano ad ottenere una precisione maggiore sui singoli esiti rimanenti, infatti la precisione totale dei due modelli in grado di prevedere la parità è scesa rispettivamente a 51.16% e 48.14%.

Vengono ora mostrati i risultati di un'analisi eseguita tramite la libreria *Shap (SHapley Additive exPlanations)* sull'importanza delle *feature* usate per l'addestramento del modello *Gradient Boosting*, per poi fare un confronto con l'importanza attribuita dal modello stesso. *Shap* analizza una vasta quantità di combinazioni di *feature* e calcola la differenze nelle predizione quando un determinato attributo è escluso o incluso. A questi assegna poi un valore di Shapley facendo una media ponderata di tutte queste differenze, pesate in base al numero di combinazioni in cui la determinata *feature* è presente.

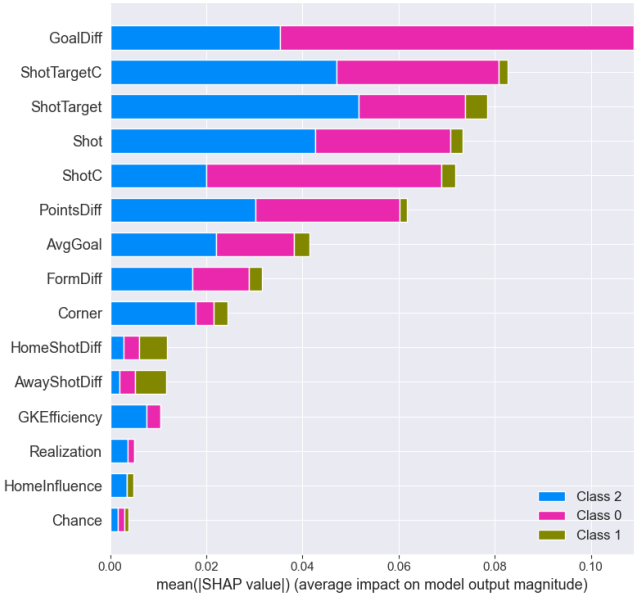

Figura 4.9: Grafico dell'importanza delle *feature* secondo l'analisi di *Shap*

Nella Figura 4.9 vengono indicate le singole influenze di ogni attributo sui vari esiti finali (0: vittoria fuori casa, 1: pareggio, 2: vittoria casa). Questo potrebbe essere interessante per analizzare quali di queste sono più rilevanti sulla predizione dei pareggi col fine di creare un modello in grado di migliorare su tale esito.
I risultati confermano la scelta degli attributi selezionati per l'addestramento del modello *SVM* riportati in Figura 3.1.

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi si è cercato di predirre al meglio l'esito di una partita di calcio partendo da statistiche di incontri precedenti della stessa stagione.
Le performance ottenute risultano nettamente inferiori rispetto agli studi citati nella Sezione 1.1, in particolare quello di Taspinar e colleghi [6] che sullo stesso campionato hanno ottenuto un punteggio decisamente migliore. Alla base di questo fenomeno, come detto più volte nel corso del documento, c'è la disponibilità e combinazione di dati più accurati e specifici, in grado di descrivere al meglio l'andamento di un match al fine di poter capire le dinamiche di partite non ancora disputate o mai viste dal modello. Reperire questi dati può essere un primo passo per migliorare le performance dei vari modelli. In aggiunta, innestare dati di natura diversa, ad esempio le valutazioni dei giocatori o della squadra nel complesso con punteggi assegnati dall'uomo o da computer, potrebbe incidere notevolmente.

Questa tesi si ferma dunque a una precisione massima di 54.32%, che nei 3 esiti totali rimane comunque superiore al semplice tirare a sorte, riuscendo a prevedere correttamente poco più di una partita su due.

Bibliografia

[1] Jong gon Shin e Robert Gasparyan. «A novel way to Soccer Match Prediction». In: 2014. URL: https://api.semanticscholar.org/CorpusID:15564057.

[2] URL: https://www.ea.com/it-it/games/fifa.

[3] Darwin Prasetio e Dra. Harlili. «Predicting football match results with logistic regression». In: *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*. 2016. DOI: 10.1109/ICAICTA.2016.7803111.

[4] Jeffrey Snyder. «What Actually Wins Soccer Matches: Prediction of the 2011-2012 Premier League for Fun and Profit». In: 2013. URL: http://arks.princeton.edu/ark:/88435/dsp012z10wq33p.

[5] Hengzhi Chen. «Neural Network Algorithm in Predicting Football Match Outcome Based on Player Ability Index». In: *Advances in Physical Education* (2019). DOI: 10.4236/ape.2019.94015.

[6] Yavuz Taspinar, Ilkay Cinar e Murat Koklu. «Improvement of Football Match Score Prediction by Selecting Effective Features for Italy Serie A League». In: *MANAS Journal of Engineering* 9 (apr. 2021). DOI: 10.51354/mjen.802818.

[7] «Home Advantage in Football: How Often Does the Home Team Win?» In: (). URL: https://www.bettingoffers.org.uk/how-big-is-home-advantage-in-football/.

[8] Shai Shalev-Shwartz e Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014, pp. vii–viii. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning.

[9] *Come funziona il machine learning?* URL: https://aws.amazon.com/it/what-is/machine-learning/.

[10] Juliano Morimoto. «Can Quantum-Mechanical Description of Physical Reality be Considered Complete?» In: (2021). DOI: 10.1186/s12052-021-00147-x.

[11] *Understanding decision trees*. URL: https://medium.com/@jainvidip/understanding-decision-trees-1ba0ef5f6bb4.

[12] Stuart Russell e Peter Norvig. *Artificial Intelligence: A modern approach*. Pearson Education, 2021, pp. vii–viii. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning.

[13] *Algoritmo Random Forest nell'apprendimento automatico*. URL: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/.

[14] *Gradient Boosting in ML*. URL: https://www.geeksforgeeks.org/ml-gradient-boosting/.

[15] URL: https://www.football-data.co.uk/italym.php.

[16] Rahul Baboota e Harleen Kaur. «Predictive analysis and modelling football results using machine learning approach for English Premier League». In: *International Journal of Forecasting* 35.2 (2019), pp. 741–755. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2018.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0169207018300116.

[17] URL: https://scikit-learn.org/stable/modules/cross_validation.html.

Ringraziamenti

Desidero esprimere la mia gratitudine al professor Chiariotti Federico, mio relatore, per il sostegno e la pazienza che mi ha dedicato durante la stesura dell'elaborato.

Vorrei soprattutto ringraziare mamma Laila, la roccia a cui posso sempre aggrapparmi, e tutta la mia famiglia.

Un ringraziamento anche a tutta la Cacao e affiliati, per la spensieratezza e i bei momenti che ormai durano da anni, e i miei colleghi per aver preso parte e reso migliore questa esperienza.

Infine un grazie speciale a Paola, che ha creduto in me forse più di quanto non lo abbia fatto io e in questi 3 anni è stata la luce dei miei occhi.

Padova, Settembre 2024