UNIVERSITÀ
DI TORINO

Università degli Studi di Torino

Corso di Laurea Triennale in Innovazione Sociale, Comunicazione e Nuove tecnologie

Sviluppo, sfide e limitazioni di un modello predittivo nel ciclismo professionistico: il caso del Tour de France 2023

Relatore

Console Luca

Candidato

Peiretti Riccardo

Matricola 949104

Anno Accademico 2022/2023

Sommario

Lista abbreviazioni

Acronimo	Spiegazione
AI	Artificial intelligence / intelligenza artificiale
API	Application Programming Interface
BS4	Beautiful Soup 4
DF	DataFrame
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ITT	Individual Time Trial / Cronometro individuale
KG	Chilogrammi
KM	Chilometri
ML	Machine learning / Apprendimento automatico
M	Metri
PCS	ProCyclingStats
PS	Profile Score
UCI	Unione Ciclistica Internazionale
VS Code	Visual Studio Code
XGBoost	eXtreme Gradient Boosting
XML	eXtensible Markup Language

Introduzione

Quando tecnologia e passione per lo sport si uniscono, nascono sfide straordinarie: è in questo incrocio che si sviluppa il cuore di questa tesi. Al giorno d'oggi, le nuove tecnologie e i sistemi di intelligenza artificiale (AI) hanno rivoluzionato il nostro modo di vivere, influenzando le scelte quotidiane in settori differenti, compreso quello dello sport.

Nell'ambito specifico del ciclismo, in tempi recenti, i modelli predittivi basati sul machine learning (ML) sono stati utilizzati, ad esempio, per prevedere e analizzare le prestazioni dei corridori, per ottimizzare gli allenamenti degli atleti[1], per le strategie tattiche da utilizzare nel corso di una gara, ma anche per calcolare il fabbisogno calorico che un ciclista deve assumere durante una gara[2].

Questa tesi propone lo sviluppo di un modello predittivo di machine learning supervisionato in grado di predire i risultati dei corridori in ogni tappa di un Grande Giro.

Nel primo capitolo, si percorre in un breve excursus il percorso storico che ha condotto all'odierna società influenzata dal potere computazionale. Inoltre, vengono approfondite le origini dell'intelligenza artificiale e del machine learning: si presenta una panoramica generale del ML focalizzandosi sugli elementi fondamentali.

Nel secondo capitolo, si introduce il caso studio della ricerca: il Tour de France 2023. L'attenzione si concentra sul processo di creazione del modello predittivo, partendo dal data gathering che comprende lo scraping dei dati; la data pre-processing, ossia l'analisi dei dati e la pulizia dei dati. Inoltre, la creazione finale del dataset con l'individuazione delle features e delle variabili target. Infine, vengono esposti i concetti teorici del modello dalla costruzione all'utilizzo.

Nel terzo capitolo, si analizzano i risultati ottenuti dall'applicazione del modello di machine learning creato al caso del Tour de France 2023.

Nel quarto capitolo, si approfondiscono i limiti del modello e i possibili miglioramenti. Si analizzano le sfide e gli impedimenti durante l'implementazione e l'applicazione del modello. In particolare, i fattori limitanti, come la disponibilità e la qualità dei dati. Inoltre, si esaminano le potenziali strategie per superare queste limitazioni.

Infine, nell'ultimo capitolo, si presentano le conclusioni generali e si evidenziano i vantaggi derivanti dall'applicazione del machine learning. Il modello predittivo sviluppato, pur avendo delle limitazioni, prova a fornire un approccio innovativo per supportare le decisioni nel campo sportivo, aprendo nuove opportunità per l'analisi delle prestazioni e le strategie di gara.

[1] Hilmkil A., Ivarsson O., Johansson M., Kuylenstierna D., van Erp T. (2018), Towards Machine Learning on data from Professional Cyclists (https://doi.org/10.48550/arXiv.1808.00198)
[2] Van Kuijk K., Dirksen M., Seiler C. (2023), Conformal Regression in Calorie Prediction for Team Jumbo-Visma (https://doi.org/10.48550/arXiv.2304.03778)

1. L’era dei dati

Come anticipato nell’Introduzione, in questo primo capitolo, viene presentato il cambiamento sociale che ha portato al concetto di società dei dati e viene introdotta una visione generale dell’intelligenza artificiale e del machine learning.

1.1. Contesto storico [3]

Dagli anni ’90 fino alla metà degli anni 2000, vi è stato un interesse multidisciplinare estremamente forte sul concetto di informazione. Infatti, l’informazione è stata considerata come il mattone che ha creato la realtà e come chiave per interpretare e comprendere la società. Da alcuni anni, però, c’è stato uno spostamento lessicale: la centralità dell’informazione è stata sostituita dalla centralità del dato. In realtà, i dati vengono ancora prima dell’informazione, per cui viene spontanea porsi la domanda del perché si sia tornati indietro al concetto di dato. Questo passaggio è avvenuto per tre fattori.

In primis, è necessario differenziare il concetto di dato e di informazione. Da una parte, un dato è inteso come una differenza che segnala una differenza, ad esempio buio-luce, rumore-silenzio, caldo-freddo. Quindi, i dati sono fondamentali per descrivere e conoscere la realtà. Dall’altra, l’informazione è un dato ben costruito dal punto di vista sintattico cui noi abbiamo attribuito un certo significato. In conclusione, i dati possono essere elaborati e processati a livello sintattico da motori sintattici. Invece, le informazioni sono elaborate e processate a livello semantico da motori semantici.

In secondo luogo, si differenziano i motori semantici e sintattici. I motori semantici, come gli umani, attribuiscono al dato un certo significato. I motori sintattici, come le macchine e i robot, stabiliscono come un input si trasforma in un output sulla base di regole, senza bisogno di un’attribuzione di significato. Le macchine possono elaborare e processare i dati in chiave sintattica: quanti più dati processano, tanto più affinano la loro capacità di produrre risultati. Occorre quindi incrementare la produzione e circolazione dei dati, in modo tale da poter addestrare i sistemi di intelligenza artificiale e machine learning a svolgere compiti in modo sempre più rapido ed efficiente.

Il terzo fattore è l’emergere del potere computazionale, ossia la capacità di trasformare gli input in output. È una capacità che hanno sia gli umani che le macchine, ma svolta in modo diverso. Il potere computazionale riguarda principalmente i dati, e non l’informazione. Quando erano gli umani ad occuparsi di questo, ovviamente ci si basava solamente sull’informazione. Invece, oggi, visto che il compito è affidato alle macchine, si è passati a quello che viene prima dell’informazione, ossia i dati. Una società di dati è soprattutto una società di macchine che processano dati per i fini più diversi.

Dopo aver tracciato il percorso storico che ha portato all’odierna società dei dati, si cerca di offrire una breve spiegazione di “intelligenza artificiale” (artificial intelligence) e “apprendimento

[3] Durante M. (2022), Potere computazionale: dalle informazioni ai dati. In Mimesis (Ed.), La politica dei dati. Il governo delle nuove tecnologie tra diritto, economia e società (pp. 59-63)

automatico" (machine learning), spesso utilizzati erroneamente come sinonimi nel pensiero comune.

1.2. Artificial intelligence

Per milioni di anni, abbiamo cercato di capire come pensiamo. Il campo dell'intelligenza artificiale, o AI, va ancora oltre: tenta non solo di capire, ma anche di costruire entità intelligenti. I primi studi risalgono al 1956, quando, nel New Hampshire al Dartmouth College, si tenne un convegno al quale parteciparono figure rilevanti del settore: un team di dieci persone avrebbe dovuto creare una macchina in grado di simulare ogni aspetto dell'apprendimento e dell'intelligenza umana. Fu in tale occasione che John McCarthy, informatico statunitense, coniò il termine "intelligenza artificiale"[4]. Numerosi studiosi, in passato, hanno ricercato la definizione di intelligenza artificiale, ponendosi domande differenti. Ad esempio, il test di Turing, proposto da Alan Turing (1950), matematico britannico, fu progettato per fornire una definizione di "intelligenza". Un sistema di intelligenza artificiale è tale se un umano, dopo aver posto delle domande, non è in grado di decifrare se le risposte provengano da una persona o da un computer. Questo test non si sofferma sull'interazione fisica diretta tra interrogatore e macchina, poiché non è necessario per definire il concetto di "intelligenza"[5].

In generale, l'AI è una vasta area di studio e sviluppo che mira a creare sistemi informatici in grado di esibire comportamenti e capacità che richiedono solitamente l'intelligenza umana. L'obiettivo dell'AI è quello di creare macchine che possano eseguire attività che richiedono ragionamento, apprendimento, percezione, comprensione del linguaggio naturale. Questi algoritmi spaziano dalla logica simbolica ai modelli statistici e di apprendimento automatico, inclusi i più recenti sviluppi nell'apprendimento profondo (deep learning) e nelle reti neurali artificiali.

1.3. Machine learning

Il machine learning (apprendimento automatico) è uno degli approcci tecnologici innovativi della nostra era moderna ed è considerato una branca dell'artificial intelligence. Si focalizza sull'uso di dati e algoritmi per imitare la metodologia di apprendimento degli umani, migliorando l'accuratezza dell'apprendimento con il tempo. I sistemi di apprendimento automatico si occupano di creare sistemi che apprendono e migliorano le performance in base ai dati ricevuti in input. Al giorno d'oggi, il ruolo del ML è di supportare le attività umane basandosi sulla matematica.

Di seguito, una semplificazione del funzionamento dei sistemi di machine learning[6].

Figura 1 - Elementi del machine learning

[4] https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History
[5] Norvig P., Russell S. (2010), Introduction. In Pearson College (Ed.), Artificial Intelligence: A Modern Approach 3rd Edition (pp. 1-5)
[6] https://hub.packtpub.com/5-key-reinforcement-learning-principles-explained-by-ai-expert/

Come riportato nella Figura 1, gli elementi fondamentali di un modello sono[7]:

- inputs: dati usati come variabili nel modello;
- algoritmo: correlazioni, dipendenze e patterns nei dati;
- output: la previsione effettuata dal modello.

Gli inputs del modello vengono chiamati tecnicamente features. Le features rappresentano le informazioni che descrivono gli oggetti su cui viene costruito il modello. Esse svolgono un ruolo cruciale nell'addestramento poiché sono la fonte da cui il modello apprenderà per prendere la decisione corretta. Come si vede nella sezione 2.3.1, la selezione delle features è un'importante fase di pre-elaborazione dei dati nel processo di creazione del modello. È possibile che non tutte le features disponibili siano utilizzabili e rilevanti. Esse sono selezionate ed elaborate in modo da catturare le caratteristiche salienti dei dati e consentire al modello di imparare i pattern e fare previsioni appropriate.

Gli algoritmi sono i motori che alimentano il machine learning, poiché sono responsabili di apprendere dai dati e di effettuare previsioni o prendere decisioni. I due tipi principali di algoritmi di ML sono il machine learning supervisionato e l'apprendimento non supervisionato, ognuno dei quali ha un approccio diverso nell'apprendimento dei dati[8]:

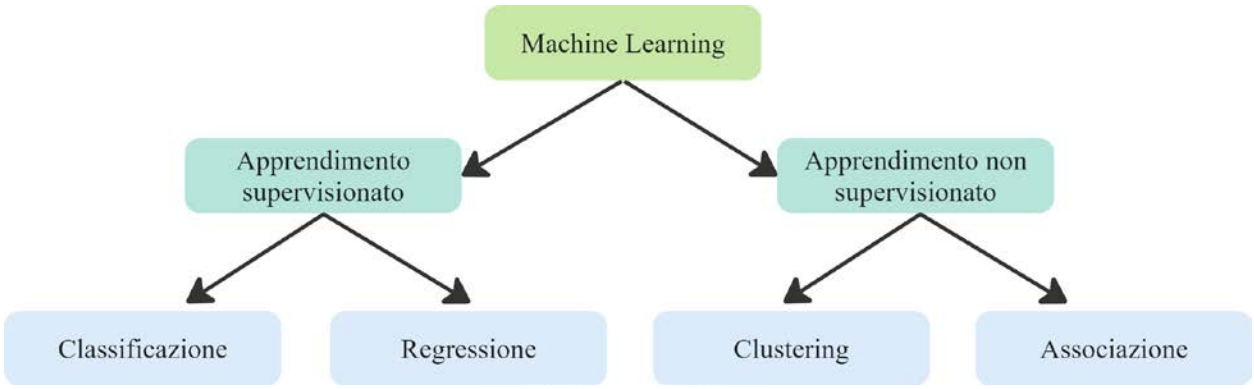

Figura 2 - Principale classificazione modelli ML

- machine learning supervisionato: questi tipi di algoritmi sono i più utilizzati. L'algoritmo apprende da un set di dati etichettati e con un output predefinito. Vi sono variabili di input (x) da cui il modello predice l'output (Y).

$$Y = f(x)$$

L'obiettivo è quello di mappare in modo approfondito la funzione in modo che, quando si ha un input (x), si predica l'output (Y) per quei dati. I modelli di ML supervisionato, inoltre, si suddividono in problemi di regressione e classificazione:

- classificazione: un problema di classificazione è quando un output è considerato una categoria, come "rosso" o "blu";

[7] https://quantdare.com/machine-learning-a-brief-breakdown/
[8] https://towardsdatascience.com/the-abc-of-machine-learning-ea85685489ef

    - regressione: un problema di regressione è quando un output è considerato un valore reale, come “stipendio” o “prezzo”;
- apprendimento non supervisionato: utilizza un approccio più indipendente dove il modello è formato da dati privi di etichette e dove non è stato definito un output specifico. Vi sono variabili di input (x), ma non esistono variabili di output corrispondenti. L’obiettivo è di modellare la struttura e la distribuzione dei dati per conoscerli in maniera più approfondita. I modelli di ML non supervisionato sono così definiti poiché gli algoritmi sono gestiti autonomamente. I problemi di apprendimento non supervisionato possono essere ulteriormente suddivisi in clustering e associazione:
    - clustering: un problema di clustering è dove si desidera scoprire dei possibili raggruppamenti nei dati che si possiedono, come il raggruppamento dei clienti attraverso il comportamento di acquisto. Un esempio sono gli algoritmi di clustering k-means;
    - associazione: un problema di associazione è dove si desidera scoprire delle regole che descrivono grandi porzioni di dati, come i clienti che acquistano X tendono anche a comprare Y. Un esempio è l’algoritmo Apriori.

L’output del modello, o target, è la variabile che rappresenta l’obiettivo che si vuole predire o stimare utilizzando il modello. È la variabile che cercherà di apprendere in base alle features fornite durante il processo di addestramento e all’algoritmo utilizzato.

Il machine learning offre una vasta gamma di modelli predittivi che possono essere utilizzati per affrontare diverse tipologie di problemi. Oggi sono moltissimi i modelli utilizzati ed è impossibile una presentazione esaustiva di essi. Non esiste un modello migliore di un altro, ma la scelta del modello dipende dal tipo di problema da affrontare, dalle caratteristiche dei dati e dagli obiettivi specifici da raggiungere.

2. Il caso studio: il Tour de France 2023

Nel capitolo 1, si introduce una visione generale dell'intelligenza artificiale e del machine learning. Nel seguente capitolo, si affronta il caso studio. L'obiettivo è di creare un modello predittivo di machine learning supervisionato in grado di predire i risultati dei corridori in ogni tappa del Tour de France 2023. In generale, il modello viene addestrato utilizzando un ampio dataset storico che copre il periodo dal 2017 al 2022, con l'eccezione dell'anno 2020 a causa della pandemia da Covid-19. Il dataset comprende dati relativi ai tre Grandi Giri: Giro d'Italia, Tour de France e Vuelta a España. Le variabili di input includono le caratteristiche specifiche della tappa e le informazioni disponibili sui corridori.

In questo capitolo, si offre una panoramica del processo di lavoro svolto, partendo dagli step fondamentali necessari per lo sviluppo di un modello: la sezione 2.1 si focalizza sul data collection; la sezione 2.2 sul data pre-processing; la sezione 2.3 approfondisce l'analisi del dataset identificando features e variabile target. Infine, rispettivamente nelle sezioni 2.4 e 2.5, vi è un approfondimento sulla costruzione e sull'utilizzo del modello.

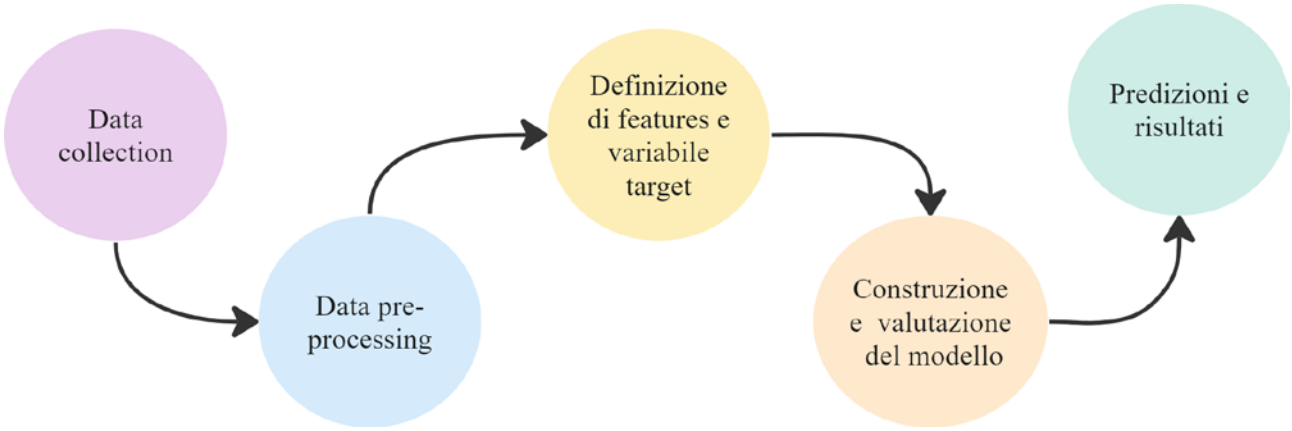

Figura 3 - Processo di lavoro

2.1. Data collection

Il primo step della ricerca è il data collection, ossia la raccolta dei dati necessari per l'analisi e l'addestramento del modello. Durante questa fase, si identificano le fonti di dati pertinenti al problema in esame. Si suddivide così il procedimento:

1. identificazione della fonte da cui raccogliere i dati;
2. raccolta dei dati: si acquisiscono i dati tramite estrazione, download o altre modalità di acquisizione.

Nel caso studio, si sono ottenuti i dati utilizzando diverse risorse e strumenti:

- il dataset principale e le meta-informazioni sugli atleti coinvolti e sulle caratteristiche dei percorsi sono stati ottenuti da fonti aperte, i cosiddetti open data, totalmente dal portale statistico ProCyclingStats[9]. Questo portale fornisce una vasta quantità di dati sul ciclismo professionistico. L'estrazione è avvenuta mediante l'applicazione di web scraping poiché non esiste un'API appropriata per l'accesso diretto ai dati;

[9] https://www.procyclingstats.com

- per lavorare con i dati e scrivere codice, è stato utilizzato Visual Studio Code come editor di codice sorgente[10]. VS Code è un editor che offre funzionalità avanzate per la scrittura e l'organizzazione del codice. In particolare, è stato scritto il codice utilizzando il linguaggio di programmazione Python[11];
- la libreria Requests è stata utilizzata per effettuare richieste HTTP[12]. La libreria Requests semplifica l'invio di richieste HTTP, consentendo di interagire con servizi web e di scaricare il loro contenuto. Questa libreria è stata utile per recuperare i dati necessari dal portale ProCyclingStats attraverso richieste HTTP;
- per l'estrazione dei dati dal markup HTML del portale PCS, viene utilizzata la libreria BS4 (Beautiful Soup 4)[13]. Questa libreria Python semplifica il processo di estrazione dei dati da pagine HTML o XML. Essa ha permesso di navigare nel documento HTML, cercare elementi specifici ed estrarre le informazioni di interesse in modo efficiente.

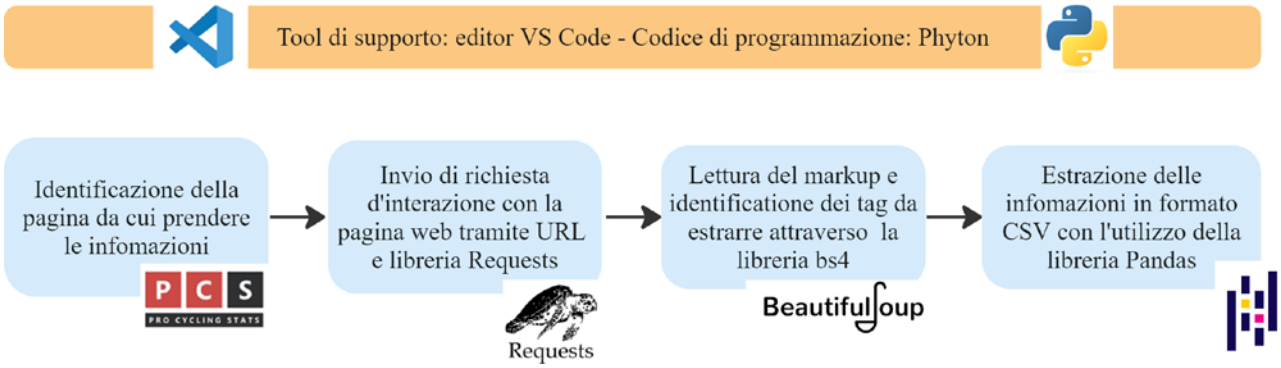

Figura 4 - Risorse e strumenti di lavoro

Questi strumenti hanno permesso di ottenere cinque dataset (2017, 2018, 2019, 2021, 2022) contenenti le informazioni passate delle tappe e dei corridori.

Si ottengono i seguenti dati in ciascun dataset:

Tabella 1 - Dati estratti

Variabile	Unità di misura	Descrizione
year	[anni]	anno della gara
grand_tour	-	nome del Grande Giro (Giro d'Italia, Tour de France e Vuelta a España)
stage_number	-	numero di tappa (1-21)
avg_speed	[km/h]	velocità media di ciascuna tappa
km	[km]	distanza in km della tappa

[10] https://code.visualstudio.com
[11] Van Rossum G., Drake Jr. F. L. (1995), Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam (https://www.python.org)
[12] Chandra R. V., Varanasi B. S. (2015), Python requests essentials. Packt Publishing Ltd (https://requests.readthedocs.io/projects/it/it/latest/#)
[13] Richardson L. (2007), Beautiful soup documentation (https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

profile_score	-	profile score (sezione 2.3.1.1)
stage_type	-	tipo di tappa (sezione 2.3.1.2)
won_how	-	modo in cui la tappa è stata vinta
leader_after_stage	-	leader della classifica generale dopo la tappa
team	-	team del ciclista
rider	-	nome del ciclista
age	[anni]	età del ciclista
nationality	-	nazionalità del ciclista
weight	[kg]	peso del ciclista
height	[m]	altezza del ciclista
position	-	posizione del ciclista nella classifica di tappa
time_tot	[HH:MM:SS]	tempo totale di percorrenza di ciascuna tappa

2.2. Data pre-processing

Il secondo step della ricerca è il data pre-processing[14]. In questa fase, i dati vengono esplorati e preparati per l'analisi e l'addestramento del modello di ML. Infatti, ci si concentra sulla trasformazione dei dati grezzi in una forma appropriata per l'elaborazione e l'utilizzo da parte del modello. Durante questa fase, possono essere eseguite diverse attività, di seguito spiegate quelle utilizzate nel caso studio.

2.2.1. Data integration

La data integration consiste nell'integrazione di differenti dati in un unico dataset. In questo step, si è creato il dataset di lavoro partendo dalle estrazioni effettuate tramite lo scraping. Grazie allo scraping, si sono ottenuti cinque files (uno per ogni anno). Questi file sono stati combinati in un unico dataset "df_storico" attraverso la funzione di Pandas "concat"[15].

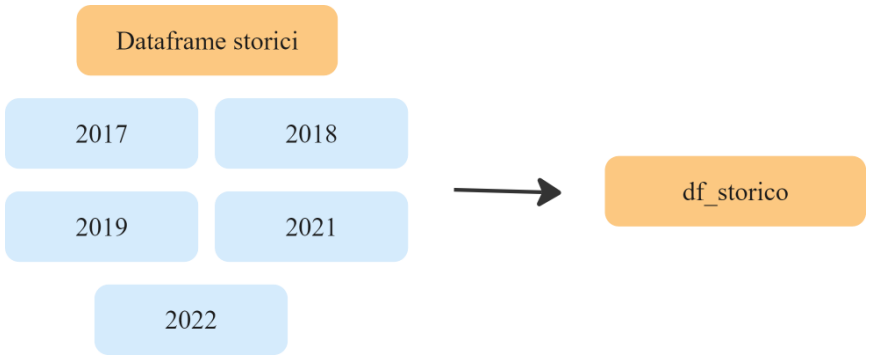

Figura 5 - Dataframe storici

[14] https://www.projectpro.io/article/data-preprocessing-techniques-and-steps/512
[15] McKinney W., & others (2010), Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56) (https://pandas.pydata.org)

Inoltre, vengono uniti al dataset storico “df_storico” anche i dati che verranno utilizzati per la predizione. In questo modo, l’intero dataset verrà sottoposto alle stesse trasformazioni e si eviteranno incongruenze di dati dovute alle diverse trasformazioni. Per creare il dataset per la predizione si sono uniti i dati delle tappe del Tour de France 2023 con la rispettiva startlist in modo da ottenere tutte le combinazioni tappa-corridore per cui era necessaria una predizione. Il “df_storico” è stato successivamente unito al dataset necessario per la predizione così da ottenere un dataset finale con tutte le informazioni, “df”.

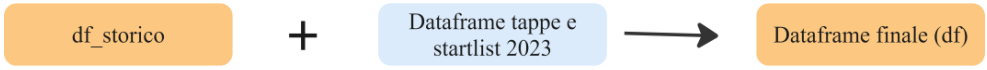

Figura 6 - Dataframe finale

2.2.2. Data cleaning

Il data cleaning è il processo di pulizia dei dati. In questa categoria rientrano: la manipolazione dei dati mancanti, l’identificazione degli outlier e la lavorazione dei dati che creano disturbi nel modello. Una delle operazioni più comuni è la lavorazione dei dati mancanti. Nel caso studio, si utilizzano due approcci differenti:

a. il processo di imputation: i valori mancanti vengono sostituiti con una stima basata su altre osservazioni. Nel caso studio, questo viene effettuato rimpiazzando i valori mancanti da una media dei valori presenti nella colonna. Ad esempio, per le colonne “weight” (peso), “height” (altezza) ed “age” (età);

```
#sostituire Nan con la media
df['weight'] = df['weight'].astype(float)
df['height'] = df['height'].astype(float)
df['age'] = df['age'].astype(float)
median_p =df['weight'].median()
median_h =df['height'].median()
df['weight'] = df['weight'].fillna(value = median_p)
df['height'] = df['height'].fillna(value = median_h)
```

Figura 7 - Processo di imputation

b. l’eliminazione: alcune righe del dataset sono state eliminate perché considerate non rilevanti per il modello. In particolare, le righe in cui il tempo impiegato dal corridore per completare la tappa è nullo, ossia DNF (Did not finish), DNS (Did not start), OTL (Outside time limit), DF (Did finish, no result) o NR (No result). Tenere questi dati nel modello porterebbe a creare inutile disturbo nei dati di training.

```
#escludo i corridori che non sono arrivati perchè creano noise nel modello = DNF o DNS o OTL o DF o NR
df = df.drop(df[df.time_tot == '-'].index)
```

Figura 8 - Processo di eliminazione

È importante considerare l’impatto della gestione dei dati mancanti sulla qualità e sull’affidabilità dei risultati ottenuti. La scelta dell’approccio dipende dal contesto in cui si sviluppa la ricerca.

2.2.3. Data transformation

La data transformation consiste nella trasformazione dei dati per adattarli alle necessità di input del modello. Ad esempio, non tutti i modelli accettano dati categorici: alcuni non sono in grado di elaborarli, quindi, richiedono una trasformazione. La scelta dell'approccio da utilizzare dipende dalla natura dei dati e dalle esigenze specifiche del problema che si sta affrontando. Per il caso studio, si sono effettuate le seguenti trasformazioni:

a. rimozione delle colonne non utili: se una colonna contente dati non è rilevante per l'analisi o l'addestramento del modello, essa può essere rimossa dal dataset;

```
#rimuovo le colonne delle features che non mi interessano = non le conosco prima che accadano
df = df.drop(columns=['Avg. speed', 'Leader after stage' 'Won how'])
```

Figura 9 - Rimozione di colonne non rilevanti

b. trasformazione della variabile di tempo dal format HH:MM:SS al format ore. Questo perché il modello non sarebbe in grado di predire il format iniziale;

```
#trasformazione del format della variabile "time_tot"
time_series = df['time_tot']

df['time_tot_h'] = time_series.dt.hour + time_series.dt.minute / 60.0 + time_series.dt.second / 3600.0
```

Figura 10 - Trasformazione variabile tempo

c. encoding delle features categoriche utili: un approccio comune che assegna a ogni valore uguale, lo stesso valore intero. Ad esempio, per le colonne "grand_tour", "stage_type", "nationality", "team", "rider".

```
from sklearn.preprocessing import LabelEncoder
df['gt_enc'] = LabelEncoder().fit_transform(df['grand_tour'].values)
df['type_enc'] = LabelEncoder().fit_transform(df['stage_type'].values)
df['nat_enc'] = LabelEncoder().fit_transform(df['nationality'].values)
df['team_enc'] = LabelEncoder().fit_transform(df['team'].values)
df['rider_enc'] = LabelEncoder().fit_transform(df['rider'].values)
```

grand_tour	gt_enc	stage_type	type_enc	nationality	nat_enc	team	team_enc	rider	rider_enc
La Vuelta ciclista a España	1	Mountains, uphill finish	4	Netherlands	31	BORA - hansgrohe	12	VAN POPPEL Danny	815
Tour de France	2	Mountains, uphill finish	4	Austria	4	BORA - hansgrohe	12	GROßSCHARTNER Felix	338
Tour de France	2	Mountains, flat finish	3	France	18	Cofidis	23	PEREZ Anthony	591
Tour de France	2	Mountains, flat finish	3	Belgium	6	Team EF Education First-Drapac p/b Cannondale	63	VANMARCKE Sep	822
Giro d'Italia	0	Mountains, flat finish	3	Italy	24	EOLO-Kometa	31	FORTUNATO Lorenzo	282

Figura 11 - Encoding delle features

Dopo gli step descritti nella sezione 2.2.3, il dataset è così composto:

Tabella 2 - Dati trasformati

Variabile	Trasformazione
year	-
grand_tour	encoded: creazione della nuova colonna gt_enc
stage_number	-

avg_speed	colonna rimossa poiché non conosco l'esito prima che la tappa avvenga: non può essere utilizzata nel training del modello
km	-
profile_score	-
stage_type	encoded: creazione della nuova colonna type_enc
won_how	colonna rimossa poiché non conosco l'esito prima che la tappa avvenga: non può essere utilizzata nel training del modello
leader_after_stage	colonna rimossa poiché non conosco l'esito prima che la tappa avvenga: non può essere utilizzata nel training del modello
team	encoded: creazione della nuova colonna team_enc
rider	encoded: creazione della nuova colonna rider_enc
age	-
nationality	encoded: creazione della nuova colonna nat_enc
weight	-
height	-
position	-
time_tot	trasformazione della variabile di tempo dal format HH:MM:SS al format ore: creazione della nuova colonna time_tot_h

2.2.4. Data exploration

La data exploration, o esplorazione dei dati, è finalizzata ad ottenere una migliore conoscenza del dataset, delle sue caratteristiche e delle relazioni tra le variabili al suo interno. È un processo fondamentale e necessario poiché aiuta a capire meglio i dati e la loro distribuzione.

Non esistono regole precise su come esplorare i dati, ma sta a ciascuno capire come affrontarla. L'obiettivo principale è arrivare a conoscere in maniera più approfondita i dati su cui si lavora facendo delle query sui dati. Il caso studio è focalizzato sulle distribuzioni dei dati e sul plottare alcuni di essi.

Esaminando attentamente il dataframe, è emerso che i ciclisti condividono caratteristiche simili. Questa somiglianza può influenzare i risultati dell'algoritmo, poiché potrebbe esserci una limitata differenziazione tra i corridori.

Ad esempio, tra i dati, si possono individuare ciclisti con età, altezza e peso:

- la distribuzione dell'età dei corridori è fortemente concentrata tra i 23 e i 33 anni. Si osserva una scarsa presenza di ciclisti oltre i 40 anni e altrettanto pochi ciclisti sotto i 20 anni;

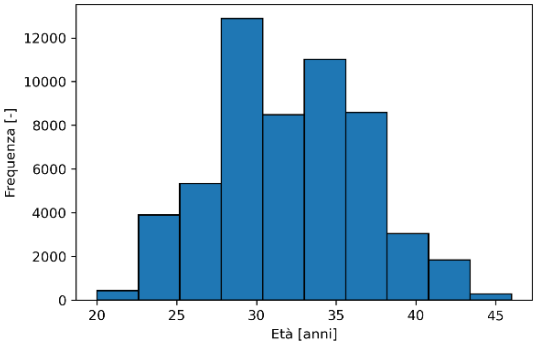

Figura 12 - Distribuzione dell'età

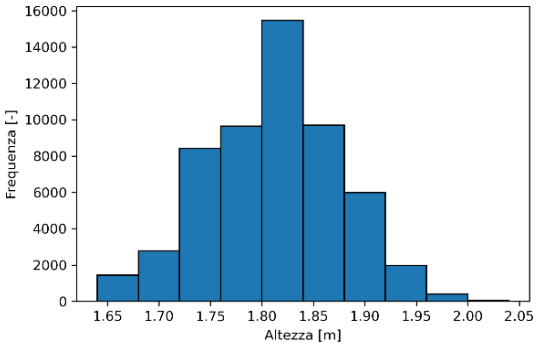

Figura 13 - Distribuzione dell'altezza

- l'altezza dei ciclisti è distribuita in modo simile a una distribuzione gaussiana. La maggior parte dei ciclisti ha un'altezza compresa tra 1.70 m e 1.90 m;

- il peso dei ciclisti è anch'esso distribuito in modo simile ad una gaussiana. La maggior parte dei ciclisti ha un peso compreso tra i 60 kg e gli 80 kg.

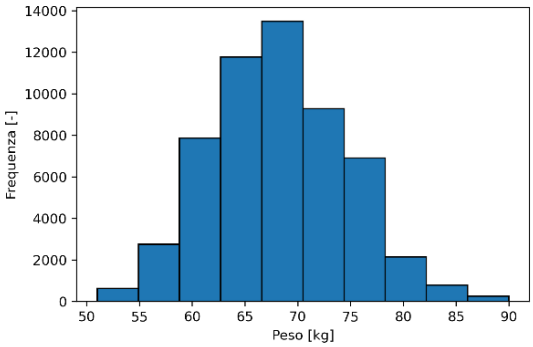

Figura 14 - Distribuzione del peso

Inoltre, un'analisi interessante riguarda la correlazione tra i dati. Di seguito, la Figura 15 mostra la matrice di correlazione tra le diverse variabili.

I valori della matrice possono assumere valori tra 1 e -1. Il valore 1 rappresenta una forte correlazione diretta; mentre, il valore -1 rappresenta una forte correlazione indiretta. Il valore 0 indica nessuna correlazione. Quindi più il valore si avvicina a 0 e più la correlazione è bassa.

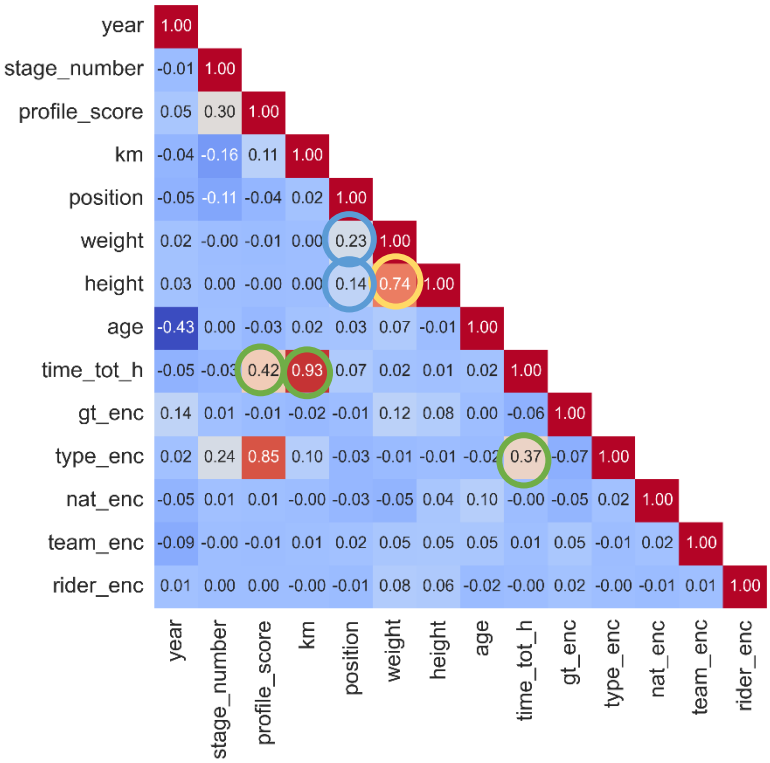

Figura 15 - Matrice di correlazione

Alcune delle correlazioni sono molto facili da comprendere: ad esempio, peso/altezza (in giallo in Figura 15).

Altre correlazioni, invece, sono più sottili e meno intuitive:

- si nota come il tempo totale sia fortemente correlato con il profile score, i km della tappa e la tipologia di tappa (in verde in Figura 15);
- la posizione del corridore correla, in modo molto basso, con il peso e l'altezza del corridore (in azzurro in Figura 15).

2.3. Il dataset

Dopo aver svolto le fasi indicate nella sezione 2.2, si è ottenuto il dataset generale delle variabili così composto in ordine alfabetico. Quando vi sono disponibili due variabili: encoded e non encoded, vengono riportate solamente le variabili encoded poiché sono quelle utilizzate nel modello.

Tabella 3 - Il dataset

Variabile
age
gt_enc
height
km
nat_enc
position
profile_score
rider_enc
stage_number
type_enc
team_enc
time_tot_h
weight
year

Successivamente, da questo dataset, si è proseguito con una selezione ulteriore. Infatti, è stato necessario:

- scegliere le features, ossia i dati da utilizzare nel modello come input per allenarlo;
- selezionare la variabile target, ossia la variabile da prevedere.

2.3.1. Le features

Di seguito, le features selezionate per allenare il modello e utilizzate come input per il calcolo della predizione:

Tabella 4 - Le features

Feature
age
gt_enc
height
km
nat_enc
profile_score

rider_enc
stage_number
type_enc
team_enc
weight
year

Per quanto riguarda le features presenti nel dataset, alcune di esse sono state spiegate nella sezione 2.1. Tuttavia, per garantire una comprensione completa e approfondita, di seguito si descrivono due features fondamentali: il profile_score e lo stage_type (type_enc). Viene di seguito approfondita la loro definizione e il significato all'interno del contesto della ricerca.

2.3.1.1. ProfileScore

Il "ProfileScore" è stato sviluppato dal Team di ProCyclingStats per assegnare dei punti ai corridori per le loro capacità di scalata. Qui è spiegato come si calcola questa variabile.

Il Team di PCS, dopo un'accurata analisi dei dati, ha scoperto che il numero dei metri verticali non è una buona indicazione su quanto sia realmente complicata una tappa. Ad esempio, una salita di 10km con pendenza all'8% seguita da 100km di strada pianeggiante sarebbe più adatta a terminare in uno sprint di gruppo. Invece, se la stessa salita è posizionata negli ultimi 10km della tappa, essa si addice maggiormente a corridori definiti scalatori.

Si includono, quindi, tre variabili nella formula del ProfileScore:

- la posizione della salita dal traguardo;
- la pendenza della salita;
- la lunghezza della salita.

Si calcola così lo score di ogni singola salita all'interno della tappa con la seguente formula:

$$\left(\frac{[Pendenza]}{2}\right)^2 * [Lunghezza\ in\ km]$$

Successivamente, si moltiplica questo punteggio per un fattore dipendente, ossia la distanza della salita dal traguardo, come segue:

Tabella 5 - Km e fattore dipendente

Ultimi km	**Fattore dipendente**
10	1.0
25	0.8
50	0.6
75	0.4
prima degli ultimi 75km	0.2

Maggiore sarà il valore del fattore dipendente, tanto più sarà alto il coefficiente di difficoltà del percorso. La formula finale sarà così formata:

$$\left(\frac{[Pendenza]}{2}\right)^2 * [Lunghezza\ in\ km] * FattoreDipendente$$

Ad esempio:

Caso A: tappa piatta con 10km all'8% negli ultimi 10km

$$\left(\frac{8}{2}\right)^2 * 10 * 1 = 160$$

Caso B: tappa piatta con 10km all'8% a 100km dal traguardo

$$\left(\frac{8}{2}\right)^2 * 100 * 0.2 = 32$$

Caso C: tappa con due salite, la prima 10km al 10% a 40km dal traguardo; la seconda 4km al 12% negli ultimi 10km

$$\left(\frac{10}{2}\right)^2 * 10 * 0.6 + \left(\frac{12}{2}\right)^2 * 4 * 0.1 = 294$$

2.3.1.2. StageType

Per definire le caratteristiche del tracciato, oltre al ProfileScore, vengono anche utilizzati cinque indicatori di tappa:

1. piatta;
2. collinare, con arrivo piatto;
3. collinare, con arrivo in salita;
4. di montagna, con arrivo piatto;
5. di montagna, con arrivo in salita;

Questi elementi forniscono un'indicazione del tipo di percorso. Potrebbe essere che una tappa in una gara ciclistica con lo stesso ProfileScore sia considerata piatta e, invece, in un'altra sia considerata collinare. Una volta calcolato il PS, esso è spesso rappresentato da uno di questi cinque indicatori che sono un punteggio assoluto della difficoltà della tappa.

2.3.2. Le variabili target

Di seguito, le variabili target:

Tabella 6 - Le variabili target

Variabile target
time_tot_h
position

Come detto in precedenza, le variabili target sono l'output del modello. Si tratta dell'obiettivo che si vuole predire o stimare utilizzando il modello. Inizialmente, l'idea era di utilizzare come target la previsione del vincitore di tappa, con una modalità di classificazione binaria: "sì" o "no". Successivamente, analizzando i dati e la loro distribuzione, si è capito che era un target difficile da prevedere a causa di molti dati mancanti delle caratteristiche dei corridori e dello sbilanciamento dei dati. Di conseguenza, il modello sarebbe stato poco accurato perché non avrebbe, nella maggior parte dei casi, identificato il vincitore ma avrebbe predetto molti "non vincitori". Quindi, osservando i dati, si è deciso di individuare come target due diverse variabili:

1. il tempo totale impiegato per concludere la tappa (time_tot_h);
2. la possibile classifica generale di ogni singola tappa (position).

Si è optato per due diverse variabili per capire come i due modelli si sarebbero comportati. Le due variabili hanno lo stesso obiettivo: predire la classifica di tappa, ma affrontano il problema in maniera differente e hanno differenti limiti. Nei due paragrafi seguenti 2.4 e 2.5, si analizzano la fase di costruzione e di utilizzo del modello.

2.4. La costruzione del modello

Per la costruzione del modello, si è utilizzato l'algoritmo XGBoost[16] e la libreria Scikit-Learn[17]. L'uso combinato di XGBoost e Scikit-Learn ha permesso di sfruttare al meglio le potenzialità di entrambi gli strumenti: XGBoost ha fornito la capacità di gestire complessi problemi di regressione e classificazione, mentre Scikit-Learn ha semplificato il processo di sviluppo del modello grazie alla sua vasta gamma di funzioni integrate. Questa sezione si suddivide in quattro fasi chiave:

- la selezione del modello da utilizzare, accompagnata da una breve spiegazione teorica del modello scelto;
- la spiegazione dei concetti di training set, validation set e test set, che sono componenti fondamentali del processo di addestramento e valutazione del modello;
- i parametri utilizzati nel modello, poiché essi sono essenziali nel determinare il comportamento e la capacità predittiva del modello stesso;
- la fase di back testing, ossia la valutazione dei due modelli creati.

2.4.1. La tipologia del modello

L'algoritmo utilizzato è un albero decisionale potenziato. Gli alberi decisionali sono costituiti da dei nodi che rappresentano eventi o scelte e da dei rami che rappresentano la decisione o la condizione. Ciascun albero consiste in nodi e foglie. Ciascun nodo rappresenta attributi e ciascuna foglia rappresenta il valore che quel nodo può assumere. Il termine "potenziato", o "boosting", fa riferimento ad una famiglia di algoritmi che convertono le debolezze nell'imparare in punti di forza. Il potenziamento è una tecnica nell'apprendimento dell'insieme che viene utilizzata per ridurre la distorsione e la varianza[18].

[16] Chen T., Guestrin C. (2016), XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (https://doi.org/10.48550/arXiv.1603.02754)
[17] Pedregosa F., & others (2012), Scikit-Learn: Machine Learning in Python (pp. 2825-2830) (https://doi.org/10.48550/arXiv.1201.0490)
[18] Mahesh B. (2018), Machine Learning Algorithms - A Review. International Journal of Science and Research

Dal punto di vista pratico, questo modello lavora imparando da un numero di alberi decisionali. Ciascun albero viene allenato su un sottoinsieme di dati e le predizioni di ciascun albero sono combinate per creare la predizione totale[19]. Da un modello iniziale, derivano degli errori da cui si costruiscono nuovi modelli. I modelli errore-predizione vengono uniti all'insieme dei modelli. Quindi, per fare una previsione, si aggiungono le previsioni di tutti i modelli precedenti.

Si può sintetizzare graficamente il processo teorico nella Figura 16.

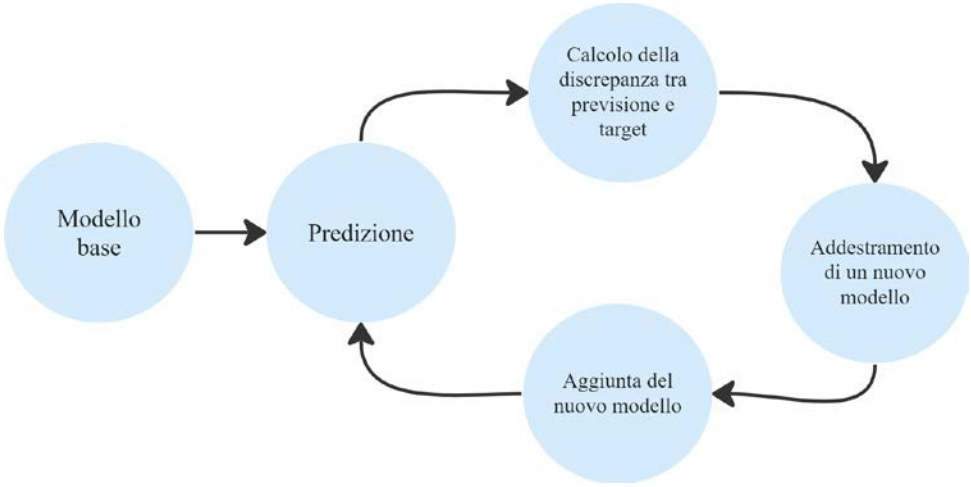

Figura 16 - Teoria del modello XGBoost

In generale, la costruzione del modello combina diversi modelli deboli per creare un modello più potente. Durante il processo di addestramento, XGBoost ottimizza una funziona obiettivo che misura la discrepanza tra le previsioni del modello e i valori effettivi dell'output target. L'obiettivo di XGBoost è minimizzare questa discrepanza con l'iterazione e l'aggiunta sequenziale di nuovi modelli deboli. Ogni nuovo modello debolmente appreso viene addestrato per predire i residui rimanenti e ridurre così l'errore complessivo del modello. In questo modo, l'apprendimento dell'insieme permette di avere delle predizioni migliori rispetto ad usare un singolo albero decisionale.

2.4.2. Training set, validation set e test set

Per poter costruire il modello, è necessario avere un training set, validation set e test set[20]:

- training set: si tratta del set di dati utilizzato per addestrare il modello a predire. È il set di dati da cui il modello impara le correlazioni tra features e variabile target. Il modello impara da questi dati eseguendo l'algoritmo e mappa la funzione f(x) dove "x" corrisponde all'input per "y", ossia l'output. In questa fase, sul set di dati disponibile, si forniscono sia le variabili di input che di output in modo che il modello sia in grado di imparare a prevedere la variabile target (output) in base alle features utilizzate (input);

[19] https://medium.com/@techynilesh/xgboost-algorithm-explained-in-less-than-5-minutes-b561dcc1ccee
[20] https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

- validation set: si tratta del set di dati per valutare il modello su dati non analizzati dal modello durante l'addestramento, o fase di training. Il validation set si utilizza per definire i parametri dell'algoritmo (sezione 2.4.3);
- test set: si tratta della validazione del modello. Il test set è fondamentale poiché è utilizzato per verificare l'accuratezza del modello confrontando il risultato previsto dal modello con l'esito effettivo del test set. In questa fase, non si forniscono variabili di output al modello, anche se si conoscono già i risultati, ma si tengono da parte per poi confrontarli con la predizione.

Non esiste una regola specifica per la divisione del dataset totale in training, validation e test, ma dipende dal tipo di modello, dal suo uso e dalla dimensione dei dati[21].

All'interno della ricerca, avendo disponibili cinque anni di dati storici, per costruire il modello, si è formato:

- il training set dagli anni 2017, 2018, 2019;
- il validation set dall'anno 2021;
- il test set dall'ultimo anno rimanente, il 2022.

Figura 17 - Training, validation, test set

2.4.3. Parametri del modello

Nel caso studio, si è utilizzato il validation set per valutare il modello ed effettuare la scelta dei parametri. Questa operazione viene chiamata tuning. Esistono dei metodi di auto-tuning dove il modello automaticamente prova differenti parametri e decide quali siano i migliori. Tuttavia, in questo caso, si è optato per un tuning manuale, provando i differenti parametri manualmente e vedendo come i risultati sul validation set variassero. Per comprendere se i valori dei parametri fossero corretti, ci si è affidati alla radice dell'errore quadratico medio, o RMSE, che misura la differenza media tra la previsione e i valori attuali. Più l'RMSE è basso, più il modello è capace di minimizzare l'errore di previsione.

Successivamente, sulla base dei risultati, si sono scelti i parametri necessari per creare il modello, di seguito nella Figura 18.

```
gbm = xgb.XGBRegressor(
                       n_estimators=1000,
                       max_depth=10,
                       objective="reg:squaredlogerror",
                       learning_rate=.01,
                       subsample=.75,
                       min_child_weight=1,
                       colsample_bytree=.8
                       )
```

Figura 18 - I parametri

[21] https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data

Spiegazione dei parametri[22]:

- n_estimators: il numero di alberi da addestrare nel processo di boosting. Un valore più alto migliorerà la capacità di previsione del modello, ma impiegherà più tempo. Il range tipico è tra 100 e 1000. Nel caso studio, avendo il dataframe di piccole dimensioni, si è potuto inserire un elevato numero di alberi;
- max_depth: profondità massima dell'albero decisionale. La profondità più è maggiore, più saranno specifiche le relazioni tra i dati. Viene utilizzato per controllare l'over fitting, ossia la capacità del modello di fare previsioni molto accurate per i dati di validation, ma non per il test set. Il range tipico è tra 3 e 10;
- objective: definisce la funzione di discrepanza da minimizzare. In questo caso, si è optato per una funzione adatta alla regressione;
- learning_rate: è il tasso di apprendimento, ossia quanto il modello si basa sui nuovi alberi decisionali aggiunti. Il range è tra 0.01 e 0.2;
- subsample: la proporzione dei campioni di addestramento utilizzati per addestrare ciascun albero. Il range è tra 0.5 e 1. In questo caso, 0.75 significa che il modello sceglie in modo randomico il 75% del dataset disponibile ed esclude il rimanente 25%, quando crea i nuovi alberi. Questo avviene per produrre previsioni più generalizzate, ossia sempre per controllare l'over fitting;
- min_child_weight: controlla la complessità dell'albero regolando la quantità di suddivisioni che vengono effettuate negli alberi. Il range è tra 0 e ∞. Più è alto il valore, più conservativo sarà l'algoritmo con meno foglie e nodi. Il valore scelto in questo caso è 1;
- colsample_bytree: il concetto è simile a subsample, ma viene effettuato sulle colonne. Il range è tra 0 e 1.

2.4.4. Back testing

Una volta terminato il manual-tuning e la definizione dei parametri, si è testato il modello sull'anno 2022, periodo in cui la variabile target da prevedere è già avvenuta. Questo è stato effettuato per poter ricavare delle metriche di valutazione del modello che sono basate sulla comparazione della predizione del modello e del reale valore verificatosi. Di seguito, la valutazione dei due modelli.

2.4.4.1. Modello predizione time

Il modello per la predizione della variabile target "time", che rappresenta il tempo totale impiegato per concludere la tappa, dimostra un'accuratezza elevata. Tuttavia, il modello non è altrettanto preciso per quanto riguarda l'ordine in cui vengono elencati i corridori. Questo accade perché, come indicato dalla Figura 19, le features più rilevanti utilizzate nel modello sono legate al percorso e non alle caratteristiche specifiche del corridore che hanno un'importanza minima.

[22] https://www.kaggle.com

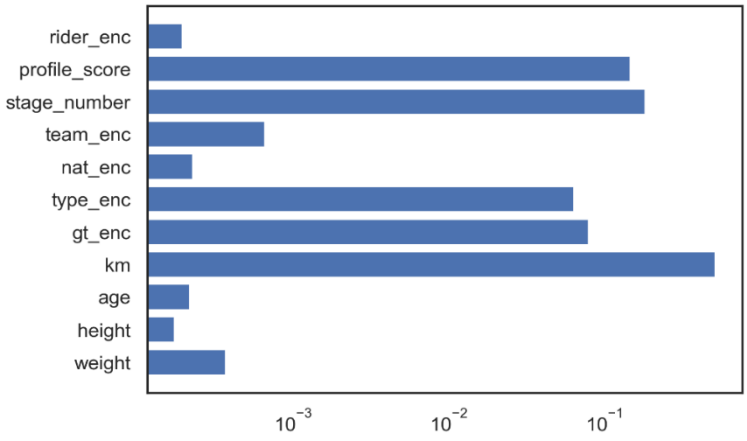

Figura 19 - Rilevanza delle features modello time

Di conseguenza, il modello sarà in grado di effettuare previsioni accurate per il tempo totale impiegato dai corridori, ma potrebbe non essere in grado di predire correttamente il nome del corridore che compie il tempo minore. L'accuratezza del modello viene misurata grazie al parametro R2, su una scala da 0 a 1, dove 1 rappresenta un'accuratezza perfetta. Questo modello ha un R2 score di 0.9: indica che il modello è in grado di spiegare il 90% della variazione della variabile target "time".

Di seguito, nella Figura 20, si osserva la distribuzione dei dati: se le predizioni fossero tutte corrette, esse sarebbero perfettamente sulla retta di regressione $x = y$.

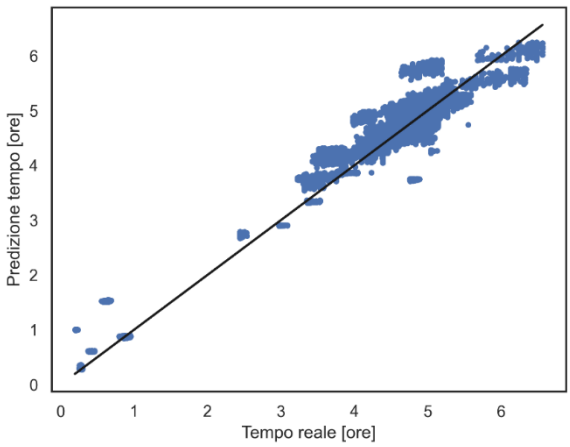

Figura 20 - Distribuzione delle features modello time

In questo caso, la maggior parte delle predizioni è molto vicina alla retta di regressione, indicando un errore del modello molto basso. Tuttavia, i diversi punti rappresentati dovrebbero corrispondere a diversi corridori. È evidente che la distribuzione di questi punti è molto concentrata. Di conseguenza, come anticipato, il modello è efficace nel prevedere un tempo medio impiegato per concludere la tappa, ma non è accurato nel definire il tempo impiegato da corridori differenti. Nel capitolo 3, si analizzano e discutono nel dettaglio i risultati ottenuti.

2.4.4.2. Modello predizione position

Il modello utilizzato per la predizione della posizione, che calcola la classifica generale dei corridori all'interno della tappa, presenta un cambiamento significativo nell'importanza delle features. Poiché il target è diverso, le caratteristiche del percorso non sono più utili ed influenti come in precedenza. Infatti, le informazioni specifiche del corridore come l'età, l'altezza e il peso assumono un peso maggiore, come osservato dalla Figura 21.

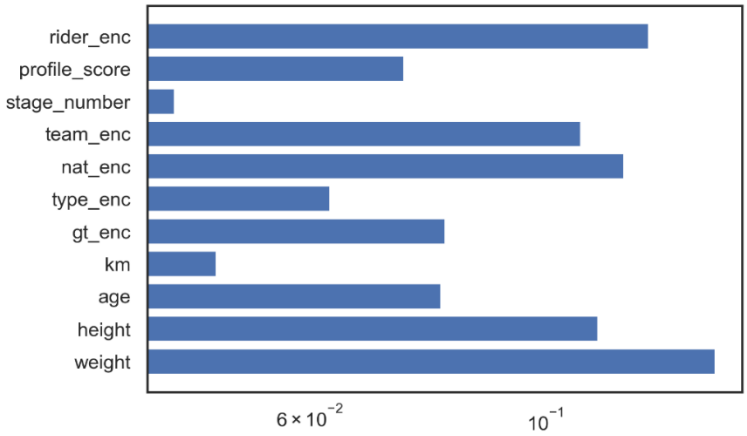

Figura 21 - Rilevanza delle features modello position

L'R2 score del modello è 0.26, il che indica che il modello è in grado di spiegare solo circa il 26% della variazione nella posizione dei corridori. Questo suggerisce che le caratteristiche del rider non sono sufficienti per determinare la posizione di arrivo con un errore limitato.

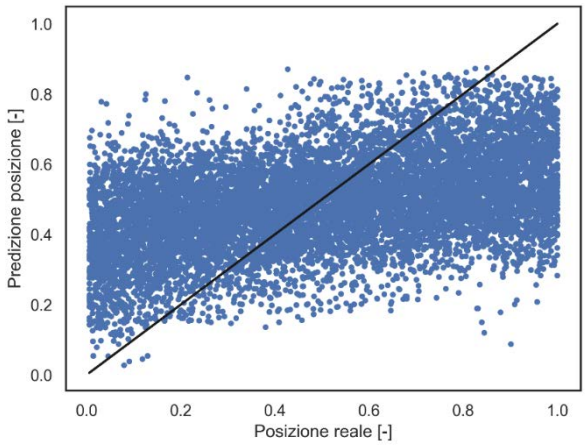

Figura 22 - Distribuzione delle features modello position

In questo caso, le predizioni non seguono la linea di regressione e, alcune volte, si trovano anche lontano da essa: questo indica un ampio errore. I diversi punti sono più definiti rispetto al modello precedente, di conseguenza, questo significa che il modello position è in grado di differenziare maggiormente i corridori.

Infine, è importante notare che le prestazioni di un ciclista sulla bici possono essere influenzate da una serie di fattori complessi che vanno oltre le caratteristiche generali del corridore, come l'età,

l'altezza e il peso. È necessario, quindi, avere delle informazioni più specifiche per aumentare l'accuratezza del modello, come si approfondisce nel capitolo 4.

2.5. L'utilizzo del modello

Dopo aver costruito il modello, il secondo step è l'utilizzo. Per fase di utilizzo, si intende l'applicazione del modello per compiere previsioni su avvenimenti futuri, di cui ancora non si possiede il valore della variabile target. Per fare previsioni si utilizzano solamente:

- training set: composto dal dataset degli anni precedenti l'evento da prevedere, ossia dal 2017 al 2022, con l'eccezione dell'anno 2020 a causa della pandemia da Covid-19, utilizzando le features descritte nella sezione 2.3.1;
- predizione: composto dalla variabile target, l'obiettivo di previsione (sezione 2.3.2).

In questa fase, il validation e il test set non sono più necessari.

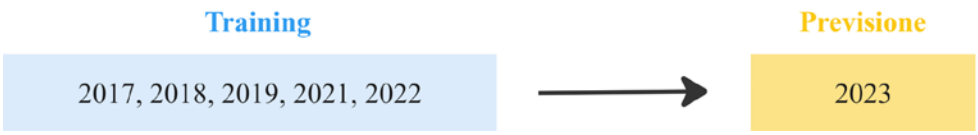

Figura 23 - Utilizzo del modello

Nel capitolo 3, si analizzano i risultati delle previsioni ottenuti dall'utilizzo dei due modelli costruiti in precedenza: il modello time e il modello position.

3. Risultati

I risultati della ricerca hanno confermato l'ipotesi iniziale, sottolineando come sia necessario integrare nuove features rilevanti per ottenere previsioni più accurate e fedeli alla realtà.

Durante alcune tappe, le previsioni dell'algoritmo hanno incluso alcuni corridori che sono caduti o addirittura si sono ritirati. Al termine della tappa, questi ciclisti sono stati esclusi dalle previsioni future, eliminandoli dalla startlist.

Inoltre, il corridore Egan Bernal, secondo il modello, era favorito in numerosissime tappe ed era uno dei potenziali vincitori di questo Tour de France, considerate le sue prestazioni passate. Tuttavia, il modello non è a conoscenza della situazione attuale: il ciclista ha subito un grave incidente nel gennaio 2022, che non gli ha permesso di avere una preparazione adeguata, come si spiega nel capitolo 4. Di conseguenza, il corridore è stato escluso dall'analisi dei risultati.

3.1. Modello time

Dalle previsioni ottenute dal modello time, che stima il tempo totale impiegato per concludere la tappa, si dimostra che:

- i tempi sono sovrastimati in media del 5%;
- solamente in quattro tappe su ventuno totali, il modello sottostima il tempo impiegato;
- nella tappa 21, il modello ha un errore dello 0%.

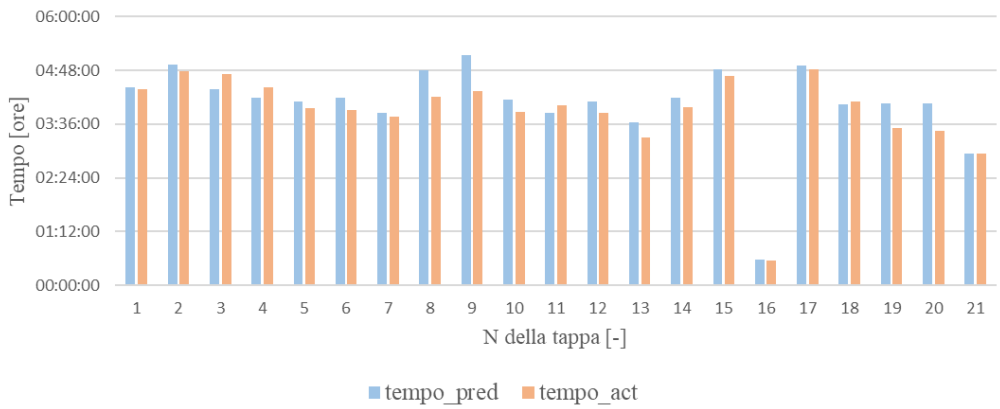

Figura 24 – Tempo predizione vs reale

In generale, sommando i tempi delle ventuno tappe totali del Tour de France 2023, il modello predice un tempo totale di 85:19:44. Mentre, sommando il tempo totale, reale e minore di tutte le tappe, il risultato è 81:33:51 (con uno scarto del modello di +4.6%).

In conclusione, come si analizza nel paragrafo 2.4.4.1, il modello è accurato nel predire il tempo minore di percorrenza di ciascuna tappa. Per dati più dettagliati, fare riferimento all'Appendice 1.

3.2. Modello position

Dalle previsioni ottenute dal modello position, che stima l'ordine di arrivo di una tappa, si dimostra che:

- il modello predice correttamente sei volte su ventuno il vincitore di tappa, ossia nel 29% dei casi;

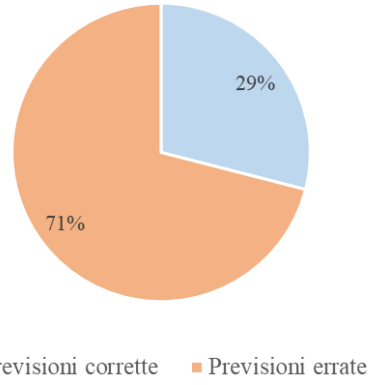

Figura 25 - Predizione vincitore di tappa

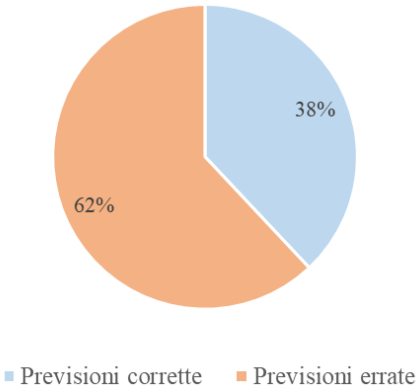

Figura 26 - Predizione top 15

- il modello predice correttamente nel 38% dei casi i corridori che terminano la tappa in top 15, escludendo l'ordine corretto della classifica. Si considerano i primi 15 corridori poiché ricevono dall'UCI (Unione Ciclistica Internazionale) dei punti, che dipendono dal piazzamento e dall'importanza della competizione stessa, per creare il World Ranking UCI.

In conclusione, questo modello, come si analizza nella sezione 2.4.4.2, non è ancora molto accurato. Tuttavia, è stato comunque in grado di predire sei vincitori di tappa. Con i miglioramenti presi in considerazione nel capitolo 4, l'algoritmo potrebbe raggiungere buoni livelli di accuratezza. Per dati più dettagliati, fare riferimento all'Appendice 2.

4. Limiti del modello e miglioramenti

Nel contesto del modello di machine learning creato, è importante riconoscere i limiti e le aree di miglioramento. Infatti, esistono diversi fattori che il modello non è in grado di considerare, ma che possono influire significativamente sulle prestazioni dei corridori durante la tappa e, di conseguenza, anche sulle previsioni dei risultati.

Innanzitutto, la mancanza di considerazione delle tattiche presenti nel mondo del ciclismo. Durante una gara si possono sviluppare diversi scenari che non sono prevedibili e che dipendono, non solo dalle prestazioni del corridore stesso, ma anche dalle prestazioni dei suoi compagni, degli altri corridori e di obiettivi che la squadra si prefissa. Per esempio, un evento difficile da riprodurre in un algoritmo è la cosiddetta, in gergo ciclistico, “fuga bidone”: si tratta di fughe che solitamente avvengono verso la metà dei grandi giri, in cui corridori che avevano come obiettivo la classifica generale, e fino a quel momento hanno deluso le aspettative per motivi più disparati (cadute, scarsa condizione...), cercano di recuperare tempo in classifica generale andando in fuga. Siccome già nella seconda settimana di una gran tour la classifica è ben stilata e i distacchi sono già consolidati, specialmente tra i primi, questi corridori vengono lasciati andare in fuga dagli uomini di classifica più forti, in quanto non costituiscono un vero pericolo per le alte posizioni, e in questo modo riescono a recuperare tempo in classifica a discapito dei loro diretti concorrenti di media classifica. Un altro esempio è il caso in cui un corridore che negli anni passati ha performato nei grandi giri si ritrovi improvvisamente, a causa di un calo fisiologico dovuto all'età e alla crescita dei corridori più giovani, a dover svolgere nella stessa squadra ruolo di gregariato e, quindi, a dover spendere le sue energie a servizio di un compagno: sebbene nel passato sia stato possibile per lui fare risultato in gare da tre settimane, probabilmente non troveremo più il corridore nell'alta classifica.

Inoltre, ci sono fattori legati strettamente ai corridori durante un Grande Giro: l'equilibrio psicofisico del ciclista e le cadute in cui incorrono durante la tappa. Infatti, alcuni potrebbero essere più suscettibili agli effetti negativi di tali variabili, portando a prestazioni non coerenti con le previsioni del modello. Ad esempio, lo sprinter Jakobsen era un nome ricorrente nelle previsioni di tappe piatte nel Tour de France 2023. Il problema è che il corridore è caduto nella tappa 4 riportando gravi conseguenze fisiche con il successivo ritiro dal Tour nella tappa 12.

Un’altra variabile esterna non prevedibile è la possibilità che accadano degli incidenti meccanici. Un guasto meccanico può influire notevolmente sulle prestazioni di un corridore.

Un altro fattore da considerare è la presenza di infortuni gravi avvenuti nel passato per corridori che attualmente cercano di recuperare la condizione fisica precedente. Come nel caso di Egan Bernal, vincitore nel 2019 del Tour de France e nel 2021 del Giro d’Italia, che ha subito un gravissimo incidente in allenamento nel gennaio 2022, rischiando la vita. Questo corridore, date le sue eccellenti prestazioni passate, sarebbe stato uno dei favoriti per la vittoria del Tour de France 2023, tanto che è uno nome ricorrente nelle previsioni effettuate dal modello. Tuttavia, il modello non è a conoscenza della situazione attuale.

Esistono anche variabili esterne legate alle condizioni meteorologiche che possono influire sulle prestazioni individuali. Vi sono corridori che con un caldo eccessivo non sono in grado di performare come se ci fosse un clima mite. In aggiunta, la presenza di pioggia durante una tappa può influenzare significativamente sulla velocità e sul tempo totale impiegato per completare la tappa. In condizioni di pioggia, le strade possono diventare scivolose, aumentando il rischio di

cadute e influenzando la strategia di gara dei corridori. Questo aspetto, non contemplato dal modello, potrebbe portare a deviazioni dalle previsioni e richiedere ulteriori considerazioni. Una possibile soluzione per affrontare l'effetto delle condizioni meteorologiche sulle prestazioni dei corridori potrebbe essere l'implementazione di API e database che forniscono uno storico delle condizioni meteorologiche del passato. Integrare tali dati nel modello consentirebbe di considerare l'influenza delle condizioni meteorologiche sulle prestazioni dei corridori durante la tappa. I dati del database potrebbero essere integrati nel processo di addestramento del modello, consentendogli di apprendere le relazioni tra le condizioni meteorologiche e le prestazioni dei corridori.

La regola dei 3 km è un'altra variabile che può influenzare i risultati. Infatti, a causa della pericolosità degli sprint, dovuta all'alta velocità e alla ricerca della vittoria da parte di un alto numero di corridori più adatti a questi tipi di finali, è stata introdotta una regola che consiste nella neutralizzazione degli ultimi tre chilometri ai fini della classifica finale. Nelle tappe piatte dove le differenze tra gli uomini di classifica sono pressoché inesistenti, i velocisti possono partecipare alla volata per la vittoria finale senza essere intralciati dalle squadre degli uomini di classifica: si diminuiscono così i rischi di incidenti, diminuendo le tensioni all'interno del gruppo e riducendo la quantità di corridori ad alte velocità negli ultimi chilometri. Il fatto che questa regola non venga inclusa nell'algoritmo sviluppato comporta di trovare uomini di classifica generale nelle classifiche di tappa che si ottengono con il modello, anche se nella realtà non partecipano agli sprint, ma spesso si piazzano involontariamente tra la quindicesima e ventesima posizione. Sebbene i corridori di classifica non siano in grado di competere con uno sprinter, a causa di caratteristiche fisiche differenti, le loro prestazioni potrebbero essere buone nel caso in cui non esistesse questa regola, e molto probabilmente riuscirebbero a piazzarsi anche nelle volate, ma in realtà non vi partecipano in quanto non rappresentano il loro obiettivo e, inoltre, comporterebbero ulteriori rischi, come spiegato precedentemente.

Infine, fattori come l'innovazione tecnologica all'intero del ciclismo, l'uso di materiali specifici, la preparazione atletica individualizzata per ogni singolo corridore in base al ruolo all'interno del team e l'alimentazione possono influenzare le prestazioni dei corridori. Il modello non tiene conto di queste variabili che possono avere un'incidenza significativa sulle prestazioni. Questi fattori non sono stati inclusi all'interno del modello poiché sono informazioni private di ogni singola squadra.

In conclusione, per migliorare l'accuratezza del modello, potrebbe essere necessario esplorare ulteriormente questi fattori e integrarli nell'analisi. Comprendere e considerare tali variabili aggiuntive potrebbe fornire un quadro più completo e preciso delle prestazioni dei corridori nelle gare ciclistiche.

Conclusione

Nella tesi che si è sviluppata, si è affrontata l'applicazione del machine learning allo sport e, in particolare, al Tour de France 2023. La tesi si focalizza dapprima sulla definizione generale di artificial intelligence e machine learning, per poi spostarsi sul caso studio e approfondire tecnicamente la costruzione e l'utilizzo di un modello XGBoost.

Grazie ai modelli costruiti, è stato possibile predire il tempo impiegato per completare una tappa e il potenziale ordine di arrivo. Come si è visto nel capitolo 3, i risultati, però, hanno ampio margine di miglioramento. I modelli costruiti hanno dei limiti, come spiegato nel capitolo 4: alcuni di questi limiti possono essere superati andando ad aggiungere ulteriori dati e features al modello. Altri, invece, rimangono limiti molto difficili da superare, come ad esempio l'introduzione della strategia di gara.

Per concludere, con questa tesi, si è dimostrato il processo di costruzione di un modello comprendendo le difficoltà in cui si può incorrere e le possibili strade da intraprendere per sorpassarle. In futuro, l'avanzamento tecnologico e la possibilità di accesso a ulteriori dati potrebbero rivelarsi chiave per sviluppare modelli sempre più accurati e precisi a sostegno degli atleti e dello sport.

Bibliografia

[1] Hilmkil A., Ivarsson O., Johansson M., Kuylenstierna D., van Erp T. (2018), Towards

[2] Van Kuijk K., Dirksen M., Seiler C. (2023), Conformal Regression in Calorie Prediction for Team Jumbo-Visma

[3] Durante M. (2022), Potere computazionale: dalle informazioni ai dati. In Mimesis (Ed.), La politica dei dati. Il governo delle nuove tecnologie tra diritto, economia e società (pp. 59-63)

[5] Norvig P., Russell S. (2010), Introduction. In Pearson College (Ed.), Artificial Intelligence: A Modern Approach 3rd Edition (pp. 1-5)

[11] Van Rossum G., Drake Jr. F. L. (1995), Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam

[12] Chandra R. V., Varanasi B. S. (2015), Python requests essentials. Packt Publishing Ltd

[13] Richardson L. (2007), Beautiful soup documentation

[15] McKinney W., & others (2010), Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56)

[16] Chen T., Guestrin C. (2016), XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[17] Pedregosa F., & others (2012), Scikit-Learn: Machine Learning in Python (pp. 2825-2830)

[18] Mahesh B. (2018), Machine Learning Algorithms - A Review. International Journal of Science and Research

Sitografia

[4] https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History

[6] https://hub.packtpub.com/5-key-reinforcement-learning-principles-explained-by-ai-expert/

[7] https://quantdare.com/machine-learning-a-brief-breakdown/

[8] https://towardsdatascience.com/machine-learning-a-gentle-introduction-17e96d8143fc

[9] https://www.procyclingstats.com

[10] https://code.visualstudio.com

[14] https://www.projectpro.io/article/data-preprocessing-techniques-and-steps/512

[19] https://medium.com/@techynilesh/xgboost-algorithm-explained-in-less-than-5-minutes-b561dcc1ccee

[20] https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[21] https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data

[22] https://www.kaggle.com

Appendice 1

Stage	Tempo previsto	Tempo reale	Scarto
1	04:25:14	04:22:49	1%
2	04:55:38	04:46:39	3%
3	04:23:28	04:43:15	-7%
4	04:11:31	04:25:28	-5%
5	04:05:46	03:57:07	4%
6	04:11:12	03:54:27	7%
7	03:50:42	03:46:28	2%
8	04:47:56	04:12:26	14%
9	05:08:20	04:19:41	19%
10	04:08:51	03:52:34	7%
11	03:51:30	04:01:07	-4%
12	04:06:36	03:51:42	6%
13	03:38:11	03:17:33	10%
14	04:10:48	03:58:45	5%
15	04:49:30	04:40:45	3%
16	00:34:14	00:32:36	5%
17	04:54:01	04:49:08	2%
18	04:02:17	04:06:48	-2%
19	04:04:04	03:31:02	16%
20	04:03:51	03:27:18	18%
21	02:56:04	02:56:13	0%
Totale	85:19:44	81:33:51	4,62%

Appendice 2

Stage	Primo classificato – predizioni corrette	Corridori top 15 – predizioni corrette	Analisi
1	no	7	Tappa decisa tra gli uomini di classifica più forti
2	no	7	Tappa decisa tra gli uomini di classifica più forti
3	sì	4	Sprint di gruppo
4	sì	9	Sprint di gruppo
5	no	7	Principalmente fuga con anche uomini di classifica
6	sì	7	Tappa decisa tra gli uomini di classifica più forti
7	sì	6	Sprint di gruppo
8	no	4	Sprint di gruppo
9	no	2	Fuga
10	no	1	Fuga
11	sì	8	Sprint di gruppo
12	no	2	Fuga
13	no	7	Principalmente fuga con anche uomini di classifica
14	no	9	Tappa decisa tra gli uomini di classifica più forti
15	no	3	Fuga
16	no	6	ITT
17	no	9	Principalmente fuga con anche uomini di classifica
18	no	6	Fuga non ripresa + sprint di gruppo
19	no	4	Fuga
20	sì	7	Tappa decisa tra gli uomini di classifica più forti
21	no	5	Sprint di gruppo
Totale	6/21 vincitori di tappa	120/315 corridori in top 15	