



Gli alberi decisionali e le foreste casuali

Come non smarrirsi in una selva oscura

In questo capitolo affronteremo l'ultimo tra gli algoritmi di machine learning supervisionato presentati in questo libro: le cosiddette **foreste casuali**, comunemente chiamate con il nome inglese *random forest*.

Si tratta di modelli di machine learning molto diversi dalle reti neurali, sia dal punto di vista concettuale sia da quello matematico e ne vedremo il perché.

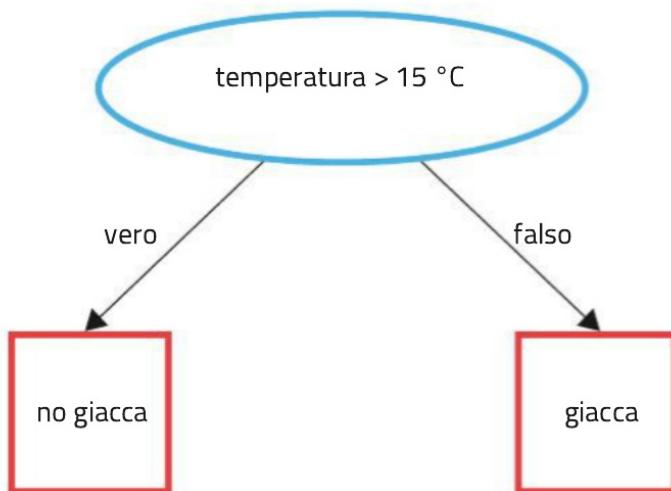
Inizieremo a introdurre il concetto di albero decisionale e vedremo che esso sarà l'elemento fondamentale della foresta casuale, proprio come una foresta naturale è costituita principalmente da alberi!

Una volta addentrati nella foresta proveremo ad applicare questo nuovo algoritmo a un problema reale di classificazione.

1 Gli alberi decisionali

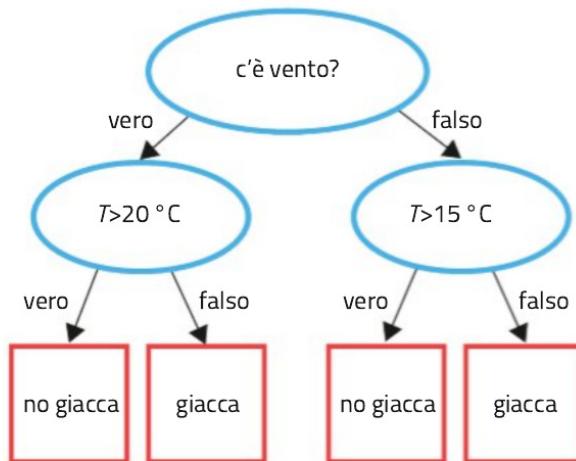
Immaginiamo di essere sul punto di uscire di casa e di porci la seguente domanda: «devo indossare una giacca?». Le possibili risposte sono due: sì oppure no.

Possiamo sciogliere il dubbio leggendo un termometro che misura la temperatura esterna. In base al suo valore possiamo poi decidere se indossare la giacca. Per esempio, possiamo decidere che se la temperatura è maggiore di 15 °C allora non indosseremo la giacca. Il nostro algoritmo di scelta è rappresentato nella figura qui sotto.



La figura mostra un **albero** decisionale a un livello costituito da un **nodo decisionale** in blu, che corrisponde alla domanda «la temperatura è maggiore di 15° C?» e due nodi foglia in rosso che corrispondono ai nodi terminali relativi alle due possibili decisioni: indosso la giacca oppure no. La temperatura è la feature presa in considerazione dal nodo decisionale.

Elaboriamo un algoritmo più sofisticato che tenga conto anche della presenza o meno di vento. In caso di vento la soglia di temperatura per decidere se indossare una giacca sarà più alta, supponiamo 20 °C. Modifichiamo l'albero di conseguenza.



La figura descrive questo nuovo algoritmo e questa volta i livelli dell'albero sono due, per un totale di tre nodi decisionali. L'esempio precedente ci aiuta a capire la definizione del concetto di **albero decisionale**.

Albero decisionale o **albero di decisione**: è un albero di **nodi decisionali**, ognuno dei quali è associato a una domanda su una feature. Da ogni nodo derivano tanti archi quante sono le possibili risposte alla domanda. Il nodo di partenza è detto **nodo radice** (o in inglese *root*). I nodi terminali sono detti **nodi foglia** (o in inglese *leaf*), e ognuno di essi è associato a uno dei possibili esiti finali della decisione (**le categorie**).

L'esempio appena visto è un albero decisionale che implementa un algoritmo per prendere la decisione di indossare oppure no la giacca, quindi con due possibili esiti decisionali. L'albero può essere reso più complesso e preciso, se si considera la velocità del vento, in luogo della semplice presenza/assenza, oppure se si tiene conto di altre variabili dell'ambiente esterno come la presenza del sole o di precipitazioni oppure la stagione dell'anno. Anche in questi casi più complessi, la velocità del vento, o la presenza del sole sono feature, nel significato appreso nel capitolo 9.

Ora ragioniamo in maniera opposta rispetto a quanto fatto sinora: normalmente quando usciamo di casa non facciamo troppi calcoli e decidiamo al volo se indossare una giacca oppure no. Potremmo però fare il seguente esperimento concettuale: quando usciamo di casa, registriamo un campione all'interno di una tabella, per esempio in un file .csv. Ogni campione contiene i valori delle feature come la temperatura esterna, la velocità del vento, la presenza o meno di sole e infine la pura constatazione di aver indossato la giacca o meno.

Dopo qualche tempo la tabella conterrà un buon numero di dati e il nostro esperimento concettuale terminerà e noi ci porremo la domanda: è possibile creare un

In informatica, **albero** si riferisce a uno schema che organizza dati. Gli elementi grafici fondamentali sono: il **nodo**, che in genere contiene informazioni, e l'**arco**, che stabilisce un collegamento gerarchico fra due nodi. L'albero nasce da un nodo detto **radice** e termina su nodi detti **foglia**. Normalmente la sua rappresentazione grafica prevede il nodo radice in alto e i nodi foglia in basso, al contrario dei veri alberi.

albero decisionale che rispecchi al meglio l'algoritmo di scelta giacca/no giacca che inconsciamente adottiamo partendo da queste osservazioni?

La risposta è affermativa e questo significa che il nostro schema decisionale intuitivo può essere formalizzato da un albero decisionale.

La risposta continua a essere affermativa anche in casi più complessi. Gli alberi di decisione e soprattutto i modelli da essi derivati, come il random forest che vedremo a breve, fanno parte degli algoritmi di machine learning che vengono usati in svariate applicazioni dell'intelligenza artificiale.

Gli alberi decisionali possono essere utilizzati sia come **classificatori**, sia come **regressori**.



È possibile convertire **problemi di regressione** in **problemi di classificazione** nel caso in cui sia tollerabile un certo livello di approssimazione per la variabile target. Per esempio, supponendo di avere un problema di regressione in cui la variabile target abbia valori continui nell'intervallo $[-1, 1]$, possiamo trasformare la regressione in un classificatore a due categorie: variabile target **positiva** oppure variabile target **negativa**, dividendo l'intervallo $[-1, 1]$ in due intervalli $[-1, 0]$ e $(0, 1]$ corrispondenti alle due categorie. Questo è possibile nel caso in cui ci basti sapere se la variabile target è positiva oppure no. Se occorre una granularità maggiore, possiamo dividere l'intervallo $[-1, 1]$ in un numero maggiore di intervalli e associare ognuno di essi a una categoria. Nei casi in cui una approssimazione del genere sia tollerabile è possibile trasformare la regressione in una classificazione.

2 Un albero decisionale in Python

Proviamo a concretizzare l'esperimento concettuale descritto nel paragrafo precedente. Scarichiamo il file **registro_utilizzo_giacca.csv** nel quale abbiamo annotato la temperatura esterna, la velocità del vento e se abbiamo indossato o meno la giacca per uscire.

Creiamo un nuovo notebook per analizzare questi dati.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from sklearn import tree
```



registro_utilizzo_giacca.csv

Abbiamo usato le librerie **numpy**, **matplotlib.pyplot**, **pandas** e il modulo **tree** di **scikit-learn** (riga #4). **Tree** è il modulo che implementa gli alberi decisionali: al suo interno sono presenti le classi **DecisionTreeRegressor** e **DecisionTreeClassifier**. Noi abbiamo usato la seconda per affrontare problemi di classificazione come quello che stiamo introducendo.

Una volta importate le librerie necessarie, carichiamo il dataset all'interno di un dataframe di **pandas**.

```
1 data = pd.read_csv("./registro_utilizzo_giacca.csv")
2 data.head()
```

	temperatura	vel_vento	condizione
0	23.7	0.0	no_giacca
1	5.3	7.6	giacca
2	19.8	0.0	no_giacca
3	18.3	0.0	no_giacca
4	12.3	9.0	giacca

La riga #1 del codice legge il file .csv e riconosce automaticamente le diverse colonne e i loro titoli, presenti nella prima riga del file. La riga #2 stampa i primi cinque record del dataframe e ci conferma che i dati sono stati caricati senza intoppi. Come ci aspettiamo, *temperatura* e *vel_vento* hanno valori numerici, mentre *condizione* è di tipo stringa e assume solo due valori possibili: *giacca* e *no_giacca*. Ogni campione è associabile a una di queste due categorie che corrispondono infatti ai due possibili esiti della decisione di indossare una giacca oppure no.

L'obiettivo che ci proponiamo è addestrare un albero decisionale sulla base di questi dati: si tratta di un classificatore poiché l'albero, a partire dalle feature, deve pre-dire una tra due categorie possibili. Controlliamo le statistiche descrittive del dataset.

```
data.describe()
```

	temperatura	vel_vento
count	365.000000	365.000000
mean	20.540548	4.107945
std	6.039244	5.696299
min	5.300000	0.000000
25%	16.300000	0.000000
50%	20.400000	0.400000
75%	24.500000	7.000000
max	37.100000	35.800000

Abbiamo 365 campioni che useremo tutti per addestrare l'albero decisionale: per il momento non ci interessa valutare l'accuratezza del modello sui dati di test, ma sol-tanto capire come viene costruito un albero decisionale a partire da un dataset. Le feature utilizzate sono due: i valori di *temperatura* e *vel_vento*, mentre la variabile target corrisponde ai valori della colonna *condizione*.

Creiamo due array di **numpy** prelevando i valori (**values**) delle colonne di interesse: **X** per la feature e **y** per la variabile target (righe #1 e #2 del codice che segue).

```
1 X = data[["temperatura","vel_vento"]].values
2 y = data["condizione"].values
3
4 albero = tree.DecisionTreeClassifier(max_depth=3, ccp_alpha=0.01)
5 albero.fit(X, y)
```

```
DecisionTreeClassifier(ccp_alpha=0.01, max_depth=3)
```

Alla riga #4 istanziamo un oggetto **tree.DecisionTreeClassifier**, che è proprio il nostro albero decisionale.

- Il parametro **max_depth=3**, indica che la profondità massima dell'albero deve es-sere pari a 3, ovvero che tra il nodo radice e i nodi foglia debbano esserci al massi-mo due nodi e quindi tre archi.
- Il parametro **ccp_alpha=0.01**, serve per gestire la complessità dell'albero otte-nuto tramite operazioni di sfoltimento. Questo parametro ha valore di default pari a 0, che indica all'algoritmo di non effettuare alcuno sfoltimento, ma in questo paragrafo utilizziamo il valore 0,01 per ottenere un albero non troppo complesso e semplice da leggere.

La riga #5 finalmente esegue l'addestramento dell'albero sui dati con il **metodo fit()**.

A questo punto abbiamo un albero decisionale addestrato. Il modulo **tree** di **scikit-learn** contiene una funzione **plot_tree()** che crea una rappresentazio-ne grafica dell'albero decisionale addestrato arricchita di tutti i dettagli per renderla chiaramente leggibile a noi umani. Proviamo!

I nomi dei **metodi** dei modelli di **scikit-learn** sono standard per tutti i modelli del modulo.

```

1 plt.figure(figsize=(20,20))
2 tree.plot_tree(albero, feature_names=["temperatura", "velocità del vento"],
3                 class_names=["giacca", "no giacca"], label="all", filled=True,
4                 proportion=True, rounded=True)
5 plt.show()

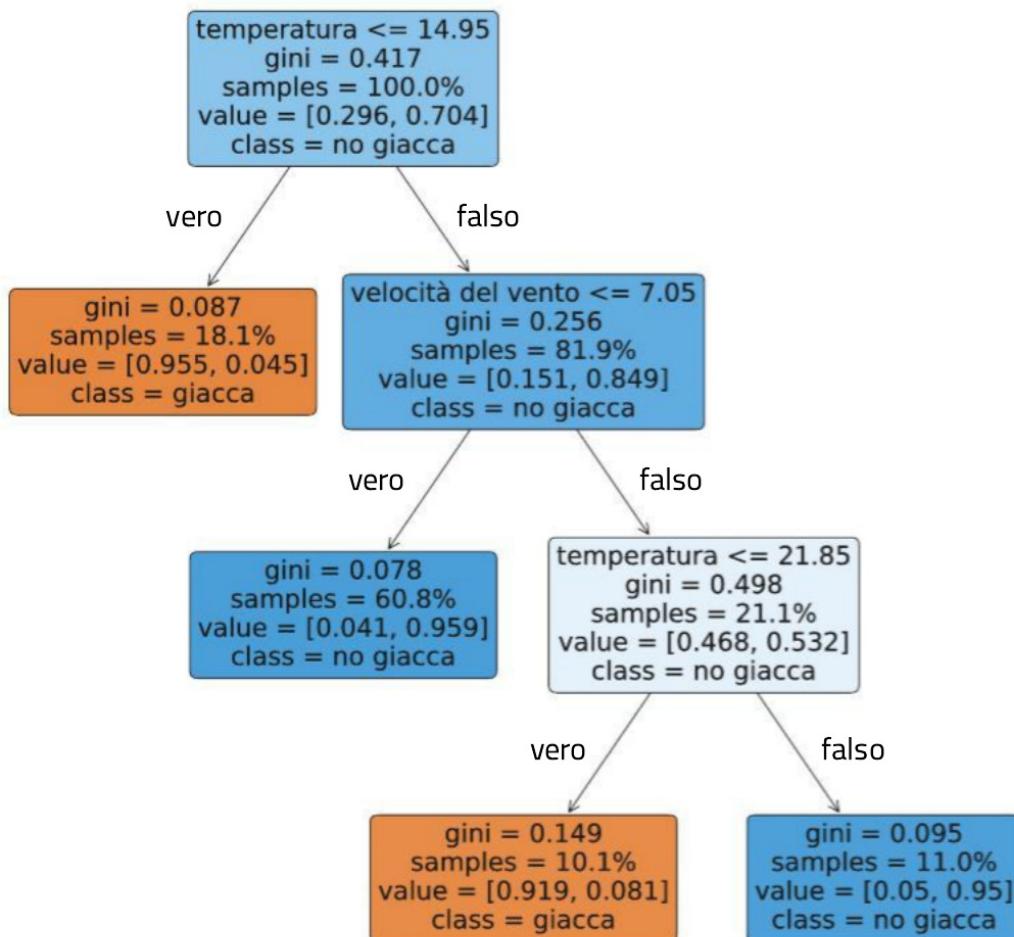
```

Notiamo che si tratta di una figura di **matplotlib** realizzata nella modalità **pyplot** (riga #1), stampata alla riga #5.

La riga #2 genera tale rappresentazione grafica e per farlo ha bisogno di:

- › l'albero decisionale (**albero**);
- › i nomi delle feature, cioè **feature_names= ["temperatura", "velocità del vento"]**;
- › i nomi delle categorie (**class_names= ["giacca", "no giacca"]**);
- › quali grandezze riportare: nel nostro caso tutte (**label="all"**);
- › alcuni parametri che regolano l'estetica: **rounded=True, filled=True**. Nell'ordine creano nodi con bordi arrotondati e li colorano con colori più o meno intensi a seconda della loro purezza, ovvero in base a come si distribuiscono i campioni nelle due categorie. Il nodo sarà bianco se i campioni sono equamente distribuiti, in gradazioni di blu o arancione sempre più intenso, a mano a mano che c'è una predominanza di campioni nell'una o nell'altra categoria;
- › il parametro **proportion=True** fa in modo che le informazioni rispetto al numero di campioni complessivi (**samples**) e a come questi si distribuiscono nelle due categorie (il vettore **value**) sia espresso in percentuale invece che in valore.

Il risultato è la figura sottostante, in cui abbiamo aggiunto, rispetto alla realizzazione di **plot_tree()**, le etichette *vero* e *falso* per facilitarne la comprensione.



La figura deve essere letta dall'alto verso il basso.

Per prima cosa distinguiamo i nodi decisionali dai nodi foglia. I nodi decisionali contengono tutti una domanda sotto forma di una diseguaglianza su di una feature. Nel nostro caso abbiamo tre nodi decisionali, che letti dall'alto verso il basso, corrispondono alle diseguaglianze:

- $\text{temperatura} \leq 14.95$;
- $\text{velocità del vento} \leq 7.05$;
- $\text{temperatura} \leq 21.85$.

I rimanenti quattro nodi (quelli che hanno come prima riga $\text{gini} = \dots$) sono nodi foglia e, come leggiamo all'interno dell'ultima riga di ciascuno, sono associati a una delle possibili categorie di output, quali $\text{class} = \text{giacca}$ oppure $\text{class} = \text{no giacca}$.

Testiamo l'albero supponendo che in questo momento ci sia una temperatura esterna di 20°C e una velocità del vento di 9 m/s . Scorrendo l'albero a partire dal nodo radice in alto dobbiamo affrontare le seguenti condizioni.

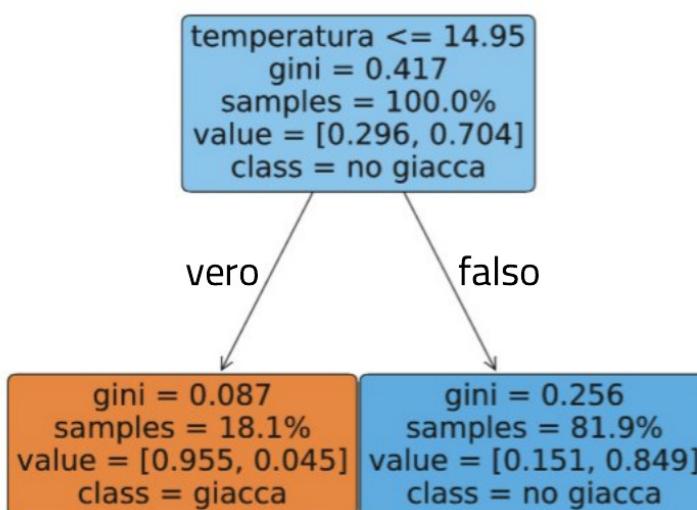
1. La temperatura è maggiore di 14.95°C , quindi seguiamo il ramo falso;
2. Il nodo decisionale raggiunto ci propone la diseguaglianza $\text{velocità del vento} \leq 7.05$, che in questo caso è anch'essa falsa (velocità del vento pari a 9 m/s): nuovamente seguiamo il ramo falso;
3. Giungiamo al nodo decisionale $\text{temperatura} \leq 21.85$ che in questo caso è vero: il ramo vero ci conduce al nodo foglia in basso di colore arancio che riporta $\text{class} = \text{giacca}$.

Quindi è proprio il caso di indossare qualcosa prima di uscire!

Come si addestra un albero decisionale a partire da un dataset?

L'addestramento di un albero decisionale ha il compito di scoprire le diseguaglianze da porre nei nodi decisionali che meglio suddividano i dati tra le loro categorie di appartenenza, partendo dal nodo radice.

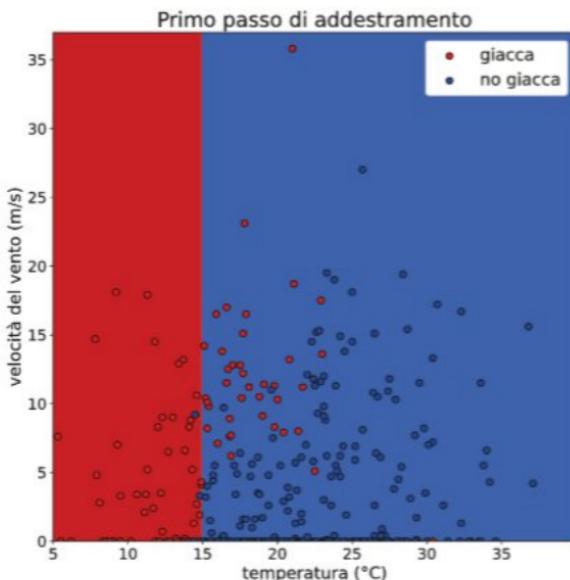
Nel nostro caso il primo passo dell'addestramento porta a creare un primo albero decisionale avente un solo livello.



Questo albero suddivide i dati nelle due categorie possibili, considerando soltanto se $\text{temperatura} \leq 14.95$ oppure no.

Per capire quanto tale suddivisione dei dati sia appropriata, visualizziamola su un piano cartesiano avente come ascissa la temperatura e come ordinata la velocità del vento. I punti rossi e blu nel piano rappresentano i campioni di training suddivisi secondo la loro categoria come indicato in legenda.

Osserviamo il grafico a lato. Il colore di sfondo rappresenta la classificazione prodotta dall'albero decisionale: se la temperatura è minore di 14,95 °C indosso la giacca (area rossa), altrimenti no (area blu). Osserviamo che la suddivisione dei dati non è ottimale poiché l'area blu contiene numerosi punti rossi: si dice che la classificazione effettuata dall'albero per la categoria `no_giacca` è **impura**. L'impurezza è misurata da una grandezza, detta **impurità di Gini**, che assume valori tra 0 e 1. Più l'impurità di Gini per una categoria è bassa, migliore è la classificazione determinata dal nodo decisionale a monte.



Se torniamo alla rappresentazione grafica dell'albero creato da `scikit-learn` possiamo ora interpretare le varie informazioni presenti nei due nodi foglia. Consideriamo la foglia di sinistra: alla riga `gini = ...` troviamo il valore dell'impurità di Gini uguale a 0,087, molto vicino allo 0. Il nodo è quasi puro, infatti i campioni che corrispondono alla domanda `temperatura <= 14,5` sono il 95,5% della categoria `giacca` e il 4,5% dell'altra categoria (`value = [0.955, 0.045]`). Pertanto i campioni sono classificati come categoria `giacca` (`class`) e il colore del nodo è di un bel arancione pieno. Analogamente per la foglia di destra: i campioni che non hanno una `temperatura <= 14,5`, sono l'81,9% dei campioni complessivi (`samples`). Hanno un'impurità di Gini di 0,256, lontana da 0. Infatti in questo nodo foglia il 15,1% sono campioni della categoria `giacca` e il colore di sfondo è un blu non del tutto pieno.

Impurità di Gini: se è dato un nodo, si calcola considerando soltanto i campioni presenti in quel nodo. Supponiamo siano N in totale. Ogni campione di questi N ha una propria categoria di appartenenza (quella annotata nel dataset di addestramento). Supponiamo siano esse A, B, C, \dots e chiamiamo con n_A, n_B, n_C, \dots il numero di campioni delle varie categorie, per cui $n_A + n_B + n_C + \dots = N$.

L'impurità di Gini si calcola come:

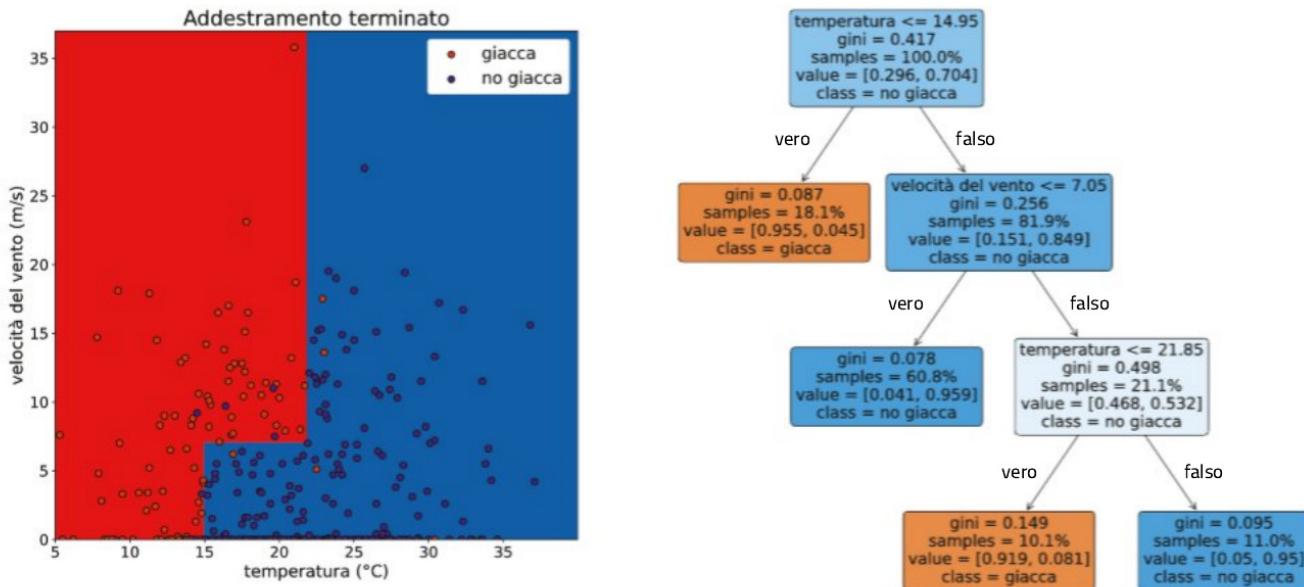
$$I_G = 1 - (n_A/N)^2 - (n_B/N)^2 - (n_C/N)^2 - \dots$$

Noi utilizziamo solo **alberi decisionali binari**: ogni nodo ha al più due nodi successori. In questo caso il valore massimo possibile di impurità di Gini è 0,5, che si ottiene quando i campioni si dividono esattamente a metà fra le due categorie. Un valore uguale a 0 significa che tutti i campioni appartengono alla stessa categoria.

Corrado **Gini** fu uno statistico, economista, sociologo e nel 1926 fondò l'istituto che poi divenne l'Istituto Nazionale di Statistica (ISTAT). Considerò la statistica da un punto di vista interdisciplinare, applicandola a scienze quali la biologia, l'economia e la sociologia.

Quindi possiamo ora affermare che l'addestramento dell'albero avviene trovando le domande da porre nei nodi decisionali che minimizzano l'impurità delle categorie ottenute. Il processo è iterato e ripetuto fintanto che non si ottenga un albero ritenuto soddisfacente, cioè che suddivida i campioni di training in categorie il più possibile pure.

Nel nostro caso l'addestramento termina con un albero, soddisfacente: le foglie sono tutte abbastanza pure. Come vediamo nella pagina seguente i tre nodi decisionali dell'albero definiscono esattamente la divisione del piano cartesiano.



3 Gli alberi decisionali non bastano

L'esempio appena visto è su un dataset piuttosto semplice, ma quanto abbiamo imparato può essere applicato a un qualunque dataset.

Un grande pregio degli alberi decisionali, tra gli algoritmi di machine learning, è che essi sono modelli **white box**. Contrariamente ai modelli **black box**, come le reti neurali, gli alberi sono perfettamente comprensibili e interpretabili da noi umani: addirittura possono essere visualizzati graficamente e letti dall'alto verso il basso. Una volta che abbiamo addestrato un albero possiamo valutarlo e verificarne la ragionevolezza semplicemente interpretandolo.

Un secondo vantaggio, che ci semplifica il lavoro di preparazione dei dati, è che non occorre standardizzare i dati prima di procedere all'addestramento. Questo poiché la determinazione della disuguaglianza $feature \leq x$ presente in ogni nodo decisionale è adattata automaticamente all'intervallo di valori della feature.

Il terzo vantaggio riguarda la **complessità di tempo** dell'algoritmo: essa è proporzionale al logaritmo del numero di campioni utilizzati per addestrare l'albero (in formula, $O(\log(n))$), quindi aumenta molto lentamente all'aumentare del numero di dati, per cui gli alberi sono ritenuti molto efficienti dal punto di vista computazionale.

Black box significa scatola nera, cioè una scatola nella quale non si può guardare. Algoritmi come le reti neurali a molti strati sono considerate scatole nere poiché è molto difficile, se non impossibile, motivare le predizioni della rete a partire dal comportamento di ogni singolo neurone.

La **complessità di tempo** è una misura del massimo tempo di esecuzione di un algoritmo espresso in funzione della dimensione n dei dati in ingresso all'algoritmo. Solitamente è indicata con la notazione $O()$. Alcuni esempi di complessità di tempo sono:

- $O(1)$ se il tempo impiegato dall'algoritmo non dipende dalla dimensione dei dati;
- $O(n)$ se il tempo cresce linearmente con la dimensione dei dati;
- $O(n^2)$ se il tempo cresce quadraticamente con la dimensione dei dati;
- $O(\log(n))$ se il tempo cresce come il logaritmo della dimensione dei dati.

Non dobbiamo però lasciarci illudere da questi vantaggi, poiché gli svantaggi sono tali da rendere gli alberi decisionali poco utilizzabili nei casi reali. Il più importante è la loro predisposizione a fare overfitting: l'addestramento di un albero decisionale porta spesso ad alberi molto complessi che non generalizzano i dati.

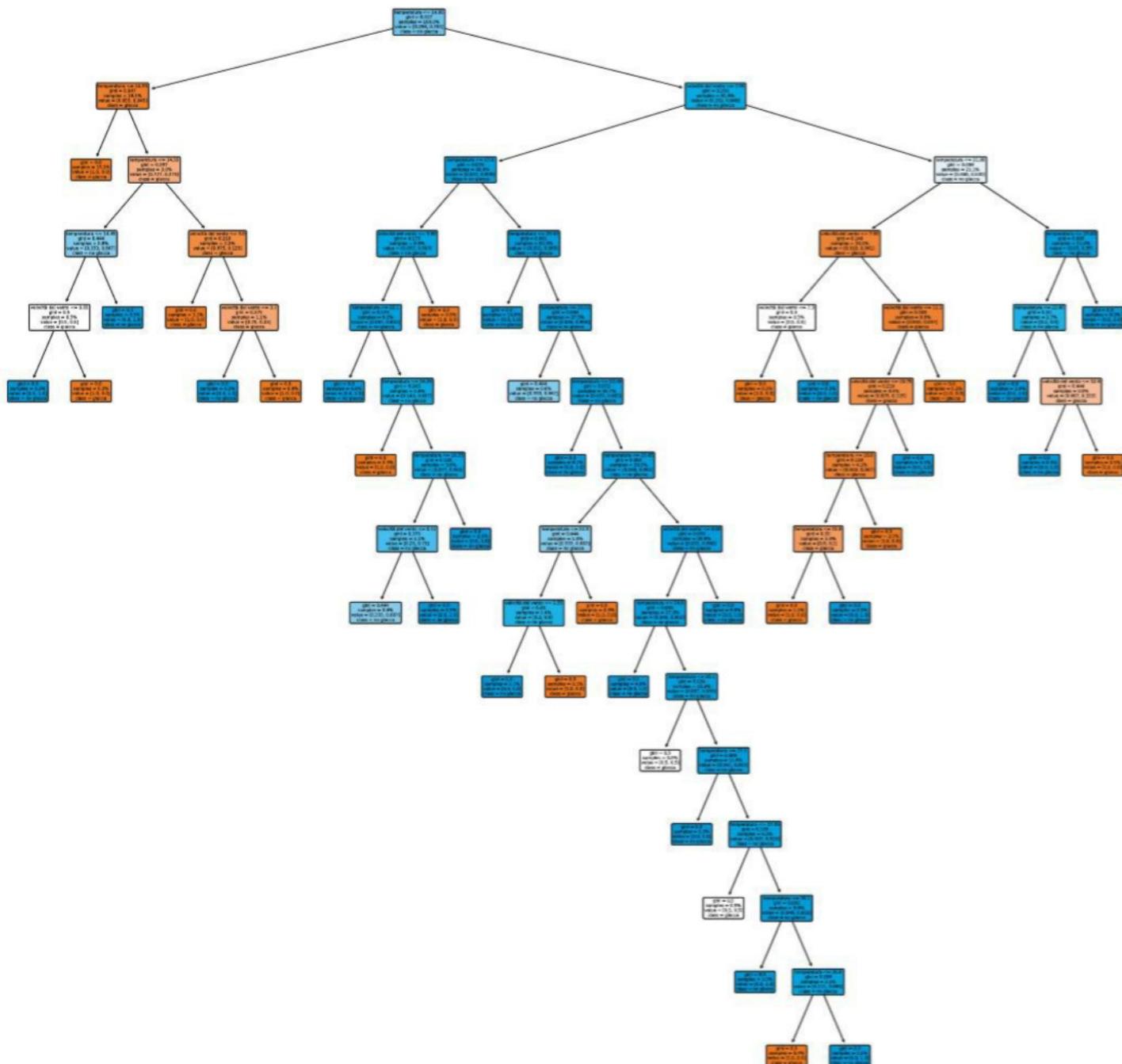
Per esempio riaddestriamo l'albero del paragrafo precedente e usiamo gli stessi dati. Inoltre non curiamoci della configurazione, alla riga #1, dei parametri nell'istanza di `tree.DecisionTreeClassifier` (riga #1).

```

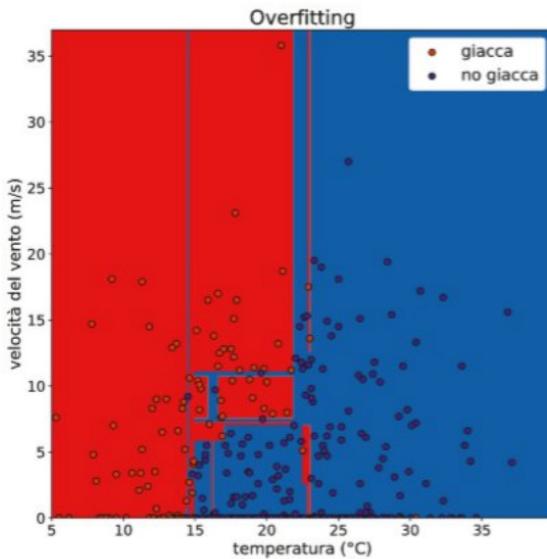
1 albero = tree.DecisionTreeClassifier()
2 albero.fit(X, y)
3 plt.figure(figsize=(20,20))
4 tree.plot_tree(albero, feature_names=["temperatura", "velocità del vento"],
5                 class_names=["giacca", "no giacca"], label="all", filled=True,
6                 proportion=True, rounded=True)
7 plt.show()

```

Ci rendiamo conto a colpo d'occhio che il risultato ottenuto (qui sotto) ha una complessità non giustificata dai dati.



Inoltre osserviamo come il risultato, mostrato qui sotto, di una classificazione con questo albero sia tradotto nel piano cartesiano delle due feature.



Notiamo varie aree blu o rosse molto sottili, che sono il segno distintivo dell'overfitting: punti blu isolati nell'area rossa oppure punti rossi isolati nell'area blu inducono l'albero a creare classificazioni per questi particolari punti, che non sono giustificabili dalla distribuzione complessiva dei campioni. Gli alberi decisionali sono inoltre instabili: alterare pochi punti in un dataset può portare alla modifica dell'intero albero decisionale e questo inficia la validità del modello realizzato. Ricordiamo che un modello di machine learning deve essere robusto contro piccole variazioni del dataset di training. Infine gli alberi sono condizionati da categorie non bilanciate: immaginiamo di addestrare l'albero su un dataset con una categoria molto più numerosa della seconda. In tale situazione può accadere che l'albero dia sempre priorità alla categoria più numerosa anche quando non è corretto farlo secondo la reale distribuzione dei campioni.

In sintesi, i vantaggi degli alberi decisionali non sono sufficienti per renderli un algoritmo di machine learning sufficientemente flessibile per applicazioni in situazioni generali. Essi sono però l'ingrediente fondamentale di uno dei principali algoritmi di machine learning supervisionato alternativo alle reti neurali: le foreste casuali.

4 Le foreste casuali

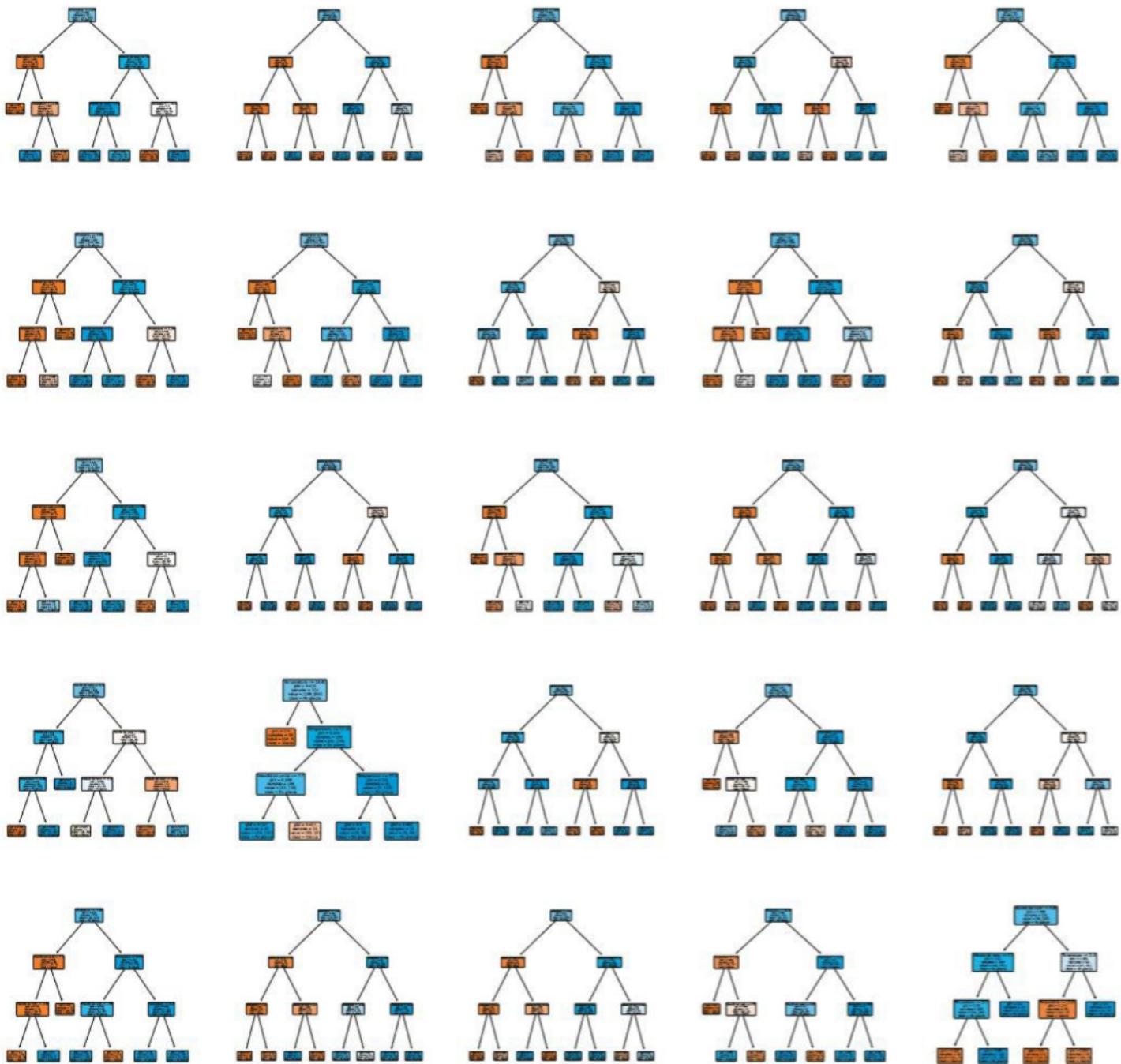
Le **foreste casuali**, **random forest** nel linguaggio comune del machine learning, sono insiemi di alberi, proprio come le vere foreste del nostro pianeta. Ma che cosa vuol dire la parola *insieme* quando si sta parlando di modelli di intelligenza artificiale?

Esiste un intero paradigma del machine learning, detto **ensemble learning** (in italiano apprendimento di insieme) in cui numerosi algoritmi semplici detti **base estimators**, sono addestrati e combinati insieme per risolvere lo stesso problema. Il principio è analogo a quello richiamato dagli esperimenti di **saggezza della folla**: chiediamo a tutti i nostri compagni di classe una stima per il numero di alunni presenti nell'intera scuola, facciamo la media di tutte le stime e scopriamo che il valore ottenuto non è troppo distante dalla realtà.

Lo stesso principio si applica ai modelli di machine learning: nel nostro caso i base estimators sono gli alberi decisionali e una moltitudine di essi costituisce una foresta.

Secondo la teoria della **saggezza della folla** una massa di inesperti può rispondere a una domanda più correttamente di un esperto.

Nella figura sotto, per esempio, abbiamo rappresentato graficamente una random forest composta da 25 alberi decisionali.



Le foreste casuali sono un esempio di algoritmo che segue il cosiddetto **paradigma dell'ensemble learning**. Questo significa che la predizione di una foresta sarà ottenuta mediando tra le predizioni di tutti gli alberi che la compongono. Nel caso dei classificatori, che stiamo trattando, ci sono due modi per ottenere la predizione:

- la foresta sceglie la categoria più predetta tra le predizioni di tutti gli alberi, è come se effettuasse una votazione tra alberi;
- oppure la foresta, per ogni categoria, calcola la media tra le probabilità per quella categoria generate da ogni albero e sceglie la categoria più probabile.

L'implementazione di **scikit-learn** utilizza la seconda modalità.

Vediamo che non ci serve troppa immaginazione per comprendere il perché questi modelli di machine learning siano stati chiamati foreste, ma dobbiamo ancora approfondire il motivo dell'aggettivo *casuale*: esso risiede nei criteri di generazione dei singoli alberi che compongono la foresta casuale. Questi criteri sono due:

- 1.** ciascun albero è generato facendo l'addestramento su un sottoinsieme dei dati di training estratto a caso. L'estrazione è effettuata con ripetizione, ovvero i campioni estratti possono essere estratti nuovamente anche per addestrare l'albero successivo. Questa metodologia di estrazione del dataset di training si chiama **bagging**. Ogni albero è addestrato con dati parzialmente diversi scelti in modo casuale;
- 2.** ogni albero della foresta è addestrato utilizzando un sottoinsieme casuale delle feature disponibili: sia il numero di feature sia le feature stesse cambiano in maniera casuale da albero ad albero.

La casualità introdotta da questi due criteri porta ad avere una foresta di alberi diversi tra loro. La diversità ha un grande valore aggiunto poiché diminuisce l'incertezza nel modello ensemble. Facendo una media delle predizioni di molto alberi, ciascuno dei quali commette errori, accade che molti errori si elidano tra loro: questo limita i problemi dell'overfitting e della instabilità, descritti nel paragrafo precedente. Possiamo parlare di saggezza delle foreste, visto che la foresta riesce ad apprendere meglio dei singoli alberi grazie alla **moltitudine** e alla **diversità**.

La prima implementazione delle foreste casuali risale al 2001 a opera dello statistico Leo Breiman, che in una sua pubblicazione mostrò come i random forest fossero utilizzabili per risolvere sia problemi di classificazione, sia di regressione.

Ora siamo pronti per mettere alla prova la saggezza delle foreste!

Il **bagging** è una tecnica di estrazione del dataset e non è l'unica. Nel caso in cui l'estrazione sia effettuata senza ripetizione, si parla di **pasting**.

5 Un'applicazione reale: i pinguini dell'arcipelago di Palmer

Spostiamoci nell'arcipelago di Palmer in Antartide, presso la Palmer station Antartica LTER, dove la biologa marina Gorman, come abbiamo scoperto nel capitolo 8, ha raccolto i dati delle tre specie di pinguini, Adelie, Chinstrap e Gentoo, e pubblicati in Horst, Hill, Gorman, *Palmerpenguins: Palmer Archipelago (Antarctica) penguin data* (2020). Gorman ha registrato i seguenti dati per 344 pinguini:

- la specie di appartenenza: Adelie, Chinstrap oppure Gentoo;
- l'isola sulla quale è avvenuto l'incontro;
- la lunghezza del becco in mm (come mostrato nella figura qui sotto);
- la profondità del becco in mm (come mostrato nella figura qui sotto);
- la lunghezza della pinna in mm;
- il peso corporeo in g;
- il sesso.

