Argomento: Introduzione al Machine Learning

Corso: Machine Learning per le applicazioni ambientali

Descrizione della lezione:

Questa lezione introduttiva fornisce una panoramica di base del machine learning (ML) e del suo ruolo nella soluzione di problemi ambientali ed ecologici. Presenta i principali compiti del ML (classificazione, regressione e clustering) e le strategie di apprendimento (supervisionato e non supervisionato). Attraverso esempi visivi chiari, si concentra su concetti fondamentali quali la preparazione dei dati, la costruzione delle caratteristiche (feature engineering), l'addestramento e la valutazione dei modelli. Vengono approfonditi i problemi di underfitting e overfitting e le relative conseguenze. Le applicazioni ambientali comprendono la classificazione dei livelli di qualità dell'aria e il rilevamento degli habitat.

Obiettivi di apprendimento:

Al termine di questa unità, gli studenti saranno in grado di:

1. Spiegare che cos'è il machine learning e descriverne le principali applicazioni nelle scienze ambientali.
2. Distinguere tra apprendimento supervisionato e non supervisionato.
3. Riconoscere i principali compiti di machine learning, come regressione, classificazione e clustering.
4. Comprendere il flusso di lavoro di base di un progetto di ML (preparazione dei dati, addestramento e valutazione).
5. Riconoscere e spiegare problemi comuni nell'apprendimento dei modelli, come overfitting, underfitting e problemi di qualità dei dati.
6. Addestrare e valutare un semplice modello di ML utilizzando dati ambientali.

Introduzione

Il machine learning (ML) permette ai computer di apprendere dai dati senza richiedere una programmazione esplicita per ogni compito. Negli ultimi anni ha trovato ampia applicazione nella soluzione di problemi ambientali, affiancando o, in alcuni casi, sostituendo gli approcci tradizionali di modellazione ambientale, basati principalmente su equazioni fisiche e chimiche e sulla loro integrazione con formalismi della biologia, della geologia e dell'ecologia. Con l'aumento del volume e della varietà dei dati ambientali (sensori, dispositivi IoT e sistemi di monitoraggio su larga scala), l'attenzione si è spostata verso metodi guidati dai dati, capaci di apprendere direttamente dalle osservazioni raccolte.

Il machine learning è una disciplina basata sui dati e uno dei settori più importanti dell'intelligenza artificiale. Anziché imporre una struttura fissa del modello, gli algoritmi di ML ricavano dagli esempi le relazioni sottostanti, consentendo a ricercatori e professionisti di modellare sistemi nei quali le leggi fisiche esplicite sono sconosciute, incomplete o troppo complesse.

I fenomeni ambientali (come i processi climatici, il degrado degli ecosistemi e le dinamiche dell'inquinamento) sono per loro natura problemi complessi, non lineari e multidimensionali. Il machine learning offre numerosi strumenti e approcci per affrontarli. Inoltre, grazie alla disponibilità di dati ricchi e articolati, questi problemi possono spesso essere trattati più efficacemente con il ML rispetto ai tradizionali modelli di regressione o simulazione.

Alcuni esempi di applicazioni del ML nelle scienze ambientali:

- Previsione climatica e meteorologica: i modelli di ML individuano regolarità nelle serie meteorologiche di lungo periodo e nelle osservazioni satellitari, al fine di prevedere anomalie di temperatura o eventi estremi di precipitazione.[1]
- Modellazione della qualità dell'aria e dell'acqua: i modelli predittivi apprendono le correlazioni tra emissioni, variabili meteorologiche e concentrazioni di inquinanti. [2]
- Telerilevamento e classificazione dell'uso del suolo: i modelli di ML basati su immagini classificano automaticamente i tipi di copertura del suolo a partire da dati satellitari multispettrali.[3]
- Monitoraggio della biodiversità e degli ecosistemi: il ML supporta la modellazione della distribuzione delle specie, la classificazione acustica e il rilevamento di anomalie ecologiche.[4]

Abbiamo già osservato che il machine learning consente a un sistema di apprendere dai dati senza dover programmare esplicitamente il compito da svolgere.

> *«Si dice che un programma apprende dall'esperienza E rispetto a una classe di compiti T e a una misura di prestazione P se la sua prestazione nei compiti T, misurata mediante P, migliora con l'esperienza E».[5]*

Questa definizione evidenzia tre componenti essenziali:

- Dati (esperienza E): la base dalla quale il modello apprende le regolarità.
- Compito (T): ciò che il modello deve svolgere, ad esempio classificare, prevedere o raggruppare.
- Prestazione (P): una misura quantitativa della qualità dei risultati del modello (accuracy, RMSE, F1-score, ecc.).

Esempio: vogliamo sviluppare un'applicazione che riconosca le automobili nelle immagini. Potremmo provare a codificare esplicitamente una serie di regole (SE–ALLORA) per descrivere un'automobile, ad esempio:

SE un'immagine mostra quattro ruote, una forma rettangolare e un parabrezza, ALLORA contiene un'AUTOMOBILE; altrimenti NON contiene un'AUTOMOBILE.

Questo approccio fallisce rapidamente: le automobili hanno aspetti diversi secondo il punto di vista, l'illuminazione influenza il riconoscimento e alcuni altri oggetti possono assomigliare ad automobili. Le regole risultano quindi troppo rigide e non producono risultati soddisfacenti.

Un approccio basato sul machine learning può risolvere meglio questo tipo di problema, a condizione di disporre di un numero sufficientemente ampio di esempi.
Decidiamo quindi di raccogliere un insieme di immagini di automobili in situazioni differenti e un numero analogo di immagini prive di automobili. Scegliamo un modello di ML adatto al riconoscimento delle immagini e assegniamo a ciascuna immagine un'etichetta corretta (AUTOMOBILE/NON AUTOMOBILE). Il compito del modello è prevedere l'etichetta corretta a partire dall'immagine. Il modello apprende dagli esempi, cioè acquisisce esperienza osservando i dati. Inizialmente commetterà molti errori, ma, con l'aumentare degli esempi, imparerà a prevedere correttamente le etichette nella maggior parte dei casi. La percentuale di immagini classificate correttamente costituisce una misura della prestazione e indica quanto bene il modello abbia appreso il compito.

Diciamo che il modello apprende perché la sua capacità di riconoscere le automobili nelle immagini migliora con l'esperienza acquisita dai dati!

Un tipico flusso di lavoro di ML comprende:

1. Raccolta e preelaborazione dei dati
2. Costruzione e rappresentazione delle caratteristiche (feature engineering)
3. Selezione e addestramento del modello
4. Valutazione e validazione
5. Interpretazione e messa in produzione

Oltre a fornire buone prestazioni predittive nelle scienze ambientali, il ML costituisce la base di sistemi decisionali di qualità in condizioni di incertezza, offrendo indicazioni interpretabili sull'importanza delle caratteristiche, sulle dipendenze spaziali e sulla sensibilità del sistema. Con la continua crescita dei dati ambientali, la conoscenza del ML diventa una componente indispensabile della moderna gestione dei dati ambientali [6][7][8][9][10],

I principi fondamentali del machine learning

Le scienze ambientali dispongono di una grande quantità di dati provenienti da fonti eterogenee ed è quindi naturale sfruttare le potenzialità dei modelli di ML in questo ambito. Il ML integra gli approcci scientifici tradizionali identificando schemi nascosti e ricavando correlazioni. In questa lezione introduttiva non approfondiremo i singoli modelli utilizzati nei diversi compiti di ML, che saranno trattati nelle lezioni successive, ma i principi fondamentali comuni a tutti. L'obiettivo principale è la comprensione concettuale, non il formalismo matematico alla base del funzionamento dei modelli.

Apprendimento supervisionato e non supervisionato

Dalla definizione di machine learning sappiamo che un modello può imparare a svolgere un compito a partire dai dati. Esistono due categorie principali: apprendimento supervisionato e non supervisionato. Nell'apprendimento supervisionato il modello viene addestrato su dati per i quali sono noti sia gli input sia gli output corrispondenti. Si consideri, per esempio, la classificazione di immagini satellitari in otto categorie di copertura del suolo: aree edificate, strade, corpi idrici, terreni agricoli, arbusteti, foreste, prati e altre categorie. Le etichette sono fornite da esperti del dominio: per ogni immagine del dataset conosciamo già la categoria corretta, mentre il modello deve imparare a prevederla. Mediante una misura appropriata delle prestazioni, il modello apprende a riconoscere tali categorie sia nelle immagini etichettate sia, soprattutto, in nuove immagini mai viste. Il termine «supervisionato» si riferisce quindi alle annotazioni fornite dagli esperti, che costituiscono il riferimento corretto (ground truth). Un altro tipico compito supervisionato è la regressione, nel quale il modello apprende da input associati a valori numerici noti. Un esempio consiste nel prevedere le concentrazioni di azoto nitrico e cloruri da variabili indipendenti come portata media giornaliera, precipitazioni nelle 24 ore precedenti, posizione del sito di campionamento, copertura colturale, temperatura, pH, ossigeno disciolto, potenziale redox e conducibilità elettrica. Le variabili dipendenti sono indicatori della qualità delle acque superficiali nei bacini agricoli. In questo caso la supervisione è indiretta, perché deriva dalle variabili indipendenti e dipendenti osservate [11][12]

Nell'apprendimento non supervisionato, invece, per gli input osservati non è noto a priori alcun output di riferimento: sono disponibili soltanto gli input. Il modello deve identificare regolarità e strutture significative nei dati, interpretabili nel dominio applicativo. Non esiste quindi un supervisore che fornisca etichette o valori corretti; per questo si parla di apprendimento non supervisionato.

Un compito tipico dell'apprendimento non supervisionato è il clustering, che mira a riunire nello stesso gruppo le osservazioni simili, separando quelle dissimili. Quando ogni osservazione appartiene a un solo gruppo si parla di clustering rigido (crisp clustering); quando può appartenere a più gruppi con differenti gradi di appartenenza si parla di clustering sfumato (soft clustering). Nelle scienze ambientali il clustering svolge un ruolo importante, ad esempio nella valutazione della qualità delle acque sotterranee.[13]

Esistono numerosi modelli di ML applicabili ai compiti descritti:

- Classificazione (SVM – macchine a vettori di supporto, alberi decisionali, RIPPER, K-NN, reti neurali artificiali, regressione logistica, Random Forest, Naive Bayes, AdaBoost, Bagging, Stacking, ecc.)
- Regressione (regressione lineare, SVM per la regressione, alberi decisionali per la regressione, percettrone multistrato, ecc.)
- Clustering (K-means, Expectation-Maximization, DBSCAN, HDBSCAN, Cobweb, clustering gerarchico, Fuzzy C-means, ecc.)

Molti di questi modelli saranno approfonditi nelle lezioni successive. Qui ne introduciamo soltanto uno, utile per spiegare concetti fondamentali quali preparazione dei dati, feature engineering, addestramento, test, valutazione delle prestazioni e fenomeni di overfitting e underfitting. Presenteremo inoltre brevemente Google Colab.

Regressione logistica – Concetti fondamentali

Nonostante il nome, la regressione logistica è un modello di classificazione, utilizzato soprattutto per problemi di classificazione binaria, cioè con due classi, tipicamente indicate con 0 e 1.

Il suo obiettivo non è prevedere un valore continuo, ma stimare la probabilità che un esempio appartenga a una determinata classe (generalmente la classe 1).

Formalmente, il modello rappresenta:

$P(y=1;x,\theta)$Conoscendo questa probabilità, si conosce anche quella dell'altra classe:

$$P(y=0;x,\theta)=1-P(y=1;x,\theta)$$

Dove:

- **x è un vettore di caratteristiche**
- y (0 oppure 1) è l'etichetta di classe
- θ sono i parametri (pesi) del modello

Alla base della regressione logistica si trova la funzione sigmoide (o logistica), che trasforma qualsiasi valore reale in un numero nell'intervallo (0, 1):

$$h(x)=\frac{1}{1+e^{-\theta^T \cdot x}}$$

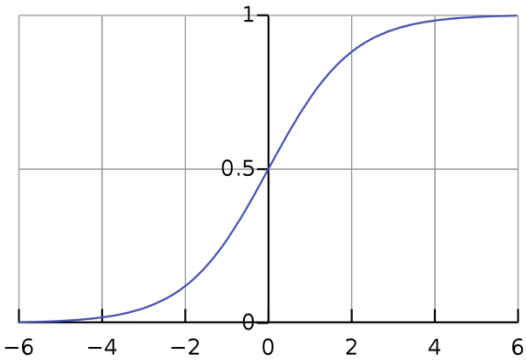


Figura 1 Funzione sigmoide (logistica) (Fonte: https://commons.wikimedia.org/wiki/File:Logistic-curve.svg)1

Questo è essenziale perché:

- Le probabilità devono appartenere all'intervallo tra 0 e 1.
- La sigmoide trasforma in modo continuo una combinazione lineare delle caratteristiche in una probabilità.$\theta^T \cdot x$

L'interpretazione riguarda quindi le probabilità e non soltanto le etichette. La regressione logistica produce un risultato numerico:

- $P(y=1 \mid x)$: probabilità della classe positiva
- $P(y=0 \mid x)=1-P(y=1 \mid x$: probabilità della classe negativa

La regressione logistica è particolarmente utile quando:

- le classi sono sbilanciate;
- occorre modificare le soglie decisionali;
- sono importanti l'ordinamento delle previsioni o la stima della confidenza.

Sorgono due domande principali:

- Come vengono determinati i pesi del modello?θ
- Quale forma assume la frontiera di decisione del modello?

$$h(x)=¿\ \frac{1}{1+e^{-(\theta_0+\theta_1 \cdot x_1+\theta_2 \cdot x_2)}}$$

$$h(x)=¿\ \frac{1}{1+e^{-(\theta_0+\theta_1 \cdot x_1^2+\theta_2 \cdot x_2^2)}}$$

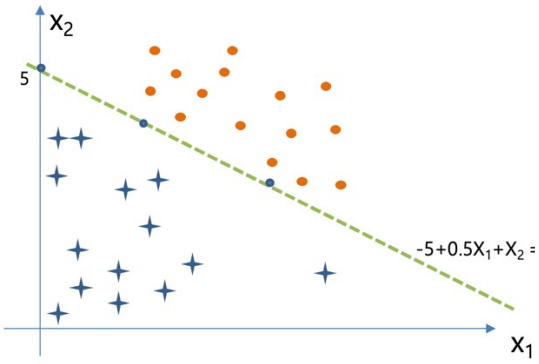


Figure 2 Linear decision boundary

Decission → all above line is assigned to class 1, otherwise to class 0

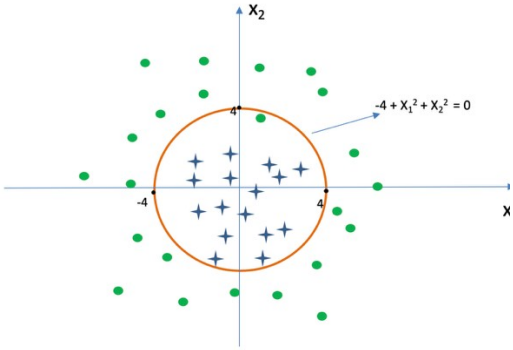


Figure 3 Nonlinear decision boundary

Decission → all out of circle is assigned to class 1, otherwise to class 0

Nel primo caso, illustrato nella Figura 2, la regressione logistica produce una frontiera di decisione lineare nello spazio originale delle caratteristiche. Nel secondo caso (Figura 3) si ottiene una frontiera non lineare, ma soltanto dopo aver trasformato le caratteristiche iniziali mediante feature engineering (per esempio introducendo termini quadratici); il modello resta comunque lineare rispetto ai parametri. I pesi vengono appresi dai dati ottimizzando una misura di prestazione prescelta.Figure 2Figure 3θ

In questa lezione non approfondiamo le misure di prestazione, che saranno illustrate nelle lezioni successive.

Accuratezza (accuracy)

L'accuratezza misura la proporzione di previsioni corrette sul totale delle previsioni.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

L'accuratezza risponde alla domanda: "Con quale frequenza il modello è corretto, nel complesso?"

Precisione (precision)

La precisione indica quale proporzione degli esempi previsti come positivi sia effettivamente positiva.

$$P = \frac{TP}{TP + FP}$$

La precisione risponde alla domanda: “Quando il modello prevede un caso positivo, quante volte ha ragione?”

Recall (sensibilità)

Il recall misura quale proporzione degli esempi realmente positivi venga identificata correttamente dal modello.

$$R = \frac{TP}{TP + FN}$$

Un recall elevato è fondamentale quando non rilevare un evento ambientale, come il superamento di una soglia di inquinamento o la presenza di un habitat, comporta rischi significativi.

F1-score

L’F1-score combina precision e recall in un’unica misura mediante la loro media armonica.

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

L’F1-score è utile quando sono importanti sia i falsi positivi sia i falsi negativi e quando il dataset è sbilanciato, come accade spesso nelle applicazioni ambientali.

Curva ROC

La curva ROC (Receiver Operating Characteristic) rappresenta il tasso di veri positivi (recall) rispetto al tasso di falsi positivi, al variare della soglia di classificazione.

$$FPR = \frac{FP}{FP + TN}$$

La curva ROC illustra il compromesso tra l'individuazione dei veri eventi ambientali e la frequenza dei falsi allarmi.

ROC-AUC

La ROC-AUC (area sotto la curva ROC) riassume la curva ROC mediante un valore compreso tra 0 e 1.

Interpretazione:

- 0,5 indica una classificazione equivalente al caso
- 1,0 indica una discriminazione perfetta

La ROC-AUC misura la capacità del modello di assegnare punteggi più alti ai casi positivi rispetto a quelli negativi, indipendentemente dalla specifica soglia decisionale.

Curva Precision–Recall (PR)

La curva Precision–Recall (PR) rappresenta la precision rispetto al recall per differenti soglie di classificazione.

La curva PR mostra come la precision cambi al crescere del recall, evidenziando il compromesso tra falsi allarmi e casi non rilevati.

PR-AUC (Average Precision)

La PR-AUC riassume la curva Precision–Recall con un singolo valore. È particolarmente informativa per dataset ambientali nei quali i casi positivi sono rari, ad esempio superamenti di soglie di inquinamento o tipi specifici di habitat. Rispetto alla ROC-AUC risente maggiormente dei falsi positivi.

Concetti chiave

Non esiste una metrica universalmente migliore. Nel machine learning ambientale la scelta deve riflettere le priorità applicative:

• Usare il recall quando perdere eventi reali comporta rischi

• Usare la precision quando i falsi allarmi sono costosi

• Usare l'F1-score per bilanciare entrambi

• Preferire PR-AUC e curve PR per il rilevamento di eventi rari

• Usare la ROC-AUC per confrontare i modelli in generale

Google Colab

Google Colab è una piattaforma cloud che offre gratuitamente, tramite browser, accesso a risorse di calcolo di Google, comprese CPU, GPU e TPU. Basata su Jupyter Notebook, permette di scrivere ed eseguire codice Python senza configurare un ambiente locale. Include numerose librerie per il machine learning e si integra con Google Drive; per utilizzarla è necessario un account Google.

Google Colab è uno dei modi più semplici per eseguire script Python senza dover installare le librerie necessarie. Offre un ambiente intuitivo per eseguire anche programmi complessi, con un interprete Python e moduli già disponibili. Queste librerie semplificano lo sviluppo e comprendono strumenti per il calcolo numerico (NumPy), l'analisi dei dati (Pandas), la visualizzazione (Matplotlib) e il machine learning (Scikit-learn, TensorFlow e PyTorch).

Per utilizzare Colab occorre un account Google. Se non ne possiedi uno, puoi crearlo tramite il collegamento indicato. Dopo l'accesso, apri Colab: dovrebbe comparire una schermata simile alla seguente.herehere

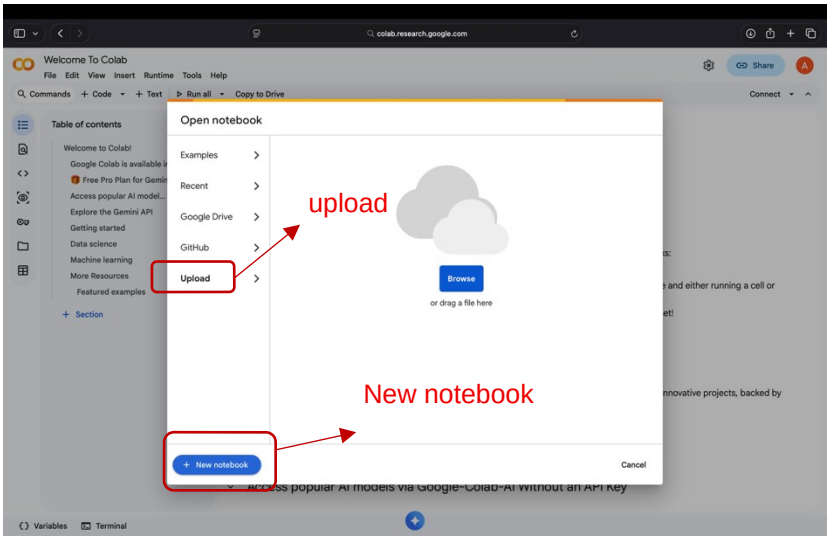


Figura 4 Finestra iniziale di Google Colab4

Da questa finestra è possibile creare un nuovo notebook, caricarne uno già preparato o aprire un notebook esistente. Le stesse operazioni sono disponibili nel menu File. Per ora apri un nuovo notebook.

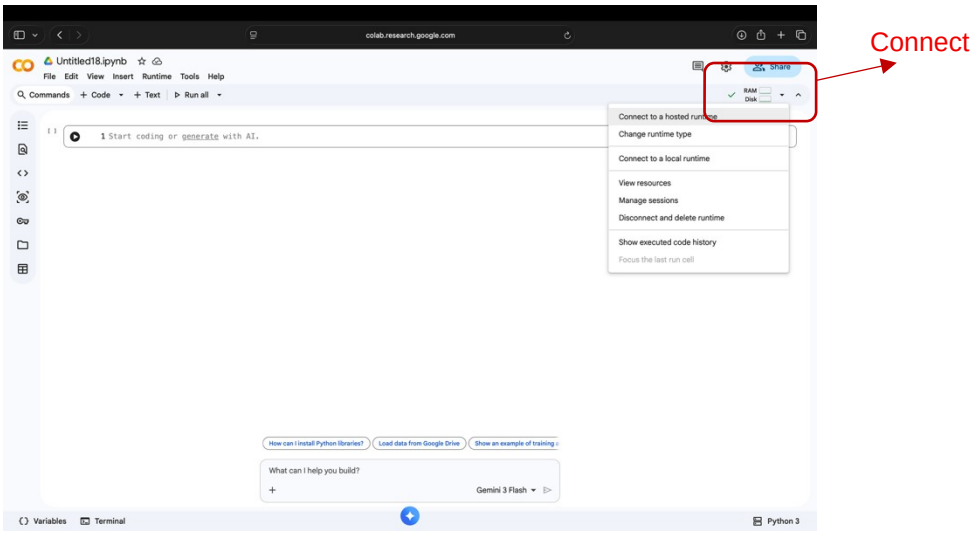


Figura 5 Avvio di un nuovo notebook Google Colab5

Nella Figura 5, in alto a destra, fai clic sulla freccia e scegli la connessione a un ambiente di esecuzione ospitato. Dopo la connessione, l'indicatore mostrerà la RAM, la CPU e un segno di spunta verde. Sono disponibili due tipi di celle: celle di codice e celle di testo (Markdown). Per eseguire Python si utilizzano le celle di codice. Una cella si esegue facendo clic sul pulsante di riproduzione alla sua sinistra oppure premendo Maiusc+Invio. Le celle di testo si modificano con un doppio clic; l'anteprima appare accanto al testo e il risultato viene visualizzato quando si esce dalla cella. Per la formattazione si può consultare il tutorial Markdown [markdown tutorial.] Per rinominare il notebook (Figura 6) basta modificare il nome in alto a sinistra, conservando l'estensione .ipynb.

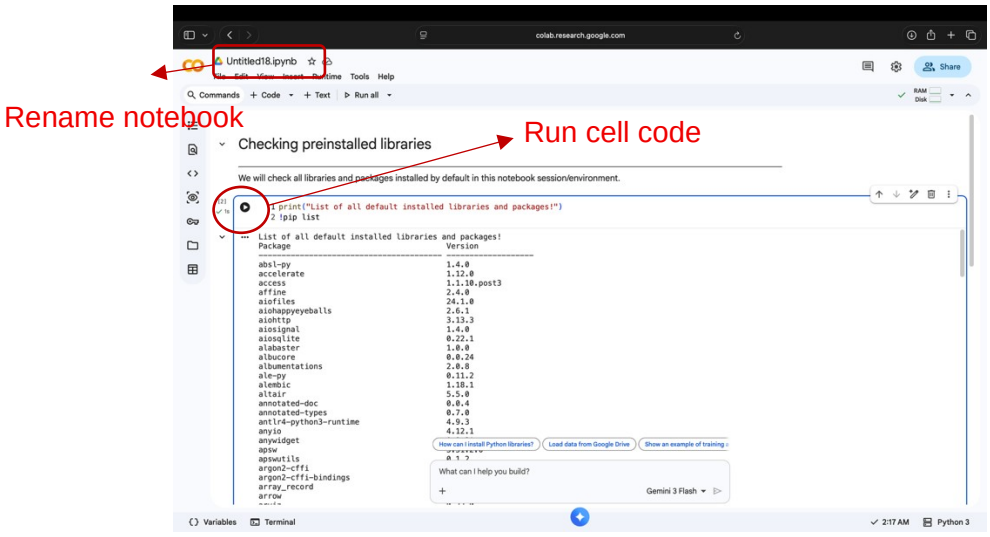


Figura 6 Esecuzione di codice Python in Colab e ridenominazione del notebook6

Oltre a salvare il notebook su Google Drive o GitHub, è possibile condividerlo tramite il pulsante Condividi in alto a destra oppure scaricarlo in formato .py o .ipynb dal menu File, scegliendo l'opzione di download (Figura 7).Figura 7 Download e condivisione del notebook7

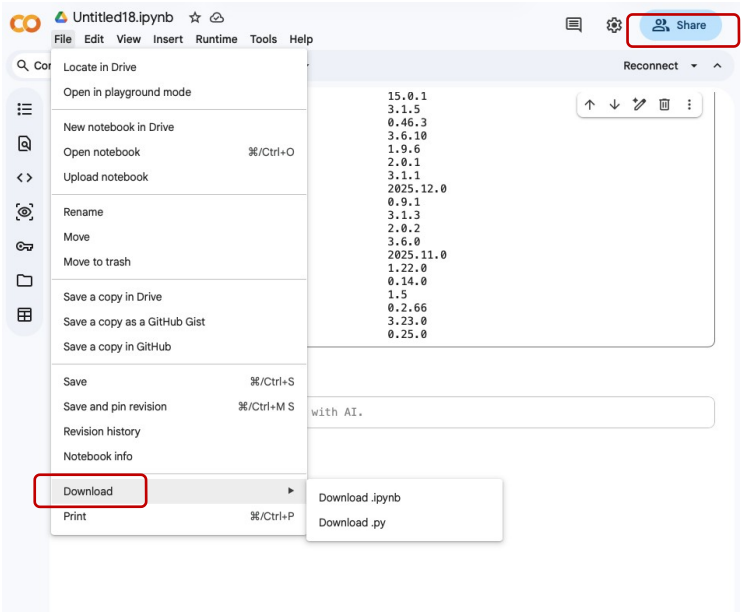


Figura 7 Download e condivisione del notebook7

Previsione del superamento della soglia di NO_2 mediante dati dei sensori di qualità dell'aria

Il monitoraggio della qualità dell'aria svolge un ruolo fondamentale nella tutela della salute pubblica e nelle politiche ambientali. Le moderne stazioni raccolgono continuamente grandi quantità di dati da sensori chimici e strumenti meteorologici. La qualità dell'aria è quindi un ambito ideale per mostrare come il machine learning possa supportare decisioni ambientali reali.

Nella pratica, le decisioni ambientali raramente dipendono soltanto dalla previsione di un valore esatto dell'inquinamento. Le domande principali sono spesso semplici:

- È stato superato un limite di legge relativo alla qualità dell'aria?
- Occorre emettere un'allerta?
- È necessario attivare misure di mitigazione?

Questo caso di studio risponde a domande di questo tipo prevedendo mediante il machine learning gli eventi di superamento della soglia di NO_2.

DATASET per il caso di studio

In questa lezione utilizziamo il dataset Air Quality dell'UCI Machine Learning Repository, un noto archivio pubblico di dati per il machine learning.**Air Quality dataset**

- **Periodo: da marzo 2004 a febbraio 2005**
- **Frequenza di campionamento: una misurazione ogni ora**
- **Luogo: stazione urbana di monitoraggio in una città italiana**
- **Numero di osservazioni: circa 9.300 registrazioni orarie**

Ogni record rappresenta le condizioni ambientali misurate in una determinata ora.

Il dataset combina tre tipologie di misurazioni:

Concentrazioni di inquinanti misurate con strumenti di riferimento

Questi valori provengono da analizzatori di riferimento e rappresentano misurazioni chimiche affidabili:

- Biossido di azoto – NO_2(GT) (µg/m³)
- Monossido di carbonio – CO(GT) (mg/m³)
- Ossidi di azoto – NOx(GT) (ppb)
- Benzene – C_6H_6(GT) (µg/m³)

Nella descrizione originale del dataset queste variabili costituiscono le variabili obiettivo dei modelli di regressione.

Segnali dei sensori

Il dataset contiene anche misure provenienti da sensori a ossidi metallici, meno costosi ma più rumorosi degli analizzatori di riferimento. I loro segnali sono spesso non lineari e sensibili a più sostanze. In questo caso di studio tali misurazioni costituiscono le caratteristiche in ingresso dei nostri modelli.

Variabili meteorologiche

Sono incluse condizioni ambientali che influenzano i livelli di inquinamento e il comportamento dei sensori:

- Temperatura
- Umidità relativa
- Umidità assoluta

Osservazioni sulla natura temporale dei dati

Tutte le misurazioni sono corredate da un timestamp e raccolte in sequenza: le osservazioni vicine nel tempo sono quindi correlate e non completamente indipendenti.

In questa lezione, tuttavia, ogni misurazione viene trattata come un'istantanea delle condizioni ambientali in una determinata ora. Per evitare problemi metodologici, suddividiamo cronologicamente i dati tra addestramento e test, simulando l'uso reale del sistema.

Osservazioni sulla formulazione originale del problema

Secondo il repository UCI, questo dataset è destinato a un problema di regressione: prevedere concentrazioni continue di inquinanti (come NO_2) a partire da segnali dei sensori e dati meteorologici.

Questa formulazione è appropriata quando l'obiettivo è una previsione numerica precisa.

Riformulazione come problema di classificazione

Nel monitoraggio ambientale spesso è più importante sapere se viene superata una soglia critica che prevedere l'esatto valore di un inquinante. I sistemi normativi e le allerte pubbliche si attivano di norma al superamento di limiti prestabiliti.

Per questo riformuliamo il problema originale di regressione come un compito di classificazione binaria: anziché prevedere la concentrazione esatta di NO_2, prevediamo se si verifica un superamento della soglia.

Soglia normativa utilizzata

La soglia adottata in questo caso di studio deriva dalla normativa europea ufficiale sulla qualità dell'aria:

Direttiva 2008/50/CE del Parlamento europeo e del Consiglio del 21 maggio 2008 relativa alla qualità dell'aria ambiente e per un'aria più pulita in Europa. La direttiva stabilisce che:

- Il valore limite orario del NO_2 è pari a 200 µg/m³.
- Il valore limite non deve essere superato più di 18 volte in un anno.

Il testo ufficiale (EUR-Lex) riporta tali limiti nell'ALLEGATO XI, sezione B (Valori limite per la protezione della salute umana).

L'utilizzo di questa soglia conferisce al compito di classificazione un riferimento normativo e un significato ambientale concreti.

Definizione delle due classi

Sulla base della normativa UE definiamo la variabile obiettivo come segue:

- **Classe 0 –** qualità dell'aria accettabile: concentrazione di NO_2 inferiore a 200 µg/m³
- **Classe 1 –** superamento della soglia: concentrazione di NO_2 uguale o superiore a 200 µg/m³

Si ottengono così due classi chiare e interpretabili.

Perché la regressione logistica è adatta a questa lezione introduttiva

Con questa riformulazione, la regressione logistica stima la probabilità di un superamento della soglia di NO_2 a partire dai dati meteorologici e dei sensori disponibili.

La regressione logistica è adatta perché:

- Fornisce stime di probabilità, non soltanto etichette di classe.
- Le sue frontiere di classificazione sono facili da interpretare.
- Permette di illustrare chiaramente il feature engineering.
- Consente di osservare facilmente overfitting e underfitting.
- Supporta visualizzazioni intuitive del comportamento del modello.

Natura di serie temporale del dataset nella riformulazione del problema

Sebbene i dati siano raccolti nel tempo, questo non compromette il caso di studio presentato.

In questa lezione:

- Non modelliamo esplicitamente le dinamiche temporali.
- Ogni osservazione viene trattata come un'istantanea indipendente.
- Utilizziamo una suddivisione cronologica tra addestramento e test per evitare la fuga di informazioni (data leakage).

Le tecniche specifiche per le serie temporali sono volutamente escluse, per mantenere l'attenzione sui concetti fondamentali di machine learning.

Questo caso di studio permette quindi di affrontare, in un contesto ambientale realistico, numerosi concetti importanti:

- trasformare un dataset di regressione in un compito di classificazione mediante soglie definite dalla normativa;
- costruire caratteristiche sulla base delle conoscenze ambientali;
- comprendere frontiere decisionali lineari e non lineari;
- separare correttamente i dati di addestramento e test;
- riconoscere overfitting e underfitting;
- utilizzare visualizzazioni per interpretare e analizzare i modelli.

Il caso di studio crea così un collegamento chiaro e pratico tra la teoria del machine learning e le applicazioni ambientali.**[14]**

Per i limiti previsti dalla direttiva ci concentriamo sulla classificazione dei superamenti orari. Carichiamo i dati con il pacchetto ucimlrepo; utilizziamo anche scikit-learn, pandas, numpy e matplotlib.[15]

[1] ✓ 8 s

```
# If running in Google Colab, install dependencies (safe to run even if already installed)
!pip -q install ucimlrepo scikit-learn pandas numpy matplotlib
```

[2] ✓ 5 s

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, ConfusionMatrixDisplay, roc_auc_score, RocCurveDisplay
)

# For reproducibility
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)


from sklearn.dummy import DummyClassifier
from sklearn.metrics import (
    precision_recall_curve, average_precision_score, PrecisionRecallDisplay
)

```

Il modello utilizzato è LogisticRegression di scikit-learn, con metriche standard per valutarne le prestazioni. Un valore random_state consente la riproducibilità dei risultati. DummyClassifier è un classificatore semplice che assegna tutte le osservazioni alla classe di maggioranza: permette di illustrare i limiti di alcune metriche nei problemi di classificazione sbilanciata, come il nostro. I dati sono caricati direttamente tramite ucimlrepo, senza download manuali.

[3] ✓ 1 s

```
from ucimlrepo import fetch_ucirepo

# UCI ML Repo dataset: Air Quality
# Note: In ucimlrepo, dataset IDs can change or multiple IDs can exist.
# We attempt fetch by name; if it fails, we'll guide a fallback.
try:
    air_quality = fetch_ucirepo(name="Air Quality")
    X_raw = air_quality.data.features
    y_raw = air_quality.data.targets  # may be empty for some UCI loaders
    meta = air_quality.metadata
    print("Loaded from UCI via ucimlrepo.")
    print("Features shape:", X_raw.shape)
    print("Targets shape:", None if y_raw is None else y_raw.shape)
    display(meta)
except Exception as e:
    print("Could not fetch via ucimlrepo:", repr(e))
    print("\nFallback option: download the dataset from UCI and upload to Colab, then read it with pandas.")
    raise

```

La pulizia dei valori mancanti e la conversione delle date in timestamp fanno parte della preparazione dei dati. In particolare:

- Combiniamo Date e Time in un timestamp.
- Sostituiamo -200 con NaN.
- Eliminiamo le righe nelle quali manca la misura di riferimento di NO_2 (necessaria per le etichette).
- Facoltativamente eliminiamo le colonne con troppi valori mancanti.

[4]
✓ 0 s

```python
df = X_raw.copy()

# In many versions of the dataset, Date and Time columns exist like 'Date' and 'Time'
# We'll try to detect them robustly.
date_col_candidates = [c for c in df.columns if c.lower() in ("date", "data")]
time_col_candidates = [c for c in df.columns if c.lower() in ("time", "ora")]

print("Date candidates:", date_col_candidates)
print("Time candidates:", time_col_candidates)

if not date_col_candidates or not time_col_candidates:
    raise ValueError("Could not find Date/Time columns. Inspect df.columns and adjust parsing.")

date_col = date_col_candidates[0]
time_col = time_col_candidates[0]

# Create a timestamp (common format is dd/mm/yyyy and hh.mm.ss or hh.mm)
# We'll try flexible parsing.
df["timestamp"] = pd.to_datetime(df[date_col].astype(str) + " " + df[time_col].astype(str),
                                 errors="coerce", dayfirst=True)

# Replace -200 with NaN in numeric columns
for c in df.columns:
    if c == "timestamp":
        continue
    # Attempt numeric conversion if possible
    df[c] = pd.to_numeric(df[c], errors="ignore")

num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols] = df[num_cols].replace(-200, np.nan)

# Sort by time
df = df.sort_values("timestamp").reset_index(drop=True)

# Identify the NO2 ground-truth column (commonly 'NO2(GT)')
no2_candidates = [c for c in df.columns if "no2" in c.lower() and "gt" in c.lower()]
print("NO2(GT) candidates:", no2_candidates)
if not no2_candidates:
    raise ValueError("Could not find NO2(GT) column. Inspect df.columns to locate it.")
NO2_COL = no2_candidates[0]

# Drop rows without timestamp or without NO2 ground truth
df = df.dropna(subset=["timestamp", NO2_COL]).reset_index(drop=True)

print("After cleaning:")
print(df.shape)
df.head()

```

DataFrame dopo la pulizia:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH	timestamp
0	4/1/2004	0:00:00	1.6	1143.0	106.0	6.3	825.0	96.0	986.0	86.0	1477.0	978.0	12.0	61.6	0.8593	2004-01-04 00:00:00
1	4/1/2004	1:00:00	1.2	1044.0	100.0	5.1	770.0	85.0	1031.0	70.0	1425.0	944.0	11.5	63.9	0.8652	2004-01-04 01:00:00
2	4/1/2004	2:00:00	1.1	1034.0	71.0	4.1	716.0	50.0	1085.0	55.0	1405.0	891.0	10.7	67.2	0.8630	2004-01-04 02:00:00
3	4/1/2004	4:00:00	0.7	909.0	44.0	2.4	615.0	57.0	1237.0	49.0	1322.0	790.0	10.2	66.6	0.8299	2004-01-04 04:00:00
4	4/1/2004	5:00:00	0.9	996.0	45.0	2.9	648.0	64.0	1176.0	50.0	1340.0	852.0	11.0	63.7	0.8325	2004-01-04 05:00:00

Per riformulare il problema come classificazione utilizziamo il livello di superamento indicato dalla direttiva UE. Definiamo le etichette:

- **y = 1** se $NO_2 \geq 200$ µg/m³
- **y = 0** altrimenti

Questa soglia è definita dalla normativa e conferisce alle classi un significato concreto.

```
[6]
✓ 0 s
```

```
1 df["y_exceed"] = (df[NO2_COL] >= EU_NO2_THRESHOLD).astype(int)
2
3 pos_rate = df["y_exceed"].mean()
4 print(f"Positive class rate (NO₂ exceedance): {pos_rate:.4f} ({pos_rate*100:.2f}%)")
5 df[["timestamp", NO2_COL, "y_exceed"]].head()
6
```

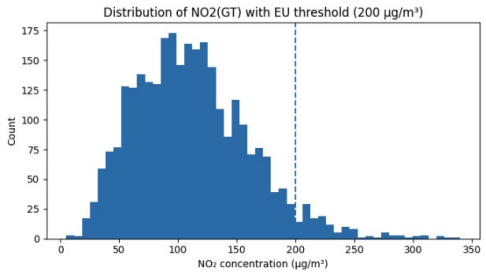
```
Positive class rate (NO₂ exceedance): 0.0478 (4.78%)
```

	timestamp	NO2(GT)	y_exceed
0	2004-01-04 00:00:00	86.0	0
1	2004-01-04 01:00:00	70.0	0
2	2004-01-04 02:00:00	55.0	0
3	2004-01-04 04:00:00	49.0	0
4	2004-01-04 05:00:00	50.0	0

Dai dati si osserva che soltanto il 4,78% delle osservazioni appartiene alla classe 1 (superamento): un tipico esempio di classificazione sbilanciata. La distribuzione del NO_2 rispetto alla soglia UE rende evidente lo sbilanciamento.

```
[22]
✓ 0 s
```

```
1 # NO2 distribution with EU threshold line
2 EU_NO2_THRESHOLD = 200.0  # µg/m³
3
4 plt.figure(figsize=(7,4))
5 plt.hist(df[NO2_COL].dropna(), bins=50)
6 plt.axvline(EU_NO2_THRESHOLD, linestyle="--")
7 plt.title(f"Distribution of {NO2_COL} with EU threshold (200 µg/m³)")
8 plt.xlabel("NO₂ concentration (µg/m³)")
9 plt.ylabel("Count")
10 plt.tight_layout()
11 plt.show()
12
13 # Class balance preview (will define label in next section)
14
15
```



Procediamo con la preparazione dei dati (imputazione della mediana) e il feature engineering (selezione delle caratteristiche): eliminiamo Date, Time e timestamp, manteniamo le variabili numeriche dei sensori e meteorologiche e infine rimuoviamo le colonne troppo sparse.

[8] ✓ 0s

```
# Drop non-features
drop_cols = [date_col, time_col, "timestamp", NO2_COL, "y_exceed"]
feature_df = df.drop(columns=[c for c in drop_cols if c in df.columns], errors="ignore")

# Keep numeric features only
X_all = feature_df.select_dtypes(include=[np.number]).copy()
y_all = df["y_exceed"].copy()

# Optionally drop columns with too much missing data (e.g., > 40% missing)
too_missing = X_all.columns[X_all.isna().mean() > 0.40]
print("Dropping columns with >40% missing:", list(too_missing))
X_all = X_all.drop(columns=too_missing)

# Median imputation
X_all = X_all.fillna(X_all.median(numeric_only=True))

print("Final feature matrix:", X_all.shape)
X_all.head()

```

```
Dropping columns with >40% missing: ['NMHC(GT)']
Final feature matrix: (2910, 11)
```

	CO(GT)	PT08.S1(CO)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
0	1.6	1143.0	6.3	825.0	96.0	986.0	1477.0	978.0	12.0	61.6	0.8593
1	1.2	1044.0	5.1	770.0	85.0	1031.0	1425.0	944.0	11.5	63.9	0.8652
2	1.1	1034.0	4.1	716.0	50.0	1085.0	1405.0	891.0	10.7	67.2	0.8630
3	0.7	909.0	2.4	615.0	57.0	1237.0	1322.0	790.0	10.2	66.6	0.8299
4	0.9	996.0	2.9	648.0	64.0	1176.0	1340.0	852.0	11.0	63.7	0.8325

Possiamo ora mostrare perché l'accuratezza può essere fuorviante nei dataset fortemente sbilanciati. Con la soglia UE ($NO_2 \geq 200$ µg/m³), la classe positiva è rara: un modello che predice sempre «nessun superamento» può raggiungere un'accuratezza elevata pur essendo inutile. Calcoliamo quindi una baseline basata sulla classe di maggioranza (DummyClassifier) e introduciamo precision, recall e PR-AUC (Average Precision).

- Accuracy ≈ 0,952: circa il 95,2% delle ore nel nostro campione non presenta superamenti. Il modello dummy predice soltanto la classe maggioritaria (0), risultando apparentemente buono in termini di accuratezza.
- Precision = 0,0; Recall = 0,0; F1 = 0,0: il modello non prevede mai un superamento (classe 1). Quindi:
    - Non rileva alcun superamento → Recall = 0
    - Non produce previsioni positive → Precision è posta a 0 (con zero_division = 0)
    - Anche F1 è pari a 0
- PR-AUC ≈ 0,0481. È il valore più informativo: per un classificatore che produce sempre la stessa previsione, la PR-AUC tende a essere vicina alla frequenza

della classe positiva. Il tasso di superamento è quindi circa il 4,8%, coerentemente con l'accuratezza del 95%.

```
[12]
✓ 0s
```

```python
# DummyClassifier baseline (accuracy trap demonstration)
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, average_precision_score

X_tmp = X_all.copy()
y_tmp = y_all.copy()

# Random split ONLY to demonstrate the accuracy trap
# (Proper chronologica Ask is used later for real evaluation)
Xtr, Xte, ytr, yte = t_ st_split(
    X_tmp, y_tmp,
    test_size=0.3,
    random_state=RANDOM_STATE,
    stratify=y_tmp
)

dummy = DummyClassifier(strategy="most_frequent")
dummy.fit(Xtr, ytr)

y_pred = dummy.predict(Xte)
y_proba = dummy.predict_proba(Xte)[:, 1]

print("DummyClassifier (majority class baseline)")
print("Accuracy :", accuracy_score(yte, y_pred))
print("Precision:", precision_score(yte, y_pred, zero_division=0))
print("Recall   :", recall_score(yte, y_pred, zero_division=0))
print("F1       :", f1_score(yte, y_pred, zero_division=0))
print("PR-AUC   :", average_precision_score(yte, y_proba))

```

```
DummyClassifier (majority class baseline)
Accuracy : 0.9518900343642611
Precision: 0.0
Recall   : 0.0
F1       : 0.0
PR-AUC   : 0.048109965635738834
```

Il modello di regressione logistica deve apprendere dai dati, ma anche generalizzare a osservazioni mai viste dello stesso dominio. Per questo suddividiamo il dataset in addestramento e test. Essendo una serie temporale, utilizziamo il primo 70% delle osservazioni per l'addestramento e l'ultimo 30% per il test (suddivisione cronologica), evitando la fuga di informazioni e simulando un contesto realistico.

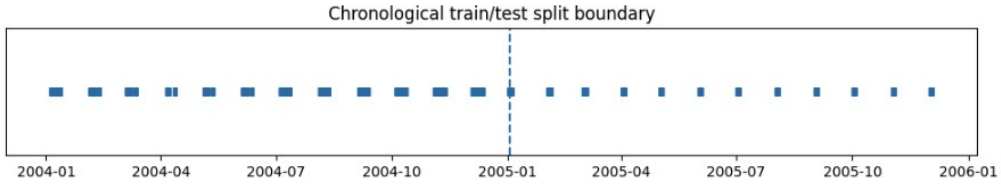
Addestriamo quindi la regressione logistica con standardizzazione e rappresentiamo:

- il diagramma a dispersione delle osservazioni (colorate per classe);
- la frontiera di decisione lineare;
- le curve di livello della probabilità (facoltative).

[13]
✓ 0 s

```python
# Chronological split index
split_idx = int(len(df) * 0.70)

X_train = X_all.iloc[:split_idx].copy()
y_train = y_all.iloc[:split_idx].copy()
X_test  = X_all.iloc[split_idx:].copy()
y_test  = y_all.iloc[split_idx:].copy()

print("Train size:", X_train.shape, "Test size:", X_test.shape)
print("Train positive rate:", y_train.mean(), "Test positive rate:", y_test.mean())

# Visualize split on timeline
Ask ure(figsize=(10,2))
plot(df["timestamp"], np.zeros(len(df)), marker="|", linestyle="None")
plt.axvline(df["timestamp"].iloc[split_idx], linestyle="--")
plt.title("Chronological train/test split boundary")
plt.yticks([])
plt.tight_layout()
plt.show()

```

```
Train size: (2036, 11) Test size: (874, 11)
Train positive rate: 0.007858546168958742 Test positive rate: 0.14073226544622425
```



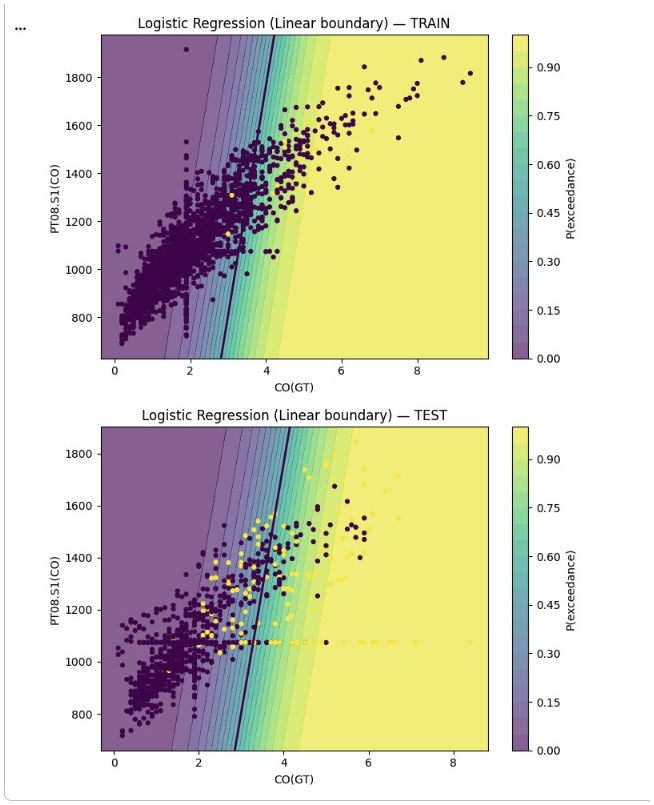
[14]
✓ 0 s

```python
# Pick two features to visualize
# We'll choose two columns with decent variance
candidate_cols = X_all.columns.tolist()
if len(candidate_cols) < 2:
    raise ValueError("Not enough numeric features to visualize.")

feat1, feat2 = candidate_cols[0], candidate_cols[1]
print("Using features:", feat1, "and", feat2)

X2_train = X_train[[feat1, feat2]].copy()
X2_test  = X_test[[feat1, feat2]].copy()

lin_model = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(max_iter=2000, class_weight="balanced", random_state=RANDOM_STATE))
])
lin_model.fit(X2_train, y_train)

print("Train accuracy:", accuracy_score(y_train, lin_model.predict(X2_train)))
print("Test accuracy :", accuracy_score(y_test,  lin_model.predict(X2_test)))

# More informative metrics on TEST (imbalanced setting)
y_pred_lin = lin_model.predict(X2_test)
y_proba_lin = lin_model.predict_proba(X2_test)[:,1]

print("TEST Precision:", precision_score(y_test, y_pred_lin, zero_division=0))
print("TEST Recall   :", recall_score(y_test, y_pred_lin, zero_division=0))
print("TEST F1       :", f1_score(y_test, y_pred_lin, zero_division=0))
if len(np.unique(y_test)) == 2:
    print("TEST PR-AUC (AP):", average_precision_score(y_test, y_proba_lin))

```

```
Using features: CO(GT) and PT08.S1(CO)
Train accuracy: 0.8742632612966601
Test accuracy : 0.8649885583524027
TEST Precision: 0.5177304964539007
TEST Recall   : 0.5934959349593496
TEST F1       : 0.553030303030303
TEST PR-AUC (AP): 0.593996588340624
```

[15] ✓ 0 s

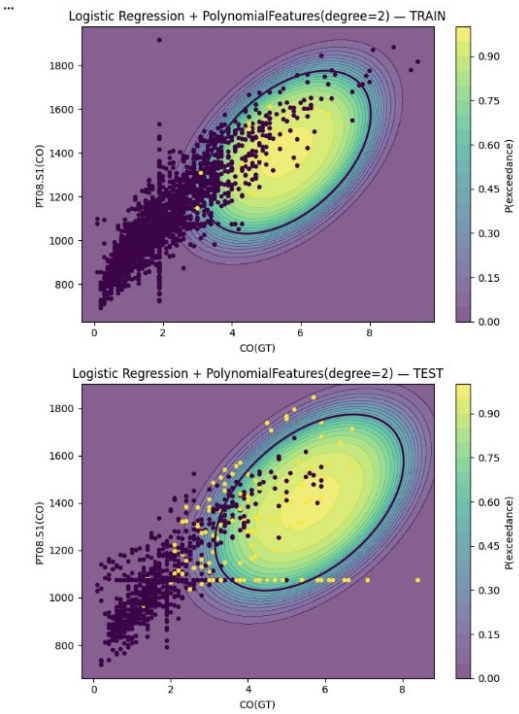
```python
def plot_decision_boundary_2d(model, X, y, title, grid_steps=200):
    # Create grid
    x_min, x_max = X.iloc[:,0].min(), X.iloc[:,0].max()
    y_min, y_max = X.iloc[:,1].min(), X.iloc[:,1].max()
    pad_x = 0.05 * (x_max - x_min + 1e-9)
    pad_y = 0.05 * (y_max - y_min + 1e-9)
    x_min, x_max = x_min - pad_x, x_max + pad_x
    y_min, y_max = y_min - pad_y, y_max + pad_y

    xx, yy = np.meshgrid(np.linspace(x_min, x_max, grid_steps),
                         np.linspace(y_min, y_max, grid_steps))
    grid = pd.DataFrame({X.columns[0]: xx.ravel(), X.columns[1]: yy.ravel()})

    # Predict probabilities for class 1
    probs = model.predict_proba(grid)[:,1].reshape(xx.shape)

    plt.figure(figsize=(7,5))
    # Probability contour
    cs = plt.contourf(xx, yy, probs, levels=20, alpha=0.6)
    plt.colorbar(cs, label="P(exceedance)")

    # Scatter points
    plt.scatter(X.iloc[:,0], X.iloc[:,1], c=y, s=12)
    # Decision boundary at p=0.5
    plt.contour(xx, yy, probs, levels=[0.5], linewidths=2)

    plt.xlabel(X.columns[0])
    plt.ylabel(X.columns[1])
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_decision_boundary_2d(lin_model, X2_train, y_train,
                          "Logistic Regression (Linear boundary) — TRAIN")
plot_decision_boundary_2d(lin_model, X2_test, y_test,
                          "Logistic Regression (Linear boundary) — TEST")
```



La visualizzazione della frontiera lineare mostra come la regressione logistica separi le due classi mediante una retta alla soglia di probabilità p = 0,5. Evidenzia sia la separazione delle classi sia la confidenza del modello nelle varie regioni dello spazio delle caratteristiche.

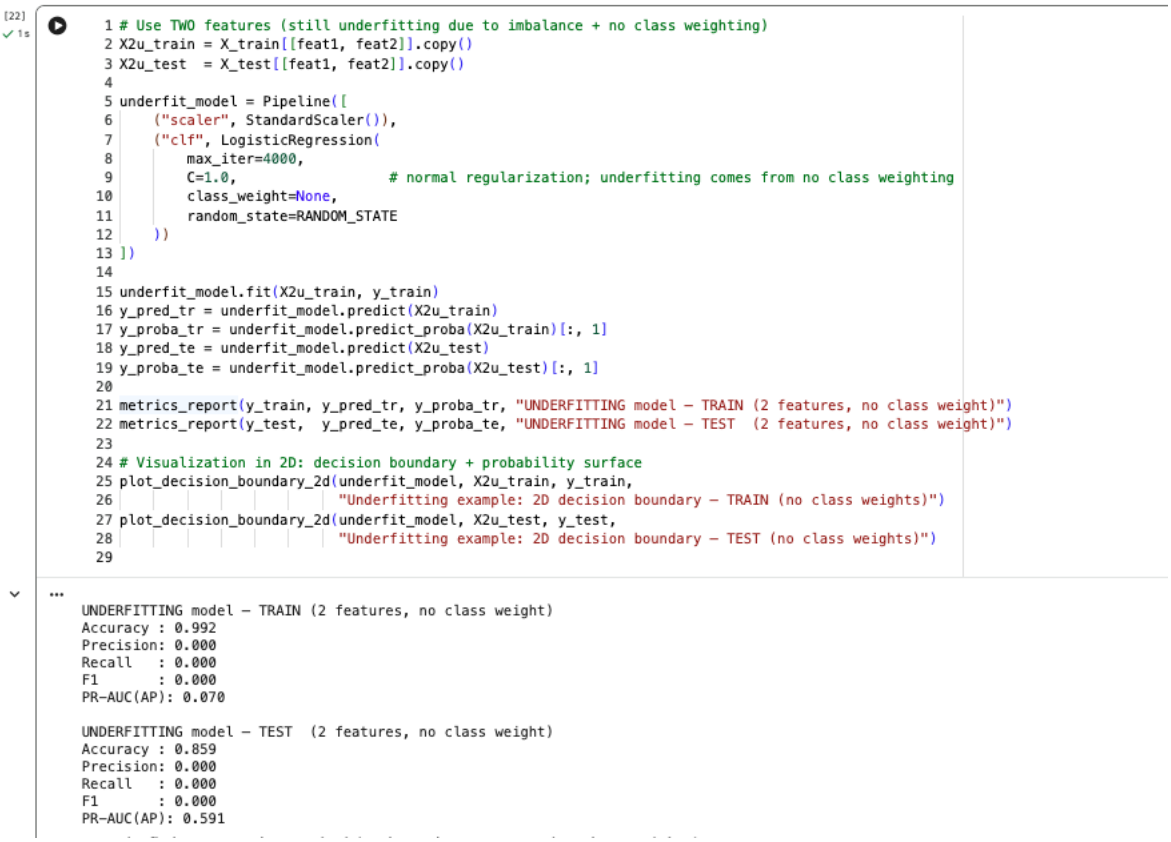
La regressione logistica è lineare nei parametri, ma trasformando le caratteristiche in ingresso (per esempio aggiungendo quadrati e termini d'interazione) possiamo ottenere una frontiera non lineare nello spazio originale. Usiamo PolynomialFeatures di grado 2 per questo feature engineering aggiuntivo.

```
[23]
✓ 4 s
poly2_model = Pipeline([
    ("scaler", StandardScaler()),
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("clf", LogisticRegression(max_iter=5000, class_weight="balanced", random_state=RANDOM_STATE))
])
poly2_model.fit(X2_train, y_train)

print("Train accuracy (degree=2):", accuracy_score(y_train, poly2_model.predict(X2_train)))
print("Test accuracy  (degree=2):", accuracy_score(y_test,  poly2_model.predict(X2_test)))
# More informative metrics on TEST
y_pred_poly2 = poly2_model.predict(X2_test)
y_proba_poly2 = poly2_model.predict_proba(X2_test)[:,1]
print("TEST Precision (deg=2):", precision_score(y_test, y_pred_poly2, zero_division=0))
print("TEST Recall    (deg=2):", recall_score(y_test, y_pred_poly2, zero_division=0))
print("TEST F1        (deg=2):", f1_score(y_test, y_pred_poly2, zero_division=0))
if len(np.unique(y_test)) == 2:
    print("TEST PR-AUC (AP) (deg=2):", average_precision_score(y_test, y_proba_poly2))


plot_decision_boundary_2d(poly2_model, X2_train, y_train,
                          "Logistic Regression + PolynomialFeatures(degree=2) — TRAIN")
plot_decision_boundary_2d(poly2_model, X2_test, y_test,
                          "Logistic Regression + PolynomialFeatures(degree=2) — TEST")


...
Train accuracy (degree=2): 0.8487229862475442
Test accuracy  (degree=2): 0.8260869565217391
TEST Precision (deg=2): 0.4013605442176871
TEST Recall    (deg=2): 0.4796747967479675
TEST F1        (deg=2): 0.43703703703703706
TEST PR-AUC (AP) (deg=2): 0.44004674954757994
```



La frontiera non lineare mostra una separazione curva prodotta dal feature engineering, che aumenta la flessibilità del modello ma anche il rischio di overfitting nei dati ambientali rumorosi e sbilanciati. Il confronto indica che la regressione logistica lineare ottiene prestazioni migliori del modello non lineare di grado 2: una frontiera semplice sembra sufficiente, mentre il modello più complesso tende ad adattarsi eccessivamente ai dati.

Due concetti fondamentali sono overfitting e underfitting. In un esperimento controllato li illustriamo variando la complessità del modello tramite il grado polinomiale (complessità delle caratteristiche) e la forza della regolarizzazione C nella regressione logistica. Per mostrare l'underfitting disattiviamo il bilanciamento dei pesi delle classi. Confrontando le prestazioni su addestramento e test possiamo osservare se il modello non apprende sufficientemente oppure si adatta troppo ai dati di addestramento.



```
[22]
✓ 1s
```

```python
# Use TWO features (still underfitting due to imbalance + no class weighting)
X2u_train = X_train[[feat1, feat2]].copy()
X2u_test  = X_test[[feat1, feat2]].copy()

underfit_model = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(
        max_iter=4000,
        C=1.0,                  # normal regularization; underfitting comes from no class weighting
        class_weight=None,
        random_state=RANDOM_STATE
    ))
])

underfit_model.fit(X2u_train, y_train)
y_pred_tr = underfit_model.predict(X2u_train)
y_proba_tr = underfit_model.predict_proba(X2u_train)[:, 1]
y_pred_te = underfit_model.predict(X2u_test)
y_proba_te = underfit_model.predict_proba(X2u_test)[:, 1]

metrics_report(y_train, y_pred_tr, y_proba_tr, "UNDERFITTING model — TRAIN (2 features, no class weight)")
metrics_report(y_test,  y_pred_te, y_proba_te, "UNDERFITTING model — TEST  (2 features, no class weight)")

# Visualization in 2D: decision boundary + probability surface
plot_decision_boundary_2d(underfit_model, X2u_train, y_train,
                          "Underfitting example: 2D decision boundary — TRAIN (no class weights)")
plot_decision_boundary_2d(underfit_model, X2u_test, y_test,
                          "Underfitting example: 2D decision boundary — TEST (no class weights)")

```

```
UNDERFITTING model — TRAIN (2 features, no class weight)
Accuracy : 0.992
Precision: 0.000
Recall   : 0.000
F1       : 0.000
PR-AUC(AP): 0.070

UNDERFITTING model — TEST  (2 features, no class weight)
Accuracy : 0.859
Precision: 0.000
Recall   : 0.000
F1       : 0.000
PR-AUC(AP): 0.591
```

Questo esempio di underfitting è ottenuto rimuovendo deliberatamente la ponderazione delle classi. Senza questa correzione il modello apprende soprattutto a prevedere la classe di maggioranza e non rileva i superamenti di NO_2. L'accuratezza elevata su addestramento e test dipende dallo sbilanciamento, mentre recall, F1 e PR-AUC sono bassi su entrambi: il modello non riconosce gli eventi rari ma importanti.[16]

```python
overfit_degree = 7
overfit_C = 50.0

overfit_model = Pipeline([
    ("poly", PolynomialFeatures(degree=overfit_degree, include_bias=False)),
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(
        solver="lbfgs",
        max_iter=50000,
        C=overfit_C,
        class_weight="balanced",
        random_state=RANDOM_STATE
    ))
])
overfit_model.fit(X2_train, y_train)

y_pred_tr = overfit_model.predict(X2_train)
y_proba_tr = overfit_model.predict_proba(X2_train)[:,1]
y_pred_te = overfit_model.predict(X2_test)
y_proba_te = overfit_model.predict_proba(X2_test)[:,1]

metrics_report(y_train, y_pred_tr, y_proba_tr, f"OVERFITTING model — TRAIN (deg={overfit_degree}, C={overfit_C})")
metrics_report(y_test,  y_pred_te, y_proba_te, f"OVERFITTING model — TEST  (deg={overfit_degree}, C={overfit_C})")

plot_decision_boundary_2d(overfit_model, X2_train, y_train, f"Overfitting: nonlinear boundary (deg={overfit_degree}) — TRAIN")
plot_decision_boundary_2d(overfit_model, X2_test,  y_test,  f"Overfitting: nonlinear boundary (deg={overfit_degree}) — TEST")
```

```
OVERFITTING model — TRAIN (deg=7, C=50.0)
Accuracy : 0.854
Precision: 0.045
Recall   : 0.875
F1       : 0.086
PR-AUC(AP): 0.128

OVERFITTING model — TEST  (deg=7, C=50.0)
Accuracy : 0.838
Precision: 0.437
Recall   : 0.537
F1       : 0.482
PR-AUC(AP): 0.520
```

Nell’overfitting il modello diventa troppo flessibile e si adatta eccessivamente ai dati di addestramento. Può così rilevare gran parte dei superamenti ma generare numerosi falsi allarmi, con precision e F1-score bassi. Sui nuovi dati il comportamento cambia: il recall diminuisce, la precision varia e le prestazioni diventano meno affidabili. La differenza marcata tra addestramento e test segnala una scarsa generalizzazione.

Esercitazione pratica: individuazione dell’habitat di abete rosso/abete mediante regressione logistica

Dataset dell’esercitazione: utilizziamo Forest Cover Type (Covertype), che contiene variabili cartografiche e ambientali (quota, pendenza, distanze idrologiche, ombreggiamento, aree naturali, tipi di suolo, ecc.). L’etichetta indica il tipo di copertura forestale.

Dal punto di vista ambientale, questo è un compito tipico:

- **Prevedere classi di habitat o di copertura del suolo a partire da descrittori del terreno e dell’ambiente.**

Il dataset originale comprende sette tipi di copertura (classificazione multiclasse). Per utilizzare la regressione logistica binaria e rendere le metriche e i grafici più facili da interpretare, trasformiamo il problema in uno binario:original dataset

- **classe positiva (1): tipo di copertura 1 = Spruce/Fir (abete rosso/abete)**
- **classe negativa (0): tutti gli altri tipi di copertura**

Questa riformulazione rappresenta uno scenario realistico: individuare uno specifico habitat di interesse rispetto a tutti gli altri tipi di copertura. Il dataset può essere scaricato direttamente con fetch_covtype() di scikit-learn.

Strumenti e ambiente di lavoro

- Linguaggio: Python (Google Colab)
- Librerie: numpy, pandas, scikit-learn, matplotlib
- Accesso ai dati: sklearn.datasets.fetch_covtype
- Modalità di lavoro: individuale

Attività: descrizione delle caratteristiche

Gli studenti devono consultare la documentazione originale del dataset e creare una tabella descrittiva delle caratteristiche con le seguenti informazioni:

- Nome della caratteristica
- Tipo di caratteristica (continua o categorica / codificata one-hot)
- Unità di misura (se presente)
- Significato fisico o ambientale

Il dataset comprende 10 variabili ambientali continue e 44 variabili categoriche (4 indicatori di aree naturali e 40 indicatori di tipi di suolo, codificati one-hot).

Compiti degli studenti (passo dopo passo)

1. Caricare i dati da scikit-learn mediante fetch_covtype().
2. Poiché il dataset è grande, estrarre un campione di N = 50.000 osservazioni mediante campionamento stratificato.

3. Importare tutte le librerie (come nel caso di studio della lezione) e garantire la riproducibilità degli esperimenti fissando un seed casuale.
4. Trasformare il problema multiclasse in classificazione binaria.
    a. Classe etichettata 1 = Spruce/Fir (classe positiva di interesse).
    b. Classe etichettata 0 = tutti gli altri tipi di copertura (classe negativa).

    Trattare Spruce/Fir come classe positiva, poiché l'obiettivo è individuare l'habitat di interesse.
5. Esaminare e riportare la distribuzione tra classe positiva e negativa.
6. Spiegare perché la riformulazione binaria è realistica nel monitoraggio ambientale.
7. Controllare i valori mancanti.
    a. Se i valori mancanti sono pochi, eliminare le relative osservazioni.
    b. Se sono numerosi, imputarli (media o moda, in base al tipo di caratteristica).
    c. Se non ve ne sono, dichiarare esplicitamente che non sono necessarie azioni.
8. Controllare facoltativamente i duplicati ed esaminare gli outlier, almeno per le variabili continue.
9. Creare set di addestramento e test (70/30) mantenendo la stessa distribuzione delle classi mediante stratify=y e spiegare il ruolo di ciascun insieme.
10. Esaminare le scale delle prime 10 caratteristiche e concludere quale preparazione iniziale richiede la regressione logistica. Importante: su quale insieme di dati deve essere effettuata?→
    a. Utilizzare una Pipeline con StandardScaler.
    b. Covertype contiene molte caratteristiche binarie one-hot (0/1); è più prudente utilizzare StandardScaler(with_mean=False).
    c. Spiegare perché la standardizzazione è importante per la regressione logistica (la regolarizzazione dipende dalla scala delle caratteristiche).
11. Addestrare un modello di regressione logistica di riferimento. Considerare che la regolarizzazione controlla la complessità: un C piccolo implica regolarizzazione più forte e modello più semplice (rischio di underfitting); un C grande implica regolarizzazione più debole e modello più flessibile (rischio di overfitting, soprattutto con caratteristiche derivate). La regressione logistica produce probabilità; la soglia predefinita 0,5 è una scelta convenzionale.→

12. Riportare le metriche di valutazione sui set di addestramento e test, spiegando quali metriche sono state scelte e perché.
13. Visualizzare la matrice di confusione sul test e spiegarla:
    - Spiegare il significato di TP, TN, FP e FN in questo contesto ambientale.
    - FP: il modello prevede Spruce/Fir ma il luogo presenta un'altra copertura; una mappa errata degli habitat può indurre decisioni sbagliate di gestione e conservazione.→→
    - FN: il modello prevede «altro», ma il luogo è realmente Spruce/Fir; il rischio è sottostimare o non proteggere adeguatamente l'habitat.→
    - Questo esempio mostra come il ML possa supportare le decisioni ambientali.
14. Confrontare le metriche ottenute al punto 9 e interpretare i compromessi tra le diverse misure di valutazione.
    a. Spiegare la natura del problema e perché sia importante rilevare gli errori FP e FN, considerando il costo di un habitat mancato rispetto a un falso allarme.
    b. Si tratta di un problema di rilevamento di habitat con costi asimmetrici: modificare la soglia decisionale influisce sul numero di FP e FN.
15. Spiegare precision, recall e PR-AUC nel caso specifico del rilevamento dell'habitat Spruce/Fir.
16. Aumentare la flessibilità del modello mediante espansione polinomiale delle sole caratteristiche continue (gradi 1, 2 e 4) e variare la forza della regolarizzazione (C = 0,1; 1,0; 50). Spiegare gli effetti attesi.
    a. Per evitare problemi di memoria in Colab applicare le features polinomiali soltanto alle 10 variabili continue.
17. Visualizzare l'F1-score sui dati di addestramento e test per tutte le configurazioni e interpretare gli andamenti.
18. Spiegare underfitting e overfitting usando i risultati del punto precedente.
19. Individuare una configurazione in underfitting e una in overfitting e motivare la scelta con dati numerici.

20. Visualizzare le frontiere di decisione in uno spazio bidimensionale mediante PCA per ridurre la dimensionalità (solo a fini di visualizzazione) e descrivere come cambiano con la complessità.
21.

Materiale da consegnare – Relazione sull'esercitazione

- Formato: notebook Google Colab con celle Markdown e celle di codice
- La consegna deve contenere:
    a. Introduzione (3–5 frasi)
        I. Qual era il problema (originale e riformulato)?
        II. Perché è rilevante per le scienze ambientali?
        III. Definire la classe positiva e spiegare i costi asimmetrici.
    b. Descrizione dei dati
        I. Numero di osservazioni e di caratteristiche (raggruppate per tipo)
        II. Variabile obiettivo e relativa distribuzione
        III. Gestione dei valori mancanti (descrivere le operazioni svolte)
        IV. Scelta del campionamento
    c. Metodi
        I. Modelli addestrati e modalità di addestramento (regressione logistica e parametri)
        II. Suddivisione addestramento/test (indicare la proporzione)
        III. Eventuale standardizzazione e insieme sul quale viene applicata
    d. Risultati
        I. Riportare accuracy, precision, recall, F1, ROC-AUC e PR-AUC in una tabella.
        II. Mostrare la matrice di confusione e spiegarla.
        III. Indicare quale modello scegliere e perché, non soltanto in base all'accuratezza, ma anche, ad esempio, al recall sulla classe positiva.
    e. Discussione

I. Quali sono i rischi degli errori di classificazione in questo esempio ambientale?
II. Qual è il ruolo della standardizzazione?
III. Quali ulteriori passi sarebbero necessari per un sistema di monitoraggio ambientale reale (ad esempio, scelta della soglia, compromesso precision–recall, importanza delle caratteristiche e interpretabilità)?

f. Conclusione
I. Riassumere in poche frasi che cosa si è appreso dall'esercitazione con dati ambientali reali, citando generalizzazione, scelta delle metriche e compromessi.

Valutazione

- **Quiz (10 domande):** verifica della comprensione dei concetti e della capacità di ragionamento applicato acquisite nella lezione e nell'esercitazione pratica sulle tecniche di classificazione nelle scienze ambientali. Il quiz verifica apprendimento supervisionato, frontiere di decisione lineari e non lineari, metriche fondamentali (accuracy, precision, recall, F1-score), suddivisione addestramento/test, underfitting, overfitting e importanza della visualizzazione.

- **Esercitazione pratica di classificazione:** previsione del superamento della soglia di NO_2. L'attività individuale permette agli studenti di affrontare un problema di classificazione relativo al monitoraggio ambientale discusso durante la lezione. Utilizzando dati di qualità dell'aria, gli studenti addestrano e valutano un modello lineare di riferimento (regressione logistica) per determinare se i livelli di biossido di azoto superano una soglia di sicurezza. Si valuta la capacità di realizzare l'intero flusso di lavoro: preparazione dei dati, gestione delle caratteristiche, addestramento e valutazione. Particolare attenzione è dedicata all'interpretazione di falsi positivi e falsi negativi.

- **Esercitazione pratica di classificazione:** rilevamento dell'habitat Spruce/Fir. L'attività individuale applica modelli di classificazione a un caso ambientale presentato nella lezione. Partendo da dati sulla copertura del suolo con variabili geografiche e ambientali, gli studenti costruiscono e valutano un classificatore lineare di riferimento (regressione logistica) per riconoscere uno specifico habitat forestale rispetto agli altri tipi di copertura. Si valuta l'implementazione dell'intero

processo, dalla preparazione alla valutazione, con particolare attenzione al significato degli errori di classificazione per le decisioni ambientali.

- **Relazione scritta breve**: costituisce la fase finale dell'esercitazione di rilevamento dell'habitat. Serve a riassumere, interpretare e comunicare chiaramente i risultati dei modelli applicati al problema Spruce/Fir. Gli studenti dovranno dimostrare di saper:
  - illustrare chiaramente le fasi del processo di classificazione, dalla preparazione dei dati alla valutazione;
  - interpretare le metriche di classificazione (accuracy, precision, recall, F1-score e matrice di confusione) nel contesto del rilevamento di habitat e delle decisioni ambientali;
  - riflettere sulla scelta del modello (lineare o non lineare), discutendo come la complessità influisca sulle prestazioni e sulle applicazioni reali, come cartografia degli habitat, pianificazione della conservazione e analisi dell'uso del suolo.

Criteri di valutazione

PER IL QUIZ

- Il quiz verifica le conoscenze fattuali, concettuali e teoriche fondamentali sulla classificazione nel machine learning, tra cui:
  - Il paradigma dell'apprendimento supervisionato;
  - La differenza tra frontiere di decisione lineari e non lineari;
  - Il ruolo della suddivisione in addestramento, validazione e test;
  - Le metriche di classificazione e i relativi utilizzi;
  - Esempi di problemi di classificazione ambientale (superamento della soglia di NO_2 e rilevamento di habitat).

Soglia di superamento: almeno il 50% delle risposte corrette.

PER L'ESERCITAZIONE PRATICA DI CLASSIFICAZIONE

- Preparazione e preelaborazione dei dati
  - Caricamento e ispezione corretti del dataset (dimensioni, obiettivo, valori mancanti).

- Gestione adeguata dei dati mancanti (eliminazione o imputazione) e, se necessario, standardizzazione/normalizzazione delle caratteristiche.
- Processo di preelaborazione trasparente e riproducibile (codice pulito e passaggi chiari).

- Implementazione dei modelli
  - Configurazione e addestramento corretti di almeno due modelli di classificazione:
    - un modello di riferimento (ad esempio, regressione logistica);
    - un modello non lineare.
  - Uso corretto della suddivisione addestramento/test (ed eventualmente della validazione).
  - Codice organizzato e parametri dichiarati esplicitamente.

- Valutazione e diagnostica dei modelli
  - Calcolo e presentazione corretti delle metriche di classificazione:
    - accuracy
    - precision
    - recall
    - F1-score
    - matrice di confusione
    - Confronto ragionato dei modelli (ad esempio: «La regressione logistica ha migliorato il recall della classe positiva rispetto al modello non lineare ottenuto con feature engineering»).
    - Consapevolezza dello sbilanciamento delle classi e dei suoi effetti sull'accuratezza.

- Scelta e giustificazione del modello
  - Motivazione chiara del modello preferito, non soltanto «accuratezza più elevata», ma, quando pertinente, «recall migliore sulla classe minoritaria o ambientalmente critica».
  - Spiegazione ragionevole della capacità di generalizzazione del modello sulla base dei risultati di test.

- Visualizzazione e presentazione dei risultati
  - La matrice di confusione è rappresentata, etichettata e interpretata.

- La scelta del modello migliore si basa sui dati effettivi riportati nella tabella o nel grafico.
- Le visualizzazioni sono leggibili e sostengono l'argomentazione.

- Collegamento al contesto ambientale
  - Lo studente dimostra di comprendere il significato degli errori di classificazione nel rilevamento degli habitat, riconoscendo che classificare un'area priva di habitat come Spruce/Fir può essere più grave del mancato rilevamento di un habitat realmente presente, secondo le priorità dello scenario.
  - Lo studente sa spiegare come il classificatore possa contribuire al monitoraggio ambientale, alla pianificazione della conservazione e alla gestione dell'uso del suolo, aiutando a individuare e seguire la distribuzione degli habitat Spruce/Fir, assegnare priorità alla protezione o alle verifiche sul campo e supportare decisioni basate sui dati.

PER LA RELAZIONE SCRITTA BREVE

- Chiarezza e struttura della relazione
  - La relazione è organizzata logicamente (Introduzione → Dati → Metodi → Risultati → Discussione → Conclusione).
  - Le fasi del flusso di lavoro sono visibili nel notebook.
  - Il linguaggio è preciso e conciso.

- Interpretazione dei risultati e discussione
  - I risultati di classificazione (accuracy, precision, recall, F1 e matrice di confusione) sono riportati correttamente e interpretati in modo significativo.
  - Lo studente dimostra di comprendere gli errori per ciascuna classe nella matrice di confusione.
  - Lo studente riflette sui motivi per cui un modello ha ottenuto risultati migliori di un altro.
  - La discussione considera overfitting, underfitting e ruolo della validazione.

- Collegamento al contesto ecologico e ambientale
  - I risultati sono collegati al loro significato ambientale (ad esempio: «Un recall più alto per Spruce/Fir conta più dell'accuratezza complessiva»).

- Lo studente comprende come la classificazione basata sui dati possa contribuire al monitoraggio ambientale e a decisioni informate.

- Qualità della presentazione grafica
  - Le figure (matrice di confusione, metriche di valutazione, confronto tra modelli) hanno etichette chiare.
  - Il testo richiama le figure (ad esempio «come mostrato nella Figura 1…»).
  - Le visualizzazioni aiutano a spiegare il comportamento del modello e non costituiscono una semplice decorazione.

- Comunicazione e stile
  - La scrittura è chiara, oggettiva e professionale.
  - I termini del ML e dell'ambiente sono utilizzati correttamente (classe, etichetta, caratteristica, precision, recall, classe positiva).
  - Lo studente riconosce i limiti (dataset piccolo, sbilanciamento, assenza di una componente spaziale), le difficoltà dei dati ad alta dimensionalità e la necessità di campionare dataset grandi quando si lavora nella versione gratuita di Google Colab.

QUIZ (10 domande)

1. Apprendimento supervisionato

Quale delle seguenti affermazioni descrive meglio l'apprendimento supervisionato nell'analisi dei dati ambientali?

A. Il modello scopre regolarità nei dati senza etichette predefinite.
B. Il modello apprende una relazione tra caratteristiche in ingresso ed etichette obiettivo note.
C. Il modello raggruppa automaticamente osservazioni ambientali simili.
D. Il modello ottimizza le regole decisionali senza utilizzare dati di addestramento.

2. Riformulazione binaria

Perché il problema originale multiclasse della copertura forestale è stato riformulato come classificazione binaria?

A. La regressione logistica non può essere utilizzata per problemi multiclasse.

B. Per semplificare l’interpretazione delle metriche e delle frontiere di decisione.
C. Il dataset contiene soltanto due tipi di copertura significativi.
D. Per evitare la standardizzazione delle caratteristiche.

3. Definizione della classe positiva (rilevamento degli habitat)

Nel compito di rilevamento dell’habitat Spruce/Fir, che cosa rappresenta la classe positiva (1)?

A. Qualsiasi area forestale.
B. Aree situate a quote elevate.
C. Luoghi ricoperti da foreste Spruce/Fir.
D. Luoghi con influenza antropica minima.

4. Falsi positivi nel rilevamento di NO_2

Nel compito di classificazione dei superamenti di NO_2, che cosa rappresenta un falso positivo?

A. Identificare correttamente una qualità dell’aria non sicura.
B. Prevedere una qualità dell’aria sicura quando NO_2 supera la soglia.
C. Prevedere un superamento di NO_2 quando l’aria è in realtà sicura.
D. Un errore durante l’addestramento del modello.

5. Gravità degli errori di classificazione (rilevamento degli habitat)

Nel rilevamento degli habitat, quale errore è generalmente considerato più rischioso dal punto di vista ambientale, secondo lo scenario discusso?

A. Falso negativo
B. Falso positivo
C. Vero negativo
D. Vero positivo

6. Scelta della metrica

Quale metrica è più appropriata se l’obiettivo principale consiste nell’individuare il maggior numero possibile di habitat Spruce/Fir realmente presenti?

A. Accuracy

B. Precision
C. Recall
D. Specificità

7. Precision rispetto a recall

In quale situazione la precision è più importante del recall?

A. Quando un habitat è molto costoso.
B. Quando i falsi allarmi comportano costose ispezioni sul campo.
C. Quando il dataset è perfettamente bilanciato.
D. Quando si visualizzano le frontiere decisionali.

8. Suddivisione addestramento/test

Qual è lo scopo principale della suddivisione dei dati in un set di addestramento e uno di test?

A. Ridurre la dimensione del dataset.
B. Ottimizzare gli iperparametri.
C. Stimare la capacità di generalizzazione su dati mai visti.
D. Migliorare la visualizzazione.

9. Underfitting e overfitting

Un modello ottiene ottimi risultati in addestramento, ma scarsi risultati sul test. Di quale fenomeno si tratta?

A. Underfitting
B. Overfitting
C. Sbilanciamento delle classi
D. Errore casuale

10. Visualizzazione delle frontiere decisionali

Perché nelle esercitazioni sono state visualizzate frontiere decisionali in due dimensioni usando la PCA?

A. Per migliorare l’accuratezza del modello.
B. Per ridurre l’overfitting.

C. Per rendere interpretabile il comportamento di un modello ad alta dimensionalità.
D. Per sostituire le metriche di valutazione.

Pesi suggeriti per la valutazione

- Quiz: 20%
- Esercitazione pratica di classificazione (notebook/codice): 50%
- Relazione scritta breve: 30%

Riferimenti bibliografici e letture di approfondimento

[1] C. O. De Burgh-Day and T. Leeuwenburg, “Machine learning for numerical weather and climate modelling: A review,” Nov. 14, 2023, *Copernicus Publications*. doi: 10.5194/gmd-16-6433-2023.

[2] M. Karmoude *et al.*, “Machine learning for air quality prediction and data analysis: Review on recent advancements, challenges, and outlooks,” Nov. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.scitotenv.2025.180593.

[3] G. O. Ojwang *et al.*, “An integrated hierarchical classification and machine learning approach for mapping land use and land cover in complex social-ecological systems,” *Frontiers in Remote Sensing*, vol. 4, 2023, doi: 10.3389/frsen.2023.1188635.

[4] K. S. Bobba, K. K, V. K. Sai, D. Bugga, and V. M. S. Bolla, “AI-Enhanced Acoustic Analysis for Comprehensive Biodiversity Monitoring and Assessment,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.12897

[5] T. M. . Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[6] A. E. Maxwell, T. A. Warner, and F. Fang, “Implementation of machine-learning classification in remote sensing: An applied review,” May 03, 2018, *Taylor and Francis Ltd.* doi: 10.1080/01431161.2018.1433343.

[7] M. J. Kochenderfer *et al.*, *Decision Making Under Uncertainty*. The MIT Press, 2015. doi: 10.7551/mitpress/10187.001.0001.

[8] F. K. Ewald, L. Bothmann, M. N. Wright, B. Bischl, G. Casalicchio, and G. König, “A Guide to Feature Importance Methods for Scientific Inference,” Aug. 2024, doi: 10.1007/978-3-031-63797-1_22.

[9] Partha Protim Roy, Md. Shahriar Abdullah, and Iqtiar Md. Siddique, “Machine learning empowered geographic information systems: Advancing Spatial analysis and decision making,” *World Journal of Advanced Research and Reviews*, vol. 22, no. 1, pp. 1387–1397, Apr. 2024, doi: 10.30574/wjarr.2024.22.1.1200.

[10] S. Zhong *et al.*, “Machine Learning: New Ideas and Tools in Environmental Science and Engineering,” *Environ. Sci. Technol.*, p. acs.est.1c01339, Aug. 2021, doi: 10.1021/acs.est.1c01339.

[11] T. F. Deressu, A. K. Bojer, T. G. Debelee, W. G. Negera, S. Nadarajah, and K. W. Gebissa, “Enhancing land use and land cover classification with deep learning-based satellite imagery segmentation,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 144, Nov. 2025, doi: 10.1016/j.jag.2025.104839.

[12] A. Elsayed, J. Levison, A. Binns, M. Larocque, and P. Goel, “Regression-based machine learning models for nitrate and chloride prediction in surface water in a small agricultural sand plain sub-watershed in southwestern Ontario, Canada,” *Front. Environ. Sci.*, vol. 13, 2025, doi: 10.3389/fenvs.2025.1543852.

[13] T. Vo-Van, A. Nguyen-Hai, M. V. Tat-Hong, T. Nguyen-Trang, and F. Gomariz, “A New Clustering Algorithm and Its Application in Assessing the Quality of Underground Water,” *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/6458576.

[14] *Machine Learning in Earth, Environmental and Planetary Sciences*. Elsevier, 2023. doi: 10.1016/C2022-0-00549-3.

[15] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” 2011. [Online]. Available: http://scikit-learn.sourceforge.net.

[16] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, Inc., 2023.