UNITÀ IA.5

Fondamenti di Machine Learning Esplicabile

Impareremo a costruire modelli di classificazione e regressione che non solo predicono, ma spiegano il perché delle loro decisioni. Questa unità didattica vi guiderà attraverso i concetti fondamentali del machine learning supervisionato, dalla teoria alla pratica con scikit-learn e Dataiku, con particolare attenzione all'interpretabilità dei modelli.

Cos'è il Machine Learning?

Definizione Formale

> *"Il campo di studio che dà ai computer l'abilità di imparare senza essere stati esplicitamente programmati."*
> **- Arthur Samuel, 1959**

Definizione Intuitiva

È il processo di insegnare a un computer a riconoscere **schemi (pattern)** nei dati, in modo che possa fare previsioni su dati nuovi e mai visti prima. Come un bambino che impara a distinguere un gatto da un cane dopo aver visto molti esempi, gli algoritmi di machine learning apprendono dalle esperienze passate per generalizzare a situazioni future.

Il machine learning elimina la necessità di programmare esplicitamente ogni possibile scenario, permettendo ai sistemi di adattarsi autonomamente a nuove situazioni basandosi sui pattern riconosciuti nei dati di addestramento.

Le 4 Tipologie di Apprendimento

Il machine learning si divide in quattro paradigmi principali, ciascuno adatto a diversi tipi di problemi e disponibilità di dati etichettati.

Supervisionato

L'algoritmo impara da dati **etichettati**. Ogni esempio ha una "risposta corretta" nota (target).

Esempio: Classificazione di email come "spam" o "non spam" basandosi su migliaia di email già etichettate.

Non Supervisionato

L'algoritmo esplora dati **non etichettati** per trovare strutture nascoste, senza una risposta predefinita.

Esempio: Raggruppare i clienti in segmenti (cluster) basandosi solo sul loro comportamento d'acquisto.

Semi-Supervisionato

Combina una **piccola quantità** di dati etichettati con una **grande quantità** di dati non etichettati.

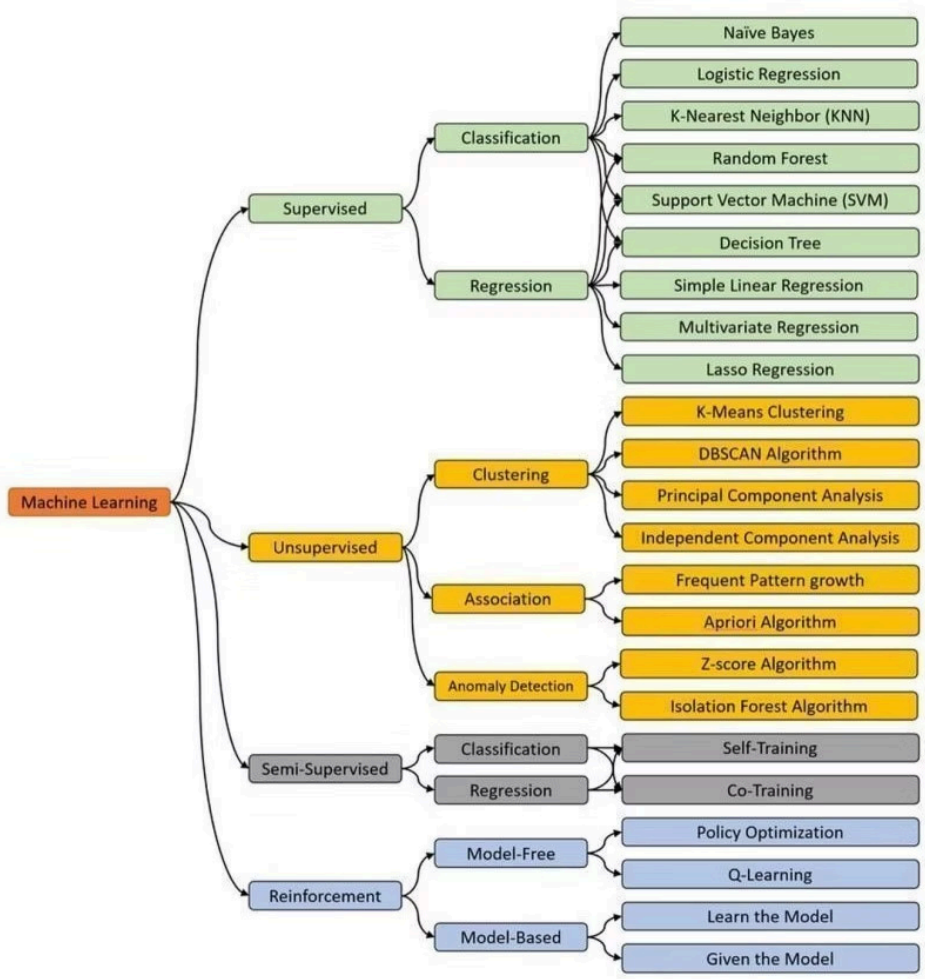
Esempio: Diagnosi medica: poche lastre refertate da medici (costose) usate per etichettare migliaia di lastre grezze.

Per Rinforzo

L'algoritmo impara attraverso **tentativi ed errori**, ricevendo ricompense o penalità per le azioni intraprese in un ambiente.

Esempio: Un agente che impara a giocare a scacchi migliorando le proprie mosse in base ai risultati delle partite (vittoria/sconfitta).

Una Panoramica sugli Algoritmi di Machine Learning



Il Machine Learning si avvale di una vasta gamma di algoritmi, ciascuno progettato per risolvere specifici problemi di apprendimento. Questi algoritmi possono essere ampiamente classificati in base alla tipologia di dati e al tipo di problema da risolvere.

Focus: Apprendimento Supervisionato

Nell'apprendimento supervisionato, la natura della variabile target determina il tipo di problema che stiamo affrontando. Questa distinzione fondamentale guida la scelta dell'algoritmo e delle metriche di valutazione.

Classificazione

L'obiettivo è predire un'**etichetta di classe** discreta.

Output: Categoria (Discreto)

Domanda: "Quale?" o "Sì/No?"

Esempio: Churn (Sì/No)

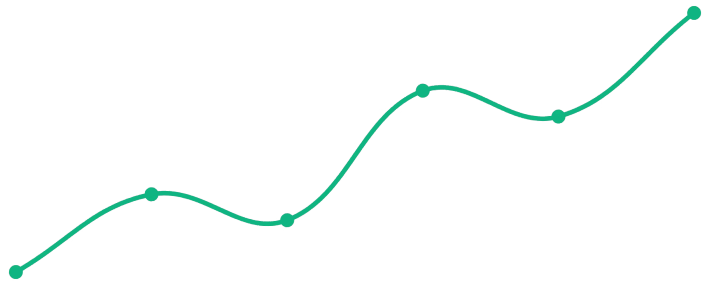
Regressione

L'obiettivo è predire un **valore numerico** continuo.

Output: Numero (Continuo)

Domanda: "Quanto?"

Esempio: Durata Noleggio (minuti)



Il Workflow di un Progetto ML

Ogni progetto di machine learning segue un percorso strutturato che trasforma un'esigenza di business in un modello predittivo operativo. Comprendere questo flusso è fondamentale per gestire efficacemente progetti di data science.

Definizione Problema

Comprendere l'obiettivo di business e cosa si vuole predire. Questa fase richiede una stretta collaborazione con gli stakeholder per tradurre esigenze aziendali in obiettivi analitici misurabili.

Raccolta Dati

Acquisizione dati da database, API o file esterni. La qualità e quantità dei dati raccolti determina il limite superiore delle performance del modello.

Preparazione Dati

Pulizia, Feature Engineering e split train/test. Spesso questa fase richiede il 70-80% del tempo totale del progetto, ma è cruciale per il successo del modello.

Addestramento

Scelta dell'algoritmo e training del modello (fit). Il modello impara i pattern nei dati di training che userà per fare previsioni su dati nuovi.

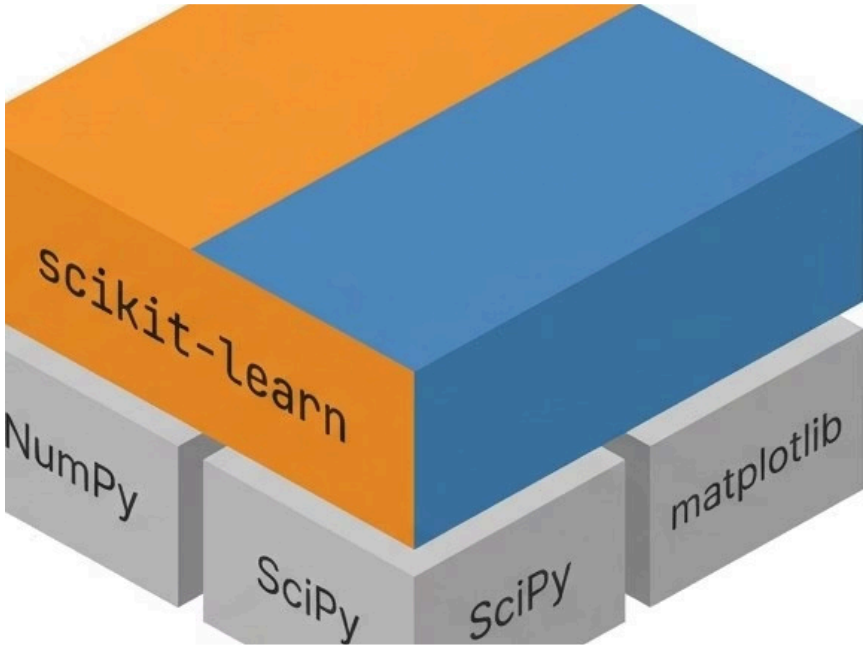
scikit-learn

Machine Learning in Python

Strumenti semplici ed efficienti per l'analisi predittiva dei dati. Scikit-learn è la libreria di riferimento per il machine learning in Python, offrendo implementazioni ottimizzate degli algoritmi più utilizzati nell'industria e nella ricerca.

VERSIONE 1.8.0

Caratteristiche Chiave di scikit-learn



Strumenti Efficienti per l'Analisi Predittiva

Semplicità

API coerente e intuitiva che riduce la curva di apprendimento. Tutti gli algoritmi seguono lo stesso pattern: fit, predict, transform.

Fondamenta Solide

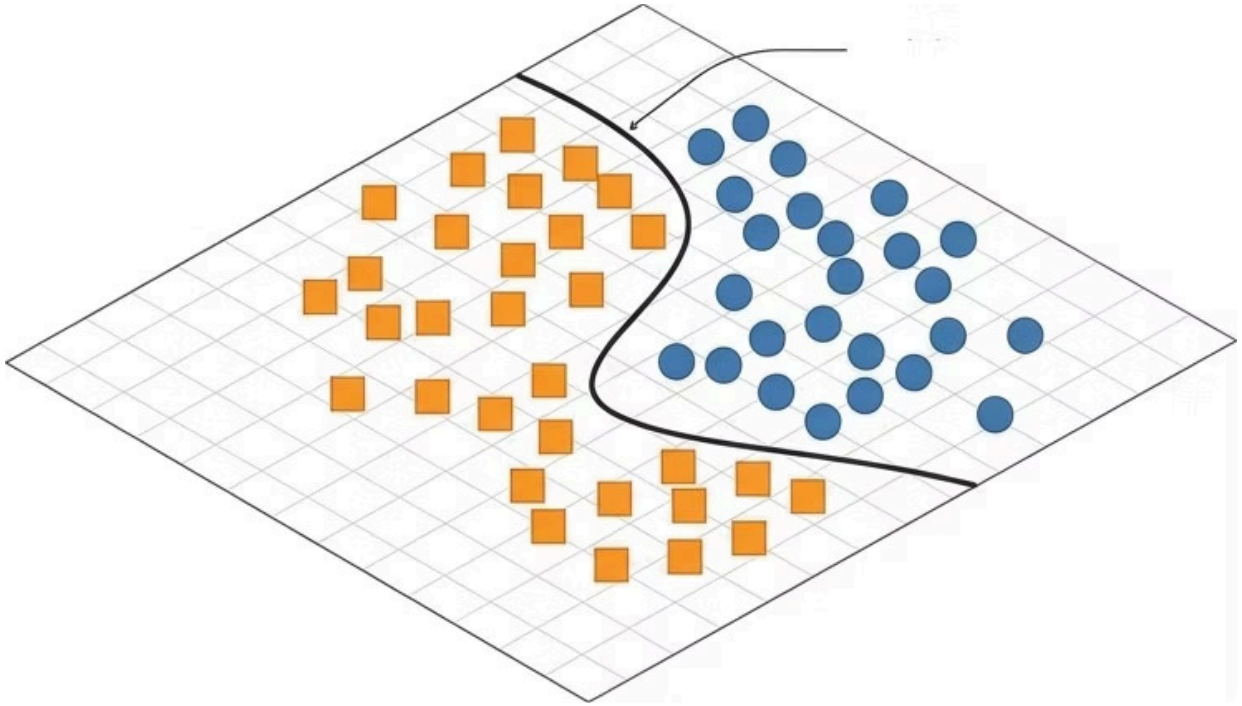
Costruito su NumPy, SciPy e matplotlib, garantendo performance ottimali e integrazione perfetta con l'ecosistema Python scientifico.

Licenza Open Source

Utilizzabile commercialmente grazie alla licenza BSD permissiva, che consente modifiche e distribuzione senza restrizioni.

Classificazione

La classificazione identifica a quale categoria appartiene un oggetto, assegnandolo a una tra diverse classi predefinite basandosi sulle sue caratteristiche.



Applicazioni

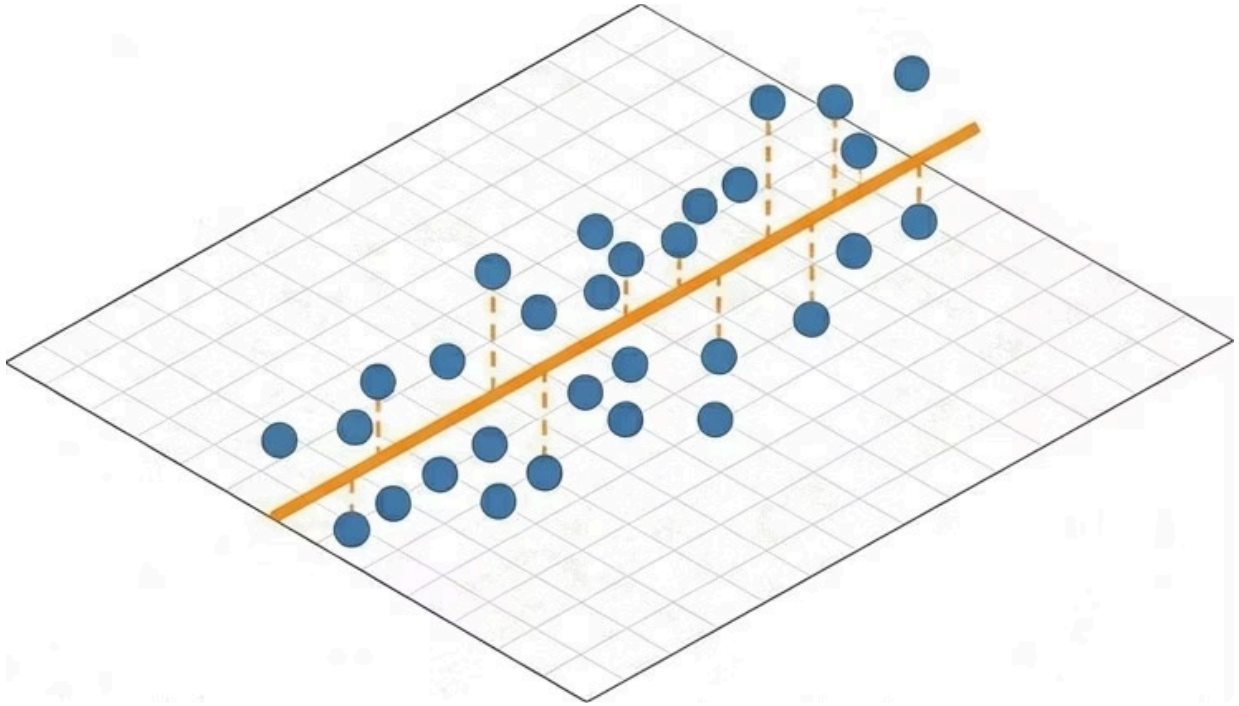
- Rilevamento spam nelle email
- Riconoscimento immagini e computer vision
- Diagnosi medica automatizzata
- Sentiment analysis nei social media

Algoritmi Principali

- Gradient Boosting
- Nearest Neighbors
- Random Forest
- Logistic Regression

Regressione

La regressione prevede un attributo a valore continuo associato a un oggetto, stimando valori numerici basandosi sulle relazioni apprese dai dati di training.



Applicazioni

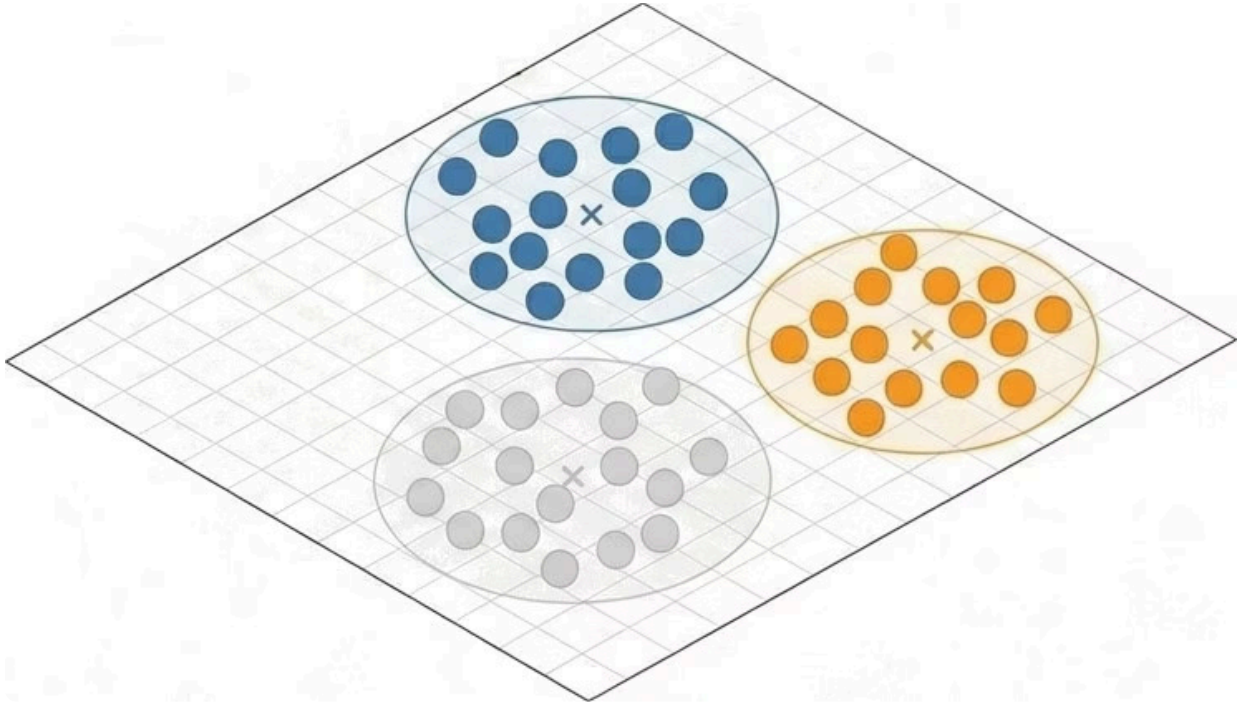
- Risposta ai farmaci e dosaggi personalizzati
- Previsione prezzi delle azioni
- Stima consumi energetici
- Valutazione immobiliare

Algoritmi Principali

- Gradient Boosting
- Nearest Neighbors
- Random Forest
- Ridge Regression

Clustering

Il clustering raggruppa automaticamente oggetti simili in insiemi, scoprendo strutture nascoste nei dati senza bisogno di etichette predefinite.



Applicazioni

- Segmentazione dei clienti per marketing mirato
- Raggruppamento risultati di esperimenti scientifici
- Rilevamento anomalie nei dati
- Compressione immagini

Algoritmi Principali

- K-Means
- HDBSCAN
- Hierarchical Clustering
- DBSCAN

Riduzione della Dimensionalità

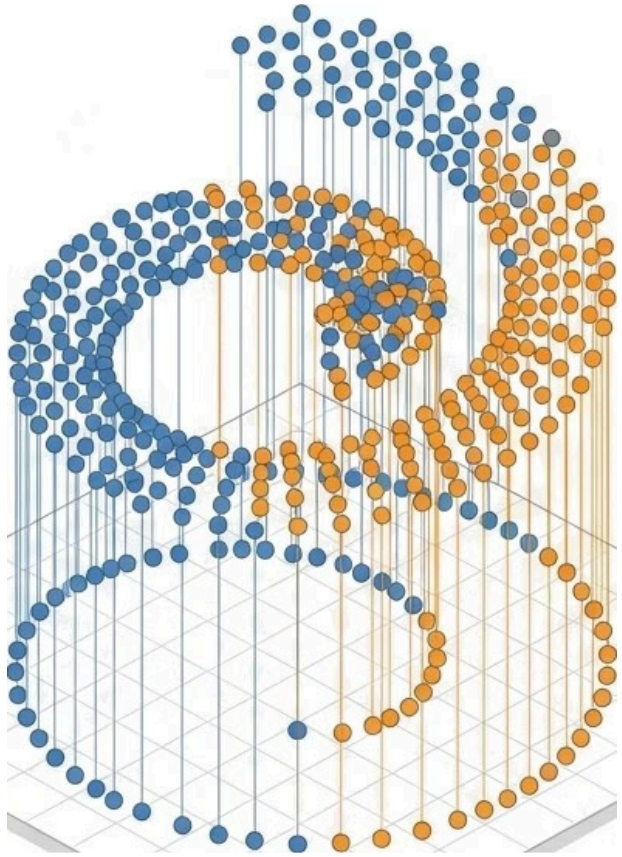
La riduzione della dimensionalità riduce il numero di variabili da considerare, mantenendo le informazioni più rilevanti e facilitando visualizzazione e calcolo.

Applicazioni

- Visualizzazione dei dati ad alta dimensionalità
- Maggiore efficienza di calcolo riducendo feature ridondanti
- Rimozione del rumore dai dati
- Compressione dati mantenendo informazioni chiave

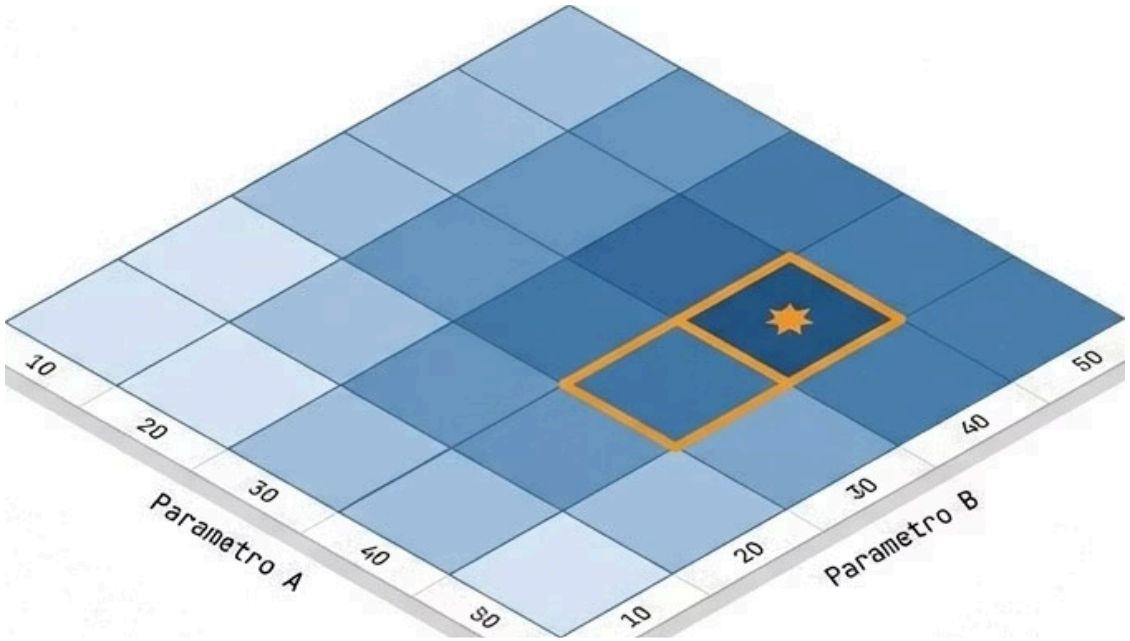
Algoritmi Principali

- PCA (Principal Component Analysis)
- Feature Selection
- Non-negative Matrix Factorization
- t-SNE



Selezione del Modello

La selezione del modello confronta, valida e sceglie parametri e modelli ottimali attraverso tecniche sistematiche di valutazione e ottimizzazione.



Applicazioni

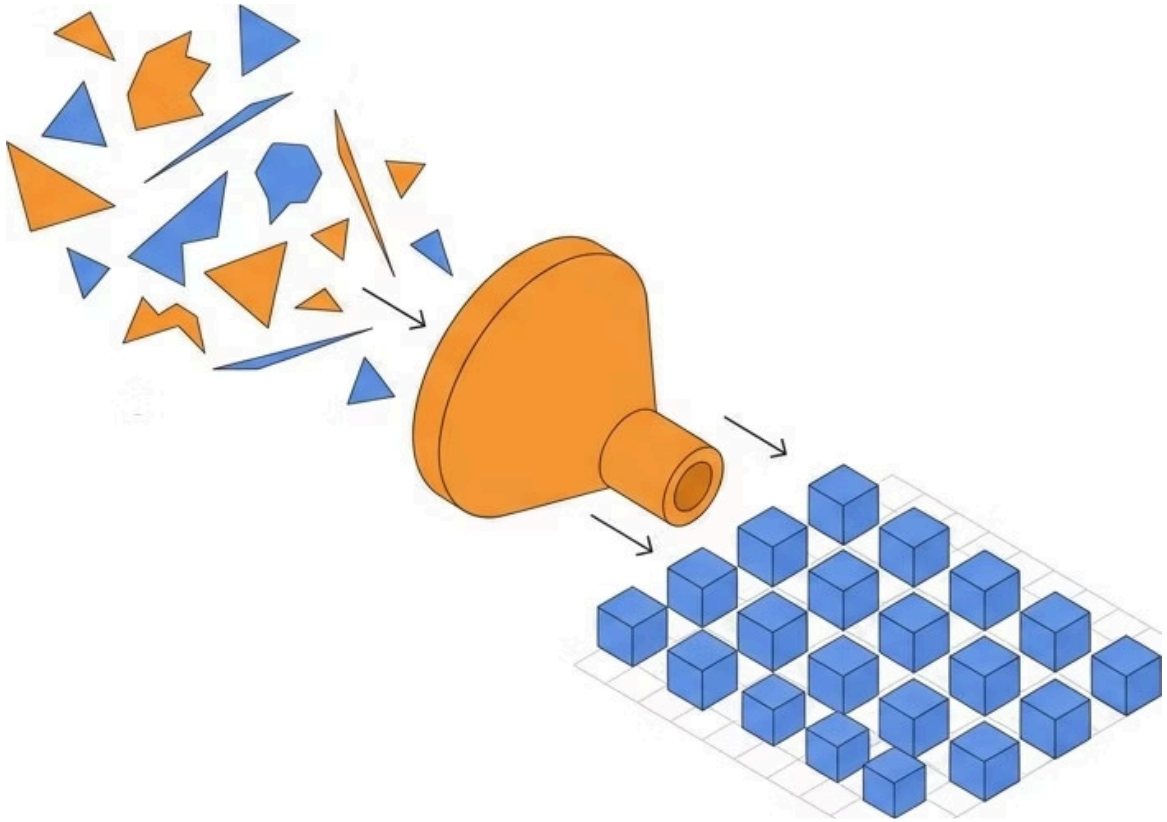
Migliore accuratezza tramite il tuning dei parametri. Permette di identificare la configurazione ottimale del modello e prevenire overfitting attraverso validazione rigorosa.

Tecniche Principali

- Grid Search per ricerca esaustiva iperparametri
- Cross Validation per valutazione robusta
- Metrics per misurare performance
- Random Search per ricerca efficiente

Pre-elaborazione

La pre-elaborazione trasforma i dati grezzi in un formato adatto all'apprendimento automatico, attraverso estrazione di feature e normalizzazione.



Applicazioni

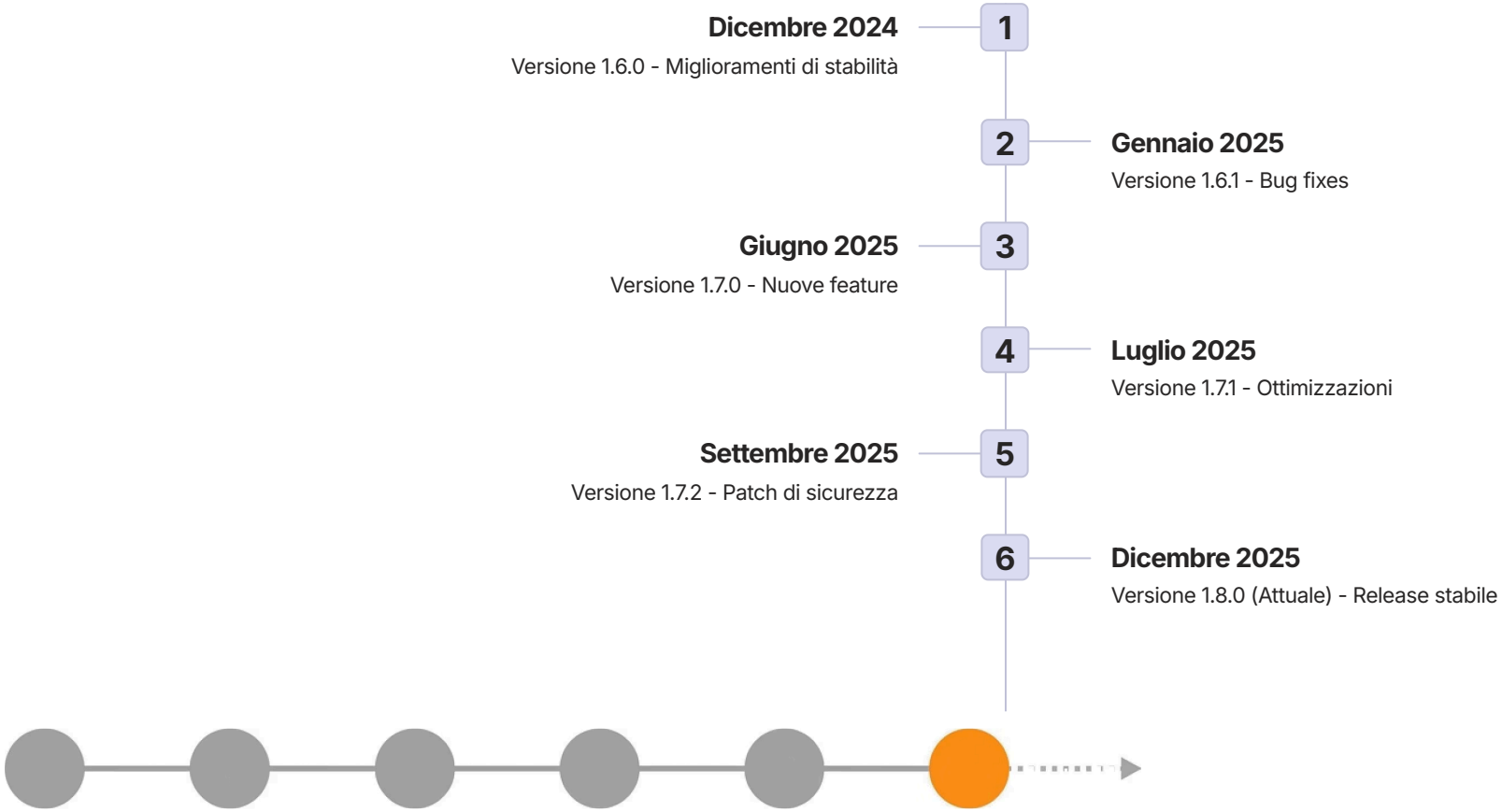
Trasformare dati di input (come il testo) per l'uso con algoritmi di machine learning. Essenziale per garantire che tutte le feature abbiano scale comparabili e per gestire dati categorici.

Tecniche Principali

- Preprocessing (scaling, encoding)
- Feature Extraction da dati complessi
- Imputation per valori mancanti
- Text vectorization (TF-IDF, Count)

Evoluzione di scikit-learn

La roadmap di sviluppo mostra l'impegno continuo della community per migliorare performance, stabilità e funzionalità della libreria.



Futuro: Sviluppo in corso per la versione 1.9 con focus su nuovi algoritmi e miglioramenti delle performance.

Rappresentazione dei Dati

In scikit-learn, i dati sono organizzati in strutture matriciali standardizzate che separano le caratteristiche di input dalla variabile target da predire.

X: Feature Matrix

Una matrice bidimensionale di dimensioni **[n_samples, n_features]**. Ogni riga è un'osservazione (es. un'ora specifica), ogni colonna è una caratteristica (es. temperatura, umidità).

Ora 1	25.2	60	12
Ora 2	24.8	62	10
...	...	...	...
Ora n	18.5	85	5

y: Target Vector

Un vettore monodimensionale di lunghezza **[n_samples]**. Contiene il valore che vogliamo prevedere (es. numero di bici noleggiate in ogni ora).

Ora 1	154
Ora 2	128
...	...
Ora n	45

Le righe di X e y devono corrispondere: la riga i-esima di X contiene le feature per la previsione della riga i-esima di y.

Le API di Scikit-Learn

Tutti gli oggetti in scikit-learn (modelli e preprocessori) sono chiamati **Estimator** e condividono la stessa interfaccia coerente. Questa uniformità semplifica enormemente l'apprendimento e l'uso della libreria.

model.fit(X, y)

Allenamento (Training)

Il cuore dell'apprendimento. L'algoritmo analizza i dati (X) e le etichette (y) per imparare i parametri interni (es. i coefficienti di una retta o la struttura di un albero decisionale).

model.predict(X)

Predizione

Usato dai modelli supervisionati. Applica i parametri imparati durante il training per stimare l'etichetta di nuovi dati mai visti prima.

model.transform(X)

Trasformazione

Usato dai preprocessori. Modifica i dati basandosi sulle statistiche imparate durante il fit (es. normalizzazione con media e deviazione standard calcolate sul training set).

L'API Estimator

L'Estimator è l'oggetto fondamentale di scikit-learn che incapsula sia l'algoritmo di apprendimento che i parametri appresi dal modello durante il training.

Cos'è un Estimator?

È l'oggetto fondamentale di scikit-learn. Incapsula l'algoritmo di apprendimento e i parametri appresi dal modello. Ogni algoritmo (LinearRegression, RandomForest, StandardScaler) è implementato come un Estimator.

Durante l'istanziazione, configuriamo gli **iperparametri** - le impostazioni che controllano *come* il modello impara. Questi sono diversi dai parametri del modello (come i coefficienti), che vengono appresi durante il training.

Esempio di Istanziazione

```
from sklearn.linear_model import LinearRegression

# Istanziazione dell'Estimator
# Configurazione degli Iperparametri
model = LinearRegression(
   fit_intercept=True,
   n_jobs=-1
)

# L'oggetto è pronto, ma non ha
# ancora "visto" nessun dato.
```

Gli iperparametri come `fit_intercept` (calcolare l'intercetta?) e `n_jobs` (quanti core CPU usare) sono impostati **prima del training** e definiscono il comportamento dell'algoritmo.

Iperparametri di Default

Senza una configurazione esplicita, ogni Estimator di scikit-learn viene istanziato con un set di iperparametri predefiniti, pensati per offrire un buon bilanciamento tra prestazioni e generalizzabilità in molti scenari comuni.

Comprendere i Valori Predefiniti

I valori di default sono cruciali perché influenzano direttamente il comportamento del modello se non vengono sovrascritti. È fondamentale conoscerli per capire come il modello funzionerà e quando è necessario personalizzarli:

- **fit_intercept=True**: Il modello calcolerà un termine di intercetta (bias), permettendo alla linea di regressione di non passare per l'origine. Questo è spesso desiderabile a meno che non ci sia una ragione specifica per forzare il passaggio per lo zero.
- **copy_X=True**: Le matrici X di input verranno copiate, evitando modifiche involontarie ai dati originali durante il processo di training del modello.
- **n_jobs=None**: Disabilita il parallelismo per default (usa 1 core CPU). Impostarlo a un valore intero (es. 4) o a -1 (per usare tutti i core disponibili) può accelerare il training per algoritmi che lo supportano.
- **positive=False**: Permette ai coefficienti di regressione di assumere valori sia positivi che negativi. Impostarlo a `True` forzerebbe coefficienti positivi, utile in scenari dove una relazione negativa è teoricamente impossibile.

Esempio: `LinearRegression` di default

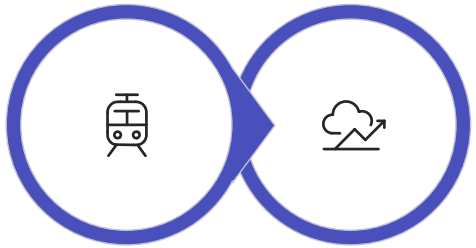
```
from sklearn.linear_model import LinearRegression

model = LinearRegression(
    fit_intercept=True,
    copy_X=True,
    n_jobs=None,
    positive=False
)

# Questi sono i valori che il modello
# userà se non specificati esplicitamente
# durante l'istanziazione.
```

Il Paradigma Fit & Predict

Il workflow fondamentale di scikit-learn si basa su due operazioni distinte e sequenziali: fit (apprendimento) e predict (inferenza).



Fit **Predict**

model.fit()

Il modello analizza i dati di training per trovare le relazioni interne (es. coefficienti della retta).

Input: X_train, y_train

Output: Modello Addestrato

model.predict()

Il modello applica ciò che ha imparato su nuovi dati per generare stime.

Input: X_new

Output: y_pred

```
# 1. Training: Il modello impara
model.fit(X_train, y_train)

# 2. Inference: Il modello prevede
predictions = model.predict(X_test)
```

Transformers e Preprocessing

I Transformer sono oggetti specializzati che modificano i dati per renderli adatti all'apprendimento automatico, seguendo un'interfaccia coerente con gli altri Estimator.

L'Interfaccia Transformer

I Transformer modificano i dati per renderli adatti all'apprendimento. Seguono un'interfaccia coerente:

StandardScaler (Scaling)

Standardizza le feature rimuovendo la media e scalando a varianza unitaria.

$z = (x - \mu) / \sigma$

Questo garantisce che tutte le feature abbiano scale comparabili, evitando che feature con valori più grandi dominino l'apprendimento.

OneHotEncoder (Encoding)

Trasforma feature categoriche in vettori binari.

Winter	1	0	...
Spring	0	1	...

01

estimator.fit(X)

Calcola i parametri necessari dai dati (es. media e deviazione standard).

02

estimator.transform(X)

Applica la trasformazione ai dati utilizzando i parametri appresi.

03

estimator.fit_transform(X)

Esegue entrambi i passaggi in sequenza (ottimizzato).

Pipelines

Le Pipeline concatenano trasformazioni e modelli in un unico oggetto coerente, semplificando il workflow e prevenendo errori comuni come il data leakage.

Da Python a Dataiku

L'integrazione tra Python e Dataiku DSS combina la flessibilità del codice con la governance e l'orchestrazione di una piattaforma enterprise, offrendo il meglio dei due mondi.

Python Ecosystem

- Librerie standard (Scikit-Learn, Pandas)
- Massima flessibilità algoritmica
- Ideale per prototipazione rapida

Dataiku DSS

- Orchestrazione visuale del flusso
- Connessione ai dati gestita
- Versioning e messa in produzione

Python Recipes: Il Meglio dei Due Mondi

Le Python Recipes permettono di scrivere codice Python personalizzato che si integra perfettamente nel flow Dataiku, beneficiando della gestione dati, del versioning e del deployment automatico della piattaforma.

Questo approccio consente ai data scientist di utilizzare gli strumenti che conoscono meglio (scikit-learn, pandas) mantenendo i vantaggi della governance aziendale di Dataiku.

Python Recipes in Dataiku

Uno script Python in Dataiku segue una struttura standardizzata che gestisce input, logica e output in modo integrato con la piattaforma.

Input

Lettura dei dataset di input dalla piattaforma Dataiku

```
import dataiku
import pandas as pd

# Leggere il dataset di input
input_ds = dataiku.Dataset("bike_trips")
df = input_ds.get_dataframe()
```

Logic

Logica Python standard utilizzando librerie come Pandas e Scikit-Learn

```
# Logica Python standard
# (Pandas, Scikit-Learn, etc.)
# Esempio: Calcolo durata media
df['duration_min'] = df['duration'] / 60
result_df = df.groupby('city').mean()
```

Output

Scrittura dei risultati nel dataset di output con schema inferito automaticamente

```
# Scrivere nel dataset di output
output_ds = dataiku.Dataset("city_stats")
# Scrive i dati e inferisce lo schema
output_ds.write_with_schema(result_df)
```

Questo pattern standardizzato garantisce che il codice Python sia sempre compatibile con il resto del flow Dataiku, permettendo una collaborazione fluida tra sviluppatori e utenti della piattaforma visuale.