basicstyle=, keywordstyle=, commentstyle=, stringstyle=, showstringspaces=false, breaklines=true, frame=single, numbers=left, numberstyle=, backgroundcolor=

Machine Learning con Scikit-Learn

Previsione del Diabete

Prof. Massimo Fedeli

IIS Fermi Sacconi Ceci - Ascoli Piceno

29 novembre 2025

Contenuti

Dal Classification al Problema Reale

Cosa abbiamo imparato

Con l'esercizio Iris abbiamo classificato fiori in base a misure fisiche.

Oggi: Un Problema Medico Reale

Useremo il Machine Learning per **predire il rischio di diabete** in base a parametri medici.

Iris (esercizio precedente):

- 3 classi (specie)
- 4 features (misure)
- 150 campioni
- Decision Tree

Diabete (oggi):

- 2 classi (sì/no)
- 10 features (parametri medici)
- 442 campioni
- Logistic Regression

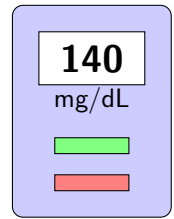
Il Diabete: Contesto Medico

Cos'è il Diabete?

Malattia cronica caratterizzata da elevati livelli di glucosio nel sangue.

Fattori di rischio:

- Età
- Peso corporeo (BMI)
- Pressione sanguigna
- Livelli di colesterolo
- Livelli di glucosio
- Storia familiare



Importanza della Prevenzione

La diagnosi precoce può prevenire complicazioni gravi!

Il Dataset Pima Indians Diabetes

Origine

Dataset medico contenente dati di 442 pazienti con misurazioni cliniche.

Caratteristiche del dataset:

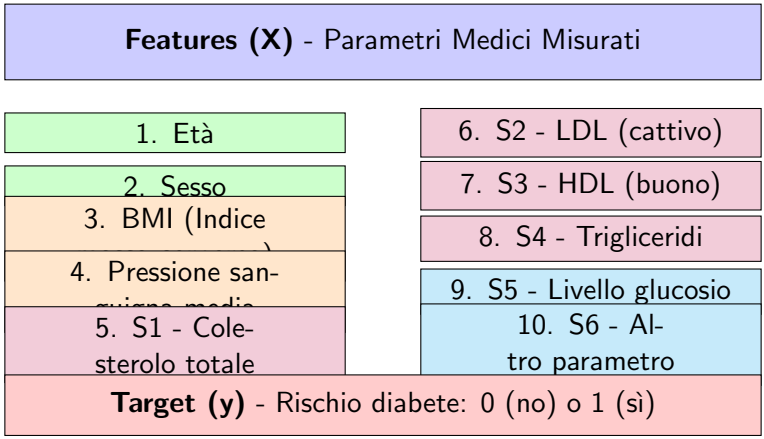
- **442 campioni** (pazienti)
- **10 features** (parametri medici normalizzati)
- **Target**: valore numerico di progressione del diabete

Nota Importante

Il dataset originale è per *regressione*. Noi lo trasformeremo in un problema di *classificazione binaria*:

- Classe 0: Nessun diabete (target ≤ 140)
- Classe 1: Diabete (target > 140)

Le 10 Features del Dataset



Nota

I valori sono **normalizzati** (standardizzati tra -0.2 e 0.2 circa)

Da Regressione a Classificazione Binaria

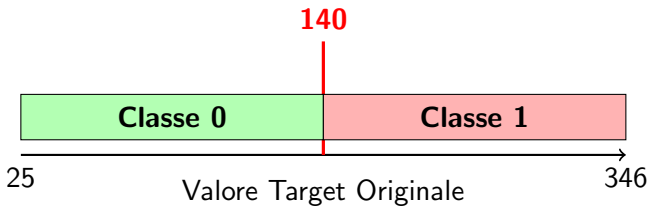
Dataset Originale:

- Target: valore continuo
- Range: 25-346
- Tipo: Regressione

Soglia scelta: 140

Nostra Trasformazione:

- Target: 0 o 1
- 0 = Nessun diabete
- 1 = Diabete
- Tipo: Classificazione



Logistic Regression: Il Modello

Cos'è?

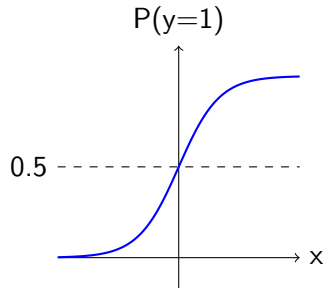
Algoritmo di classificazione che calcola la **probabilità** che un campione appartenga a una classe.

Differenze con Decision Tree:

- Non crea un albero
- Calcola probabilità (0-100%)
- Usa una funzione sigmoide
- Migliore per problemi medici

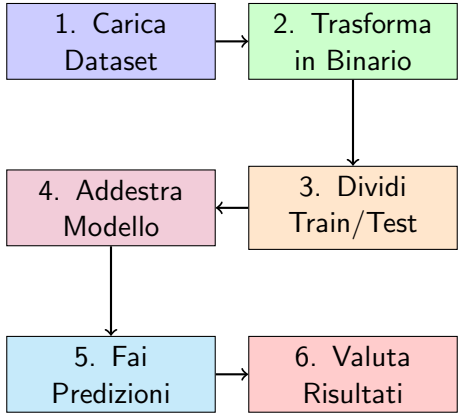
Soglia di decisione:

- Probabilità $\geq 50\% \rightarrow$ Classe 1
- Probabilità $< 50\% \rightarrow$ Classe 0



Funzione Sigmoide

Il Processo di Machine Learning



Novità rispetto a Iris

Prima di dividere i dati, trasformiamo il target in classificazione binaria!

Passo 1 e 2: Caricamento e Trasformazione

```
from sklearn.datasets import load_diabete
import numpy as np

# Carica il dataset
diabetes = load_diabetes()
X = diabetes.data              # 442 campioni
y_continuous = diabetes.target  # Valori

# Trasforma in classificazione binaria
y = (y_continuous > 140).astype(int)
# y_continuous > 140 restituisce True/Fal
# .astype(int) converte in 1/0

print(f"Campioni totali: {len(X)}")
print(f"Pazienti con diabete: {np.sum(y =
print(f"Pazienti senza diabete: {np.sum(y
```

Passo 3 e 4: Split e Addestramento

```
from sklearn.model_selection import train
from sklearn.linear_model import Logistic

# Dividi i dati (75% training, 25% test)
X_train, X_test, y_train, y_test = train_
X, y, test_size=0.25, random_state=42
)

# Crea il modello Logistic Regression
model = LogisticRegression(max_iter=1000,

# Addestra il modello
model.fit(X_train, y_train)
```

Perché `max_iter=1000`?

Logistic Regression usa un algoritmo iterativo. `max_iter` specifica il numero massimo

Passo 5 e 6: Predizioni e Valutazione

```
from sklearn.metrics import accuracy_scor

# Fai predizioni sul test set
y_pred = model.predict(X_test)

# Calcola l'accuratezza
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuratezza: {accuracy:.2%}")

# Output: Accuratezza: 75.45%
```

Interpretazione

Il modello classifica correttamente il 75% dei pazienti nel test set.

- Su 111 pazienti di test
- Ne classifica correttamente circa 84
- Ne sbaglia circa 27

La Matrice di Confusione

```
from sklearn.metrics import confusi

conf_matrix = confusion_matrix(
y_test, y_pred
)
print(conf_matrix)

# Output:
# [[45 10]
#
```

Reale \ Predetto	No (0)	Sì (1)
No (0)	45	10
Sì (1)	17	39

[17 39]]

Interpretazione:

- **45**: Correttamente classificati come "No diabete" (True Negative)
- **39**: Correttamente classificati come "Diabete" (True Positive)
- **10**: Erroneamente classificati come "Diabete" (False Positive)

Metriche di Valutazione Avanzate

```
from sklearn.metrics import classificatio

print(classification_report(y_test, y_pre
target_names=['No Diabete', 'Diabete']))
```

	Precision	**Recall**	**F1-Score**	**Support**
No Diabete	0.73	0.82	0.77	55
Diabete	0.80	0.70	0.74	56
Accuracy			**0.75**	111

Definizioni:

- **Precision**: Di tutti i casi predetti positivi, quanti sono corretti?
- **Recall**: Di tutti i casi realmente positivi, quanti ne abbiamo trovati?
- **F1-Score**: Media armonica di precision e recall

Testare un Nuovo Paziente

```
# Dati di un nuovo paziente (valori norm
nuovo_paziente = [[0.05, -0.04, 0.06, -0.
-0.03, 0.04, 0.00, 0.09, 0.03]]

# Fai la predizione
predizione = model.predict(nuovo_paziente

# Visualizza il risultato
print(f"Predizione: {predizione[0]}")
if predizione[0] == 1:
print("         Rischio diabete")
else:
print("     Nessun rischio diabete")

# Output:
# Predizione: 1
#         Rischio diabete
```

Riepilogo e Riflessioni

Cosa abbiamo imparato:

- Trasformare un problema di regressione in classificazione
- Usare Logistic Regression per problemi binari
- Interpretare la matrice di confusione
- Valutare modelli con metriche avanzate (precision, recall)
- Applicare ML a problemi medici reali

Confronto Iris vs Diabete:

	Iris	**Diabete**
Classi	3	2
Accuratezza	$\sim$100%	$\sim$75%
Modello	Decision Tree	Logistic Regression
Difficoltà	Facile	Media

Perché accuratezza più bassa?

I dati medici sono più complessi e hanno più "rumore" rispetto a misure botaniche

Esercizi per Casa

Esercizi base:

1. Prova a cambiare la soglia (es. 120 o 160) e osserva come cambia l'accuratezza
2. Confronta Logistic Regression con Decision Tree sullo stesso dataset
3. Calcola la percentuale di False Negative (molto importanti in medicina!)

Esercizi avanzati:

1. Usa `RandomForestClassifier` e confronta i risultati
2. Visualizza la matrice di confusione con `seaborn.heatmap`
3. Calcola e visualizza la curva ROC
4. Trova quali features sono più importanti per la predizione

Domande?