Michelangelo Rossi – Matr. 152633

Fondamenti di Machine Learning

30 gennaio 2024

Differentiated Thyroid Cancer Recurrence

Report per l'esame di Fondamenti di Machine Learning

Dataset utilizzato:

https://archive.ics.uci.edu/dataset/915/differentiated+thyroid+cancer+recurrence

Abstract

Il dataset *Differentiated Thyroid Cancer Recurrence* è una raccolta di dati utilizzata per aiutare i medici a predire una possibile ricaduta di diversi tumori alla tiroide, una volta nota l'anamnesi di un paziente. I dati sono stati raccolti in 15 anni, e ogni paziente è stato seguito per almeno 10 anni.
L'obiettivo finale è quello di sviluppare un algoritmo che possa aiutare i medici ad effettuare diagnosi accurate e poter individuare facilmente pazienti potenzialmente predisposti ad una possibile ricaduta di un tumore tiroideo.
Il programma è stato scritto in linguaggio Python ed utilizza le librerie NumPy, Pandas, Pyplot e Sklearn.

1. Problema e acquisizione dei dati

Il dataset comprende 383 istanze, ognuna delle quali rappresenta un paziente, ciascuna definita da 16 attributi, tra i quali l'età, il sesso, valori di funzioni tiroidee, diversi risultati di esami diagnostici e altro.
La variabile target è *Recurred*, di tipo categorico, che indica l'avvenuta ricomparsa o meno di un tumore tiroideo; essa può assumere due valori: *Yes* oppure *No*. Di conseguenza il problema si configura come un **task di classificazione binaria**.
Sulla base di questi campioni l'obiettivo è definire un algoritmo che riesca a predire, dato un paziente, se esso è suscettibile di ricaduta o no.
A parte la feature *Age*, che è numerica, tutte le altre sono categoriche.

2. Exploratory Data Analysis

Come primo passo è stato necessario analizzare il dataset fornito: come accennato in precedenza, l'unica variabile numerica (di tipo int64) è *Age*, tutte le altre feature sono categoriche. È stato effettuato un controllo per ricercare eventuali dati mancanti, e non ne sono stati rilevati.
Successivamente tutte le feature categoriche sono state discretizzate per ricondursi a valori numerici e, successivamente, è stato manualmente diviso il *Training Set* dal *Testing Set*, optando per 80% di training e 20% di testing.
Come ultima operazione di preparazione dei dati è stato effettuato lo scalamento dei dati tramite standardizzazione base, in modo tale da avere valori delle feature con media 0 e varianza 1, tramite l'impiego dello *StandardScaler*, in modo tale da velocizzare l'addestramento dei modelli.

3. Scelta dei modelli

Una volta esaminato e preparato il dataset, sono stati scelti quattro modelli:

1. K-NN Nearest Neighbours;
2. Logistic Regression;
3. Support Vector Machine;
4. Decision Tree;

Per ogni modello sono stati identificati gli iperparametri, ognuno dei quali associato ad una lista di possibili valori.
Per il K-NN, scelto per la sua facilità di implementazione e di realizzazione, ci si è concentrati sul numero di "vicini" da considerare.
Per la Logistic Regression, metodo parametrico pensato apposta per la classificazione binaria, gli iperparametri scelti sono stati la penalità (se norma L1 o L2) e il fattore di regolarizzazione C.
Per la SVM, scelto per la robustezza al fenomeno dell'*overfitting*, si è andati invece a ricercare i valori migliori del fattore di regolarizzazione C e il tipo di kernel (lineare, polinomiale o gaussiano).
Infine per l'albero decisionale, scelto per la sua economicità di realizzazione e la sua velocità a *inference time*, l'obiettivo è stato massimizzare il guadagno di informazione, pertanto come criteri di split si andranno a valutare sia l'indice di Gini che l'entropia di Shannon.

4. Tuning degli iperparametri

Una volta definiti i modelli, i relativi iperparametri e i diversi valori che possono assumere, sono stati settati "manualmente" gli iperparametri tramite la funzione *GridSearch*. Tale metodo infatti, per ogni classe di modelli, indaga tutte le possibili combinazioni, istanziando e addestrando un modello per ciascuna combinazione, sottoponendolo a *cross validation* di tipo *K-fold* (con k = 5), e valutandone le performance in base ad una metrica scelta dall'utente, scegliendo infine il modello con il punteggio più alto.

In questo caso è stata scelta la *accuracy*, che esprime il numero di previsioni corrette sul numero di previsioni totali.

Alla fine di questo procedimento per ogni tipologia di modello è stata trovata la configurazione migliore di iperparametri e ne è stata stampata a video la sua accuratezza:

```
Model name:  K-NN
Accuracy:  0.8692226335272343
        The best choice for parameter n_neighbors:  3

Model name:  Logistic Reg.
Accuracy:  0.8825489159175041
        The best choice for parameter penalty:  l1
        The best choice for parameter C:  1

Model name:  SVM
Accuracy:  0.8988365943945003
        The best choice for parameter C:  0.01
        The best choice for parameter gamma:  0.001
        The best choice for parameter kernel:  linear

Model name:  Decision Tree
Accuracy:  0.941142252776309
        The best choice for parameter criterion:  gini

The best model is:  Decision Tree
```

Da tale analisi risulta che il miglior modello è il *Decision Tree*, e il criterio di split migliore è l'indice di Gini.

5. Cross-validation e Training

Una volta definito il modello ottimale si è andati ad effettuare una Cross-Validation utilizzando la funzione di Sklearn *cross_validate*. Si è optato per una *K-fold validation*, dividendo il training set in 5 parti e valutandone l'*accuracy* e il *f1-score*. Questi i risultati:

```
The Accuracy of the final model is:  0.9247488101533581
The F1-score of the final model is:  0.9234624645562393
```

Essendo la scelta ricaduta sull'albero decisionale, che è per sua natura molto sensibile al dataset ed è intrinsecamente più soggetto ad overfitting, prima di addestrare il modello vero e proprio si è proceduto ad effettuare una *feature selection* tramite il metodo Wrapper *SequentialFeatureSelection,* in modo da evitare il più possibile l'effetto di sovradattamento. Di 16 feature ne sono state selezionate 8.

5. Testing e risultati finali

Una volta addestrato il modello, si è proceduto alla fase di testing, utilizzando il testing set per andare a predire i valori della variabile target. Questi i risultati finali:

```
              precision    recall  f1-score   support

          No       0.94      0.98      0.96        52
         Yes       0.96      0.88      0.92        25

    accuracy                           0.95        77
   macro avg       0.95      0.93      0.94        77
weighted avg       0.95      0.95      0.95        77

Accuracy is:  0.948051948051948
Precision is:  0.9483656440178179
Recall is:  0.948051948051948
F1-score is:  0.9474597729314711
```

Una misura "visiva" dell'elevata performance predittiva dell'algoritmo può essere colta osservando l'istogramma seguente, che confronta i valori di *ground truth* presenti nel testing set e i valori della variabile target predetti dall'algoritmo:

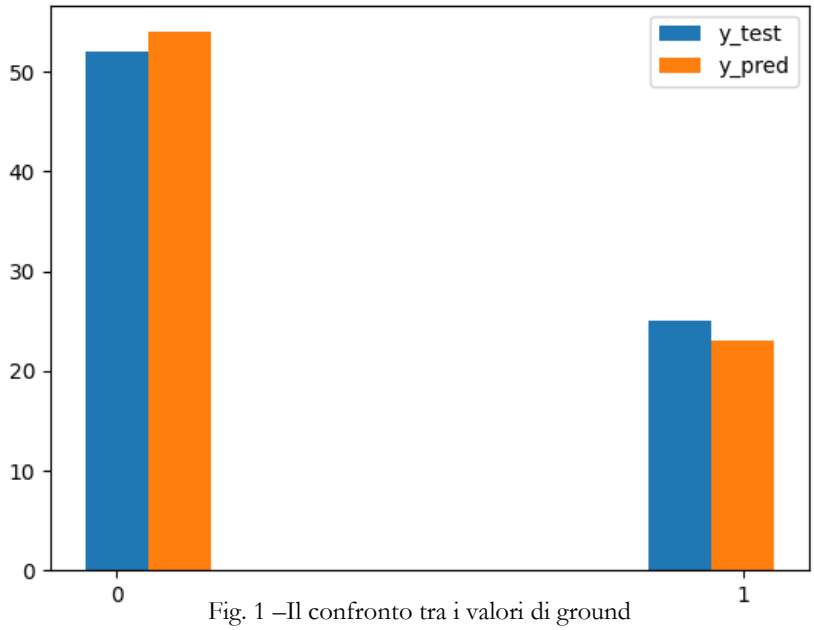

Fig. 1 –Il confronto tra i valori di ground truth (blu) e i valori predetti dall'algoritmo (arancione)