Heart Disease Prediction

Progetto di Ingegneria della Conoscenza

Ercole Thomas, 698992
t.ercole@studenti.uniba.it

Introduzione

Lo scopo di questo progetto è predire, in base alla conoscenza presente nel dataset e i valori delle feature di input, se una persona possa avere una patologia cardiaca.
Nella prima parte di questo progetto ci si occupa della classificazione, utilizzando diversi algoritmi che lavorano in modo diverso sui dati, ossia l'algoritmo K-nearest neighbors, l'Albero di decisione, la Random Forest, Boosting Adattivo, l'Histogram Gradient Boosting, la Support-Vector Machine e l'Extreme Gradient Boosting. Le prestazioni di questi classificatori vengono confrontate mediante una K-fold Cross Validation (con k=10) e sono memorizzate le seguenti metriche a fine di ogni iterata: Accuracy, Precision, Recall e F1-score. A fine della k-fold cross validation, viene restituita una media di ciascuna metrica.
Nella seconda parte di questo progetto ci si occupa dell'inferenza probabilista. Ossia, viene costruita una Belief Network (o Rete Bayesiana) che permette al sistema di rispondere a query probabilistiche definite dall'utente.

Dataset

Il dataset utilizzato, "heart.csv", contiene dati su 400 soggetti riguardo la presenza o meno di possibili patologie cardiache.

Le feature presenti nel dataset sono:

1. age
2. gender (0 = donna, 1 = uomo)
3. cp = tipologia di dolore al petto (valori 0,1,2,3)
4. trestbps = Pressione del sangue a riposo
5. chol = Colesterolo totale sierico in mg/dl
6. fbs = Glicemia a digiuno > 120 mg/dl (0 = no, 1 = si)
7. restecg = risultati elettrocardiografici a riposo (valori 0,1,2)
8. thalach = frequenza cardiaca massima raggiunta
9. exang = angina indotta da esercizio (0 = no, 1 = si)
10. oldpeak = depressione del tratto ST indotta dall'esercizio rispetto al riposo
11. slope = la pendenza del segmento ST di picco dell'esercizio
12. ca = numero di vasi principali (0-3) colorati da fluoroscopia
13. thal: 0 = normale; 1 = difetto fisso; 2 = difetto reversibile

Come feature target:

14. target = presenza di malattie cardiache nel paziente (0 = nessuna malattia e 1 = malattia)

Linguaggio di programmazione e Librerie

Per la codifica di questo progetto è stato scelto Python come linguaggio di programmazione, che mette a disposizione le librerie scikit-learn e pgmpy, ampliamente utilizzate nel progetto per via della loro messa a disposizione degli algoritmi di classificazione, della k-fold cross validation e per la costruzione della Belief Network.

Classificazione

Per il task di Classificazione sono stati utilizzati diversi algoritmi (facente tutti parte del macro-filone di algoritmi per l'Apprendimento Supervisionato) che lavorano in modo diverso sui dati:

- K-nearest neighbors, dove vengono utilizzati gli esempi di training più vicini per predire il valore target di un esempio nuovo. La predizione può essere determinata dalla moda, media o un'interpolazione delle predizioni dei k esempi più vicini. La metrica di default per determinare la vicinanza degli esempi è la distanza Euclidea.
- Decision Tree, albero di decisione dove nei nodi interni sono presenti le condizioni applicabili agli esempi e nei nodi foglia è presente la stima puntuale della classe. Quindi partendo dalla radice, per un nuovo esempio, viene valutata ogni condizione incontrata e si segue l'arco corrispondente al risultato, fino ad arrivare alla foglia dove si assegna la classe target corrispondente. Nella fase di costruzione di un albero di decisione, le condizioni da applicare ad un nodo interno vengono scelte in base ad una misura definita da minimizzare o massimizzare.
- Random Forest, tipologia di ensemble learning dove si addestrano contemporaneamente un certo numero di alberi su un sottoinsieme casuale di feature/sottoinsieme di esempi di training e si aggregano le predizioni (attraverso un meccanismo di voto) per formare la predizione finale della foresta per il nuovo esempio.
- AdaBoost (Adaptive Boosting) è un algoritmo di apprendimento automatico per la classificazione binaria, utilizzato per migliorare la precisione di un modello debole (cioè un modello con una performance leggermente migliore di quella casuale) combinandolo con una serie di modelli più forti. In ogni iterazione, il peso dei dati di addestramento viene aggiornato in base ai loro risultati di classificazione, con i dati classificati correttamente che vengono assegnati un peso inferiore rispetto ai dati classificati erroneamente. Il modello finale è una combinazione pesata dei singoli classificatori di base, con il peso di ciascun classificatore determinato dalla sua performance sulla base dei pesi dei dati di addestramento.

- HistogramGradientBoosting è una tecnica di ensemble learning che utilizza una versione approssimata dell'algoritmo Gradient Boosting che usa l'istogramma delle feature per rendere l'algoritmo molto più veloce e meno intensivo in termini di memoria. A differenza di altri algoritmi di Boosting, come AdaBoost, HGB utilizza un albero di decisione approssimato che utilizza l'istogramma delle feature invece di una suddivisione ricorsiva basata su un singolo punto di taglio.
- Support-Vector Machine, un classificatore basato sulla funzione Kernel (che produce nuove feature a partire da quelle iniziali, ossia crea nuove dimensioni nello spazio per risolvere il problema della separabilità lineare e definire l'iperpiano). Dopo la costruzione dell'iperpiano, il classificatore definisce il margine, ossia la distanza minima degli esempi dall'iperpiano, gli esempi più vicini ad esso diventano il supporto della superficie di decisione. Scopo è quello di trovare l'iperpiano di massimo margine.

- XGB anche noto come eXtreme Gradient Boosting è un algoritmo di machine learning basato sull'idea del boosting, ovvero una tecnica che combina più modelli semplici al fine di crearne uno nuovo più complesso ma maggiormente accurato. XGB usa gli alberi di decisione, in particolare una versione migliorata dell'algoritmo di boosting chiamato Gradient Boosting. Questo algoritmo addestra una serie di alberi di decisione sequenziali, ovvero ogni albero successivo rispetto a quello attuale, correggerà gli errori del precedente così da minimizzare iterativamente l'errore commesso dal modello. Introduce una tecnica di pruning degli alberi e inoltre sfrutta regolarizzazione L1 e L2 per il problema dell'overfitting, ovvero l'alta aderenza del modello ai dati di addestramento. Questo algoritmo vanta prestazioni elevate con un alto tasso di precisione, rimarcato dall'efficiente tecnica di minimizzazione dell'errore commesso. Regolarizzazione, quindi sarà in grado di prevenire problematiche come l'overfitting, come già esplorato precedentemente

Le prestazioni di questi classificatori vengono poi confrontate mediante una K-fold Cross Validation (con k=10), ossia un algoritmo che usa dati classificati come esempi di training per valutare il modello appreso prima dell'utilizzo degli esempi di test. In particolare, la k-fold cross validation consente il riuso degli esempi per il training e per la valutazione, cioè suddivide gli esempi in fold di uguali dimensioni e, a turno, viene utilizzata una di queste per valutare il modello mentre le rimanenti lo addestrano. A turno ogni fold viene utilizzata per la validazione, poiché ci sono ripetizioni (definite da k).

A fine di ogni iterata vengono memorizzate le seguenti metriche per ognuno dei nostri algoritmi di classificazione:
Accuracy (proporzione degli esempi predetti correttamente sia positivi che negativi su tutti gli esempi), Precision(proporzione degli esempi veramente positivi su tutti gli esempi predetti come positivi), Recall (proporzione degli esempi veramente positivi su tutti i positivi effettivi) e F1-score (media armonica tra precision e recall).
A fine della k-fold cross validation, viene restituita una media di ciascuna metrica.
I risultati della valutazione dei classificatori sono:

```
Media delle metriche del KNN
Media Accuracy: 0.844423
Media Precision: 0.844803
Media Recall: 0.866548
Media f1: 0.853110

Media delle metriche del DecisionTree
Media Accuracy: 0.877051
Media Precision: 0.911464
Media Recall: 0.850378
Media f1: 0.853110

Media delle metriche del RandomForest
Media Accuracy: 0.904744
Media Precision: 0.903677
Media Recall: 0.921612
Media f1: 0.910460

Media delle metriche del HistgradientBoosting
Media Accuracy: 0.899808
Media Precision: 0.891311
Media Recall: 0.924781
Media f1: 0.904920

Media delle metriche di AdaBoost
Media Accuracy: 0.907179
Media Precision: 0.916356
Media Recall: 0.906606
Media f1: 0.909195

Media delle metriche del SVM
Media Accuracy: 0.691603
Media Precision: 0.665664
Media Recall: 0.832072
Media f1: 0.737851

Media delle metriche del ExtremeGradientBoosting
Media Accuracy: 0.912244
Media Precision: 0.914014
Media Recall: 0.920960
Media f1: 0.915425
```

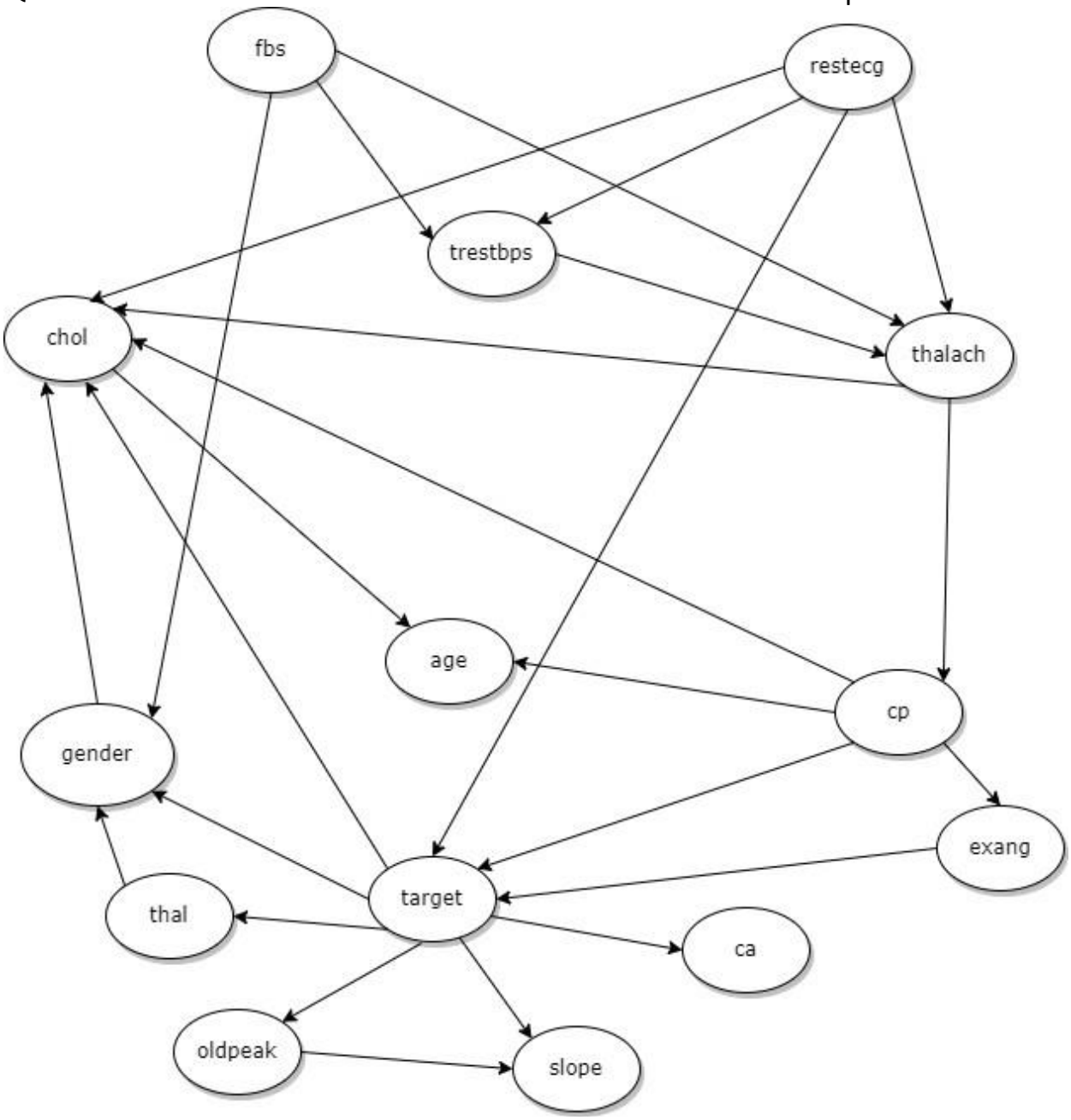
Prendendo in considerazione le performance dell'F1-score,
l'ExtremeGradientBoosting risulta essere il classificatore migliore.
Invece, tra i 3 modelli ensemble proposti RandomForest risulta essere il migliore.

Si è fissato random_state=0 ove possibile perché se il "random state" non viene fissato, il modello potrebbe produrre risultati diversi ogni volta che viene eseguito, anche con gli stessi dati di input. Fissando il "random state", invece, si garantisce che il modello utilizzi gli stessi numeri casuali in ogni esecuzione, rendendo i risultati riproducibili e consentendo una comparazione accurata tra i diversi modelli e parametri.
Riguardo agli iperparametri utilizzati dai vari classificatori, scelti attraverso diverse prove, si è optato per inserire:
Per il KNN si è optato per dare un peso ai k vicini in base alla distanza, aggiungendo weights="distance". Il metodo per il calcolo della distanza è rimasto di default (Distanza euclidea). Aggiungendo i pesi alle distanze, il valore delle metriche ha subito un discreto incremento.
Per RandomForest è stata scelta una profondità non troppo elevata(max_depth=3) in modo da evitare l'overfitting.

Inferenza

Per il task di Inferenza è stata creata una Belief Network, ossia un grafo aciclico orientato (DAG) che può essere utilizzato per modellare una base di conoscenza per poter lavorare con la conoscenza in presenza di incertezza. Poiché si lavora con l'incertezza, si ottengono delle risposte alle query che non rispondono solo con vero o falso, ma definiscono quanto è vera o quanto è falsa la query definita, ossia delle risposte probabilistiche. Queste Belief Network (o Reti Bayesiane) stabiliscono l'indipendenza condizionata nell'insieme di variabili che si hanno a disposizione. Così facendo il valore di una variabile è influenzato direttamente dalle sue variabili genitori, ossia un insieme minimale di variabili di cui è condizionatamente dipendente.

Quindi la rete viene creata e modellata sui dati che si hanno a disposizione:



Infine, viene utilizzata per rispondere a delle query probabilistiche determinate dall'utente che contengono evidenza.
L'utente va a scegliere una feature dal dataset come ipotesi e, subito dopo, definisce l'evidenza scrivendo le variabili e i rispettivi valori.

```
Seleziona una o più di queste variabili (in caso di multiple, separale con uno spazio):
age
gender
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
 >
```

Ad esempio, data P(target | fbs:0 restecg:1 thalach:165 exang:0 oldpeak:2 slope:1 ca:0 thal:2), le probabilità che una persona con queste caratteristiche abbia una patologia cardiaca sono:

```
 > fbs:0 restecg:1 thalach:165 exang:0 oldpeak:2 slope:1 ca:0 thal:2
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.3020 |
+-----------+---------------+
| target(1) |        0.6980 |
+-----------+---------------+
```