Documentazione progetto Ingegneria della Conoscenza 2021-2022

Indice

Introduzione

Questo progetto è stato realizzato per l'esame di Ingegneria della Conoscenza dagli studenti Massaro Claudio (matricola 705108), Trotti Francesco (matricola 703010) e Marino Angela (matricola 683710).
Abbiamo preso in considerazione un dominio legato ad uno dei disturbi più pericolosi per l'uomo: quello cardiaco.
Il nostro sistema è in grado di prevedere se si potrebbe essere affetti da una malattia cardiaca a seconda delle risposte fornite in input, grazie alle informazioni contenute all'interno del dataset di riferimento.

Requisiti Funzionali

Avendo progettato il programma in Python, si richiede un ambiente in grado di eseguire codice, nel nostro caso abbiamo deciso di utilizzare PyCharm poiché ci permetteva inoltre di installare le librerie esterne direttamente dall'IDE.
Relativamente alle librerie esterne importate, vediamo la necessità di installare sulla macchina:

- **pandas**, usato per la manipolazione e l'analisi dei dati;
- **scikit-learn**, per applicare i concetti del Machine Learning;
- **pybbn**, per la rete bayesiana;
- **pytholog**, per utilizzare la programmazione logica in Python;

Nel caso in cui non si utilizzi Pycharm, potrebbe essere necessario l'utilizzo del terminale. Ad esempio, su Windows ritroviamo il comando:
"python -m pip install nome_libreria"

Manuale utente

E' possibile scaricare il progetto dal seguente link https://github.com/clah865/icon_project.git
Dopo averlo scaricato si dovrà eseguire il file "***system***" su un IDE.

```
Benvenuto nel sistema per predire se sei affetto di una malattia cardiaca.
    Per capire cio' verranno somministrate una serie di domande riguardanti il tuo stile di vita e il tuo stato psicofisico.
    Ti è consigliato di rispondere con la massima serietà.'
    Grazie.

La maggior parte delle domande richiederà risposte di questo tipo (ti verrà indicato quando):
    ->si-s-yes-y
    ->no-n
```

Questo screenshot rappresenta il messaggio iniziale che vi si presenterà una volta avviato correttamente il programma.
È presente una breve descrizione in cui è esplicitato all'utente di cosa il sistema effettivamente si occupa.

```
Hai fumato almeno 100 sigarette (5 pacchetti) in tutta la tua vita? [s-n]
si
Hai mai avuto un ictus? [s-n]
no
Hai il diabete? [s-n]
no
Sei in uno stato di pre-diabete? [s-n]
si
Hai svolto attività fisica o esercizio fisico negli ultimi 30 giorni in modo diverso dal normale lavoro? [s-n]
si
Mangi frutta 1 o più volte al giorno?[s-n]
no
Mangi verdura 1 o più volte al giorno? [s-n]
si
Bevi più di 14 drink a settimana? [s-n]
si
Indica, secondo te, il tuo stato di salute da 1 a 5 [valore massimo 1, valore minimo 5]:
3
Per quanti giorni negli ultimi 30 giorni la tua salute mentale non è stata buona?
15
Per quanti giorni negli ultimi 30 giorni la tua salute fisica non è stata buona?
5
Hai difficoltà nel camminare o nel salire le scale? [s-n]
no

Secondo i dati analizzati, dovresti godere di buona salute.
```

Subito dopo, inizieranno una serie di domande, che verranno richieste sequenzialmente. Al termine, il sistema restituirà la predizione.

Scelte progettuali

Abbiamo utilizzato il dataset dal seguente link: [Heart Disease Health Indicators Dataset] per addestrare il nostro sistema.
I modelli e le tecniche utilizzate:

- Prima di passare direttamente all'apprendimento supervisionato abbiamo osservato il dataset scelto, e mediante una matrice di correlazione, abbiamo notato alcune features che risultavano poco coerenti (poiché riferiti ad un contesto statunitense) nei confronti delle features obiettivo e pertanto abbiamo effettuato *features selection.*

```
data = pd.read_csv(r".\..\dataset\heart_disease_health_indicators_BRFSS2022.csv")

data = data.drop('AnyHealthcare', axis=1)
data = data.drop('NoDocbcCost', axis=1)
data = data.drop('Education', axis=1)
data = data.drop('Income', axis=1)

data.to_csv(r".\..\dataset\heart_disease_health_indicators_BRFSS2022.csv", index = False)
```

- Calcolando accuratezza del sistema prima e dopo l'eliminazione delle features abbiamo notato una leggera diminuzione della precisione (circa 0.01), ma che abbiamo pensato fosse accettabile, al fine di ricevere risposte reali e non date a caso dall'utente.

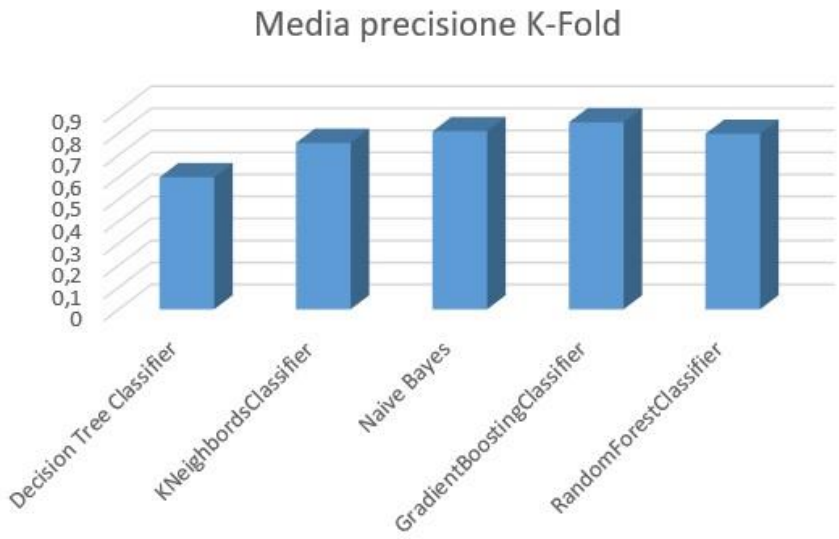
- Abbiamo utilizzato il Gradient Boosting come classificatore, per la predizione della feature obiettivo.
  Questa scelta non è stata casuale ma ci siamo basati sui seguenti dati che mostrano l'accuratezza riscontrata in 5 esecuzioni del sistema, ognuna delle quali utilizzando una K-fold cross validation, mostra la media riscontrata tra le precisioni calcolate nelle varie convalide, *deviation* indica la deviazione standard:

Model	K-Fold	Deviation
DecisionTree Classifier	0,602675	0,004
DecisionTree Classifier	0,598434	0,006
DecisionTree Classifier	0,599227	0,006
DecisionTree Classifier	0,599615	0,002
DecisionTree Classifier	0,599150	0,005
Media	0,599820	0,005
KNeighborsClassifier	0,751648	0,003
KNeighborsClassifier	0,752512	0,003
KNeighborsClassifier	0,759912	0,004
KNeighborsClassifier	0,758845	0,003
KNeighborsClassifier	0,756231	0,005
Media	0,755830	0,004
Naive Bayes	0,808618	0,003
Naive Bayes	0,808520	0,004
Naive Bayes	0,808635	0,005
Naive Bayes	0,808608	0,005
Naive Bayes	0,808635	0,002
Media	0,808603	0,004

GradientBoostingClassifier	0,849331	0,003
GradientBoostingClassifier	0,849330	0,003
GradientBoostingClassifier	0,849283	0,003
GradientBoostingClassifier	0,849187	0,003
GradientBoostingClassifier	0,849136	0,003
Media	0,849253	0,003
RandomForestClassifier	0,797856	0,004
RandomForestClassifier	0,797349	0,003
RandomForestClassifier	0,797074	0,003
RandomForestClassifier	0,797488	0,004
RandomForestClassifier	0,797091	0,005
Media	0,797372	0,004

Questo classificatore è stato scelto a causa delle alte prestazioni rispetto agli altri e non abbiamo preso affatto in considerazione i regressori a causa del contenuto del dataset, che contiene soli valori discreti.

Il file di cui è stato mostrato l'estratto è disponibile nel repository fornito, denominato "test_models.ods" del quale di seguito un breve confronto.



- Come strumento per misurare l'accuratezza del sistema, ci siamo avvalsi della K-Fold cross-validation. In particolare usavamo 10 fold e come metrica per l'accuratezza la "Compute Area Under the Receiver Operating Characteristic Curve"(roc_auc), abbiamo scelto questa a causa della sua compatibilità con features binarie.
  Abbiamo implementato anche una funzione che permette di visionare la precisione del sistema, nell'esecuzione corrente. Anche qui come metrica per le precisione abbiamo la roc_auc, che utilizza di default una macro-media.
- Cosi come per l'accuratezza, abbiamo pensato potesse tornare utile tenere conto dell'errore che il sistema presenta, quindi sempre mediante un'apposita funzione, è possibile osservare il "Mean Absolute Error", cioè l'errore assoluto riscontrato tra le features predette e quelle reali sia in fase di training che di test.
  Es: del Decision Tree Regressor;

```
L'accuratezza sulle predizioni fatte in fase di apprendimento e' di 0.9257173041381166
L'accuratezza sulle predizioni fatte in fase di test e' di 0.5891838980654001

Mean Absolute Error
MAE train 0.014538000630715862
MAE test 0.13757489750867236

Accuratezza media dei k-fold e': 0.601024 con deviazione di +/- (0.005)
```

- Ci siamo poi voluti cimentare con il prolog, effettuando una semplice KB, composta da asserzioni che specificano la manifestazione, o meno, di specifici sintomi comunicati dall'utente, cosi da consigliare di andare da uno specialista per una visita più approfondita, ciò a prescindere dalla predizione effettuata dal sistema. In particolare viene consigliata una visita se l'utente comunica di avere colesterolo alto, un

BMI superiore a 30, difficoltà a salire le scale, ictus, il chè potrebbe potenzialmente manifestare una complicazione del quadro clinico del paziente.

- Nel nostro applicativo è stata implementata una funzione che sfrutta una Rete Bayesiana. Questa è in grado di predire la percentuale di possibilità di assunzione dell'utente nell'azienda. L'assistente, infatti, ponendo delle domande all'utente, calcolerà un float compreso tra 0 e 1, che moltiplicato per 100 darà come risultato la percentuale di possibilità di assunzione. In base alla percentuale restituita:

```
if max[1] < 0.39:
    print("Godi di una buona salute mentale!\n"
          "Anche se potrebbe risultare non necessario, ti è consigliata una visita da un professionista, anche per prova.
elif max[1] < 0.59:
    print("Potresti soffrire di sintomi di lieve-media entità che potrebbero causare problemi alla tua salute psicofisica
          "Anche se non potrebbe risultare non necessario, ti è consigliata una visita da un professionista, anche per pr
elif max[1] < 0.79:
    print("Potresti soffrire di sintomi medio gravi legati alla tua salute mentale che potrebbero causarti malattie come:
          "E' consigliata una visita da uno professionista. \n")
else:
    print("Potresti soffrire di sintomi gravi legati alla tua salute mentale che potrebbero causarti malattie come: solit
          "E' consigliata vivamente una visita da uno professionista. \n")
```

Ogni domanda è pesata differentemente e riguardano:

- *Sintomi medi*:
  - **Panico**: se l 'utente ha mai avuto attacchi di panico:
  - **Prendersela con se stessi**: se l'utente tende ad incolparsi;

- *Sintomi gravi*:
  - **Mancanza di speranza**: se l'utente crede di non poter far nulla per cambiare questo scenario cupo che prevede solamente la presenza di eventi negativi;
  - **Tendenze suicide**: se l'utente ha mai avuto pensieri riguardanti il porre fine alla sua vita;

ESEMPI RETE BAYESIANA:

P(panic = si) = 0.60
P(panic = no) =0.40
P(blamingYourself = si) = 0.55
P(blamingYourself = no) = 0.45
P(mediumSymptom = si | panic = si ^ blamingYourself = si) = 0.85
P(mediumSymptom = no | panic = si ^ blamingYourself = si) = 0.15
P(mediumSymptom = si | panic = si ^ blamingYourself = no) = 0.8
P(mediumSymptom = no | panic = si ^ blamingYourself = no) = 0.2
P(mediumSymptom = si | panic = no ^ blamingYourself = si) = 0.4
P(mediumSymptom = no | panic = no ^ blamingYourself = si) = 0.6
P(mediumSymptom = si | panic = no ^ blamingYourself = no) = 0.25
P(mediumSymptom = no | panic = no ^ blamingYourself = no) = 0.75

P(hopelessness = si) = 0.70
P(hopelessness = no) = 0.30
P(suicidalThought = si) = 0.80
P(suicidalThought = no) = 0.20
P(hardSymptom = si | hopelessness = si ^ suicidalThought = si) = 0.90
P(hardSymptom = no | hopelessness = si ^ suicidalThought = si) = 0.10
P(hardSymptom = si | hopelessness = si ^ suicidalThought = no) = 0.60
P(hardSymptom = no | hopelessness = si ^ suicidalThought = no) = 0.40
P(hardSymptom = si | hopelessness = no ^ suicidalThought = si) = 0.85
P(hardSymptom = no | hopelessness = no ^ suicidalThought = si) = 0.15
P(hardSymptom = si | hopelessness = no ^ suicidalThought = no) = 0.20
P(hardSymptom = no | hopelessness = no ^ suicidalThought = no) = 0.80

P(prediction = si | mediumSymptom = si ^ hardSymptom = si) = 0.99
P(prediction = no | mediumSymptom = si ^ hardSymptom = si) = 0.01
P(prediction = si | mediumSymptom = si ^ hardSymptom = no) = 0.55
P(prediction = no | mediumSymptom = si ^ hardSymptom = no) = 0.45
P(prediction = si | mediumSymptom = no ^ hardSymptom = si) = 0.90
P(prediction = no | mediumSymptom = no ^ hardSymptom = si) = 0.10
P(prediction = si | mediumSymptom = no ^ hardSymptom = no) = 0.05
P(prediction = no | mediumSymptom = no ^ hardSymptom = no) = 0.95

Architettura del sistema

Il sistema presenta la seguente struttura:

- **Progetto**
  - ***dataset***, contenente i dataset utilizzati. In particolare, *heart_disease_health_indicators_BRFSS2015* è quello originale, cioè così come reperito da Internet, che abbiamo pensato fosse bene conservare, e il secondo H*eartDiseaseHealthIndicators* che rappresenta il dataset effettivamente utilizzato in fase di apprendimento;
  - ***src***, contenente tutto il codice sorgente.
    - *model_performance.py*, contiene le funzioni utili a osservare le prestazioni del sistema (accuratezza, K-Fold e l'errore assoluto). Per quanto riguarda queste funzioni, abbiamo deciso di implementarle, non renderle utilizzabili all'utente ma conservarle per applicazioni/modifiche future, in cui possano tornare utili.
    - *system.py*, costituisce il cuore del progetto, si occupa di richiamare le librerie necessarie, aprire il dataset, creare un modello, addestrare il sistema e in seguito al richiamo dei metodi necessari all'acquisizione dei sintomi dell'utente, effettuare la predizione vera e propria.
    - *request.py*, contiene la parte di interazione con l'utente in cui vengono poste una serie di domande;
    - *user.py*, una classe con metodi e attributi necessari ad asserire i sintomi comunicati dall'utente;

- *prolog.py*, contiene la base di conoscenza che include i sintomi più gravi che possono suggerire una visita da uno specialista;
- *bayes.py*, contiene una funzione la quale sfrutta una rete bayesiana.

Processo e sviluppo dell'organizzazione del lavoro

Il nostro progetto è stato svolto sull'IDE PyCharm, abbiamo collaborato attraverso la piattaforma Microsoft Teams nel caso di problemi relativi a scelte implementative o codice e usato github come piattaforma di code hosting.

Conclusioni

Abbiamo riscontrato delle difficoltà nell'interfacciarci con questo nuovo linguaggio mai utilizzato. Nonostante questo, siamo soddisfatti del lavoro svolto che ci ha permesso di applicare le conoscenze ottenute durante questo corso e di utilizzare la nostra curiosità per superare gli ostacoli che si sono presentati in fase di progettazione.