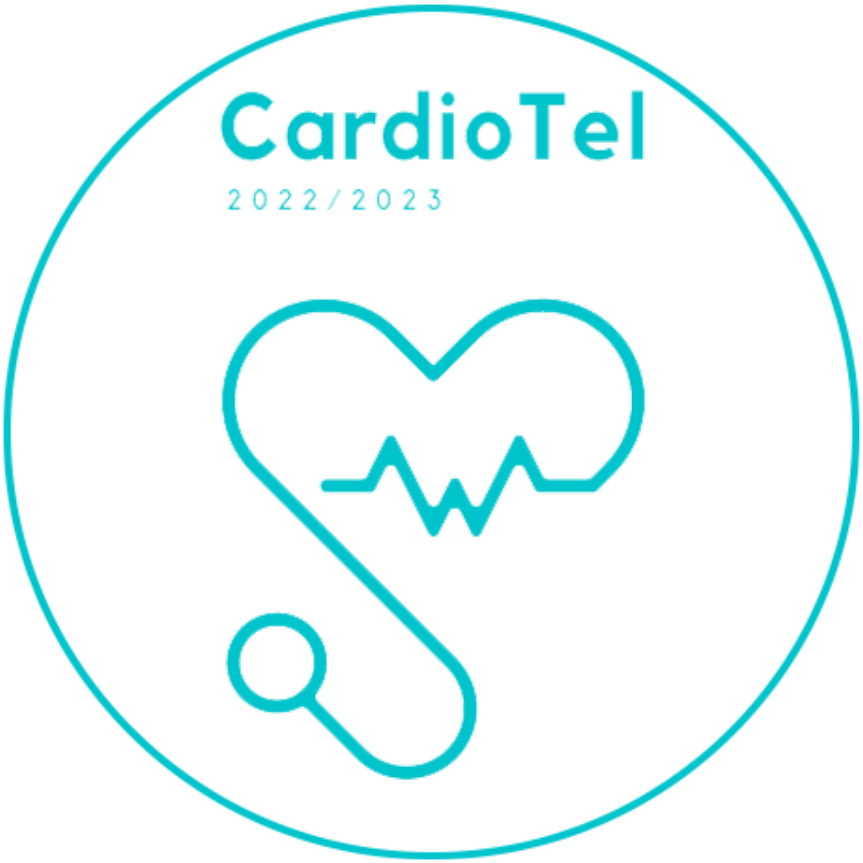
Università degli Studi di Salerno
Corso di Laurea Triennale in
Informatica



CardioTel

Repository GitHub: `https://github.com/MGiella/FIA_Project`

Alessio Romaniello Carmine Pascale
Mattia Giuseppe Giella Sabrina Pannullo

Corso di Fondamenti di Intelligenza Artificiale
Prof. Fabio Palomba
Anno Accademico 2022/2023

Indice

Capitolo 1

Introduzione

1.1 Cos'è CardioTel?

CardioTel è un progetto software nato dalla collaborazione di studenti della Laurea Triennale e Magistrale per l'esame di Ingegneria del Software e Gestione dei Processi Software.
Come suggerisce il nome (parola macedonia composta dai termini "cardiovascolare" e "telemonitoraggio"), lo scopo del progetto è creare un'applicazione (disponibile sul proprio browser che rende possibile il telemonitoraggio dei parametri vitali legati al sistema cardiovascolare, e delle malattie annesse. Per vigilare su quest'ultime, si è introdotto un modulo di Machine Learning che calcola la probabilità, per un utente, di soffrire di una malattia.
Segue le indicazioni della Nazioni Unite, soddisfando uno degli obiettivi dell'agenda 2030, la transizione al digitale.

Capitolo 2

Definizione del problema

2.1 Obiettivi

L'obiettivo del modulo di Intelligenza Artificiale di CardioTel è determinare le probabilità (espressa in percentuale) di rischio di infarto e di aterosclerosi.
Per infarto si intende l'arresto del miocardio; le variabili che suggeriscono un possibile infarto sono:

- frequenza cardiaca;
- colesterolo;
- pressione arteriosa.

Per aterosclerosi si intende l'indurimento e la perdita di elasticità delle arterie a causa dell'accumulo di colesterolo, cellule infiammatorie e altro. I valori che influiscono sulla suddetta condizione sono:

- colesterolo;
- pressione arteriosa.

2.2 Specifiche P.E.A.S.

- Performance (misura di prestazione di un agente intelligente): bontà delle predizioni (attendibilità, velocità di aggiornamento);
- Environment (descrizione dell'ambiente e degli oggetti che lo compongono): paziente, misurazioni effettuate;
- Actuators (attuatori dell'agente): monitor sul quale mostrare le percentuali calcolate;
- Sensors (sensori dell'agente): sensori IoT, comunicano con il sistema restante tramite protocollo MQTT.

2.2.1 Caratteristiche dell'ambiente

- Parzialmente osservabile: è possibile rilevare soltanto le misurazioni fino all'istante corrente;
- Stocastico: lo stato successivo non dipende soltanto dell'azione dell'agente;
- Sequenziale: l'esperienza è data dall'insieme degli episodi;
- Dinamico: la rilevazione delle misurazioni continua durante la delibera dell'agente;
- Discreto: l'insieme di rilevazioni è finito;
- Singolo: unico agente.

Capitolo 3

Analisi del problema

3.1 CRISP-DM

Come approccio ingegnereristico, sono state seguite le linee guida del CRISP-DM (Cross-Industry Standard Process for Data Mining), standard per la realizzazione di un progetto di Machine Learning.
È un modello a cascata con feedback e non sequenziale (le fasi possono essere effettuate più volte); risulta tradizionale ma allo stesso tempo flessibile.
Le fasi del CRISP-DM sono:

- **Business Understanding**: raccolta dei requisiti, definizione degli obiettivi da raggiungere e dei criteri di valutazione annessi, e ancora le tecnologie e i tool da utilizzare. Output della fase è il piano di progetto, che spiega il progetto dal punto di visto socio-tecnico;
- **Data Understanding**: identificazione, raccolta e analisi dei dataset necessari al raggiungimento degli obiettivi di business. Si documentano i dati, si identificano eventuali relazioni tra di essi e si analizza la loro qualità. Output della fase è il documento di analisi dei dati;
- **Data Preparation**: preparazione dei dati per il loro utilizzo nelle fasi successive. In questa fase si selezionano le caratteristiche con maggiore potenza predittiva (feature engineering), si utilizzano tecniche di pulizia dei dati per risolvere eventuali problemi di qualità rilevati in precedenza. Output della fase è l'insieme di dati formattati per il loro utilizzo nel modello di ML;

- **Modeling**: individuazione dell'algoritmo che meglio si presta al problema affrontato, e addestramento con i dataset della fase precedente. Output della fase è il modello di Machine Learning;
- **Evaluation**: validazione della correttezza del modello rispetto ai criteri di successo. Output di questa fase sono i risultati di validazione;
- **Deployment**: definizione della modalità grazie alla quale il modello sarà reso disponibile agli utenti. Output di questa ultima fase è il report del progetto, incluso il piano di manutenzione e di monitoraggio.

3.2 Business understanding

Il RAD del progetto complessivo comprende tutti gli aspetti interessanti di questa fase. È disponibile al link `https://github.com/MGiella/FIA_Project/blob/master/Documents/2022_RAD_C10.pdf`.

3.3 Raccolta e analisi

Data la non reperibilità di dataset interessanti (la difficoltà più grande è la mancanza di caratteristiche definenti il problema), si è proceduto all'approfondimento dei valori tipici e poi alla creazione ex-novo di un dataset. La generazione è pseudo-casuale: i valori sono originati in un range base di riferimento (cosidetti valori "normali") e, se questi risultano essere agli estremi del range, allora sono sostituiti da un valore oltre il valore regolare. I dati generati sono poi salvati in file ARFF.
Non è stato necessario effettuare un'analisi sui dati, data la riproduzione controllata.

3.4 Processing dati

Il dataset costruito ha come attributi:

- frquenza cardiaca;
- colesterolo;
- pressione arteriosa (max e min);
- ossigenazione;
- temperatura corporea.

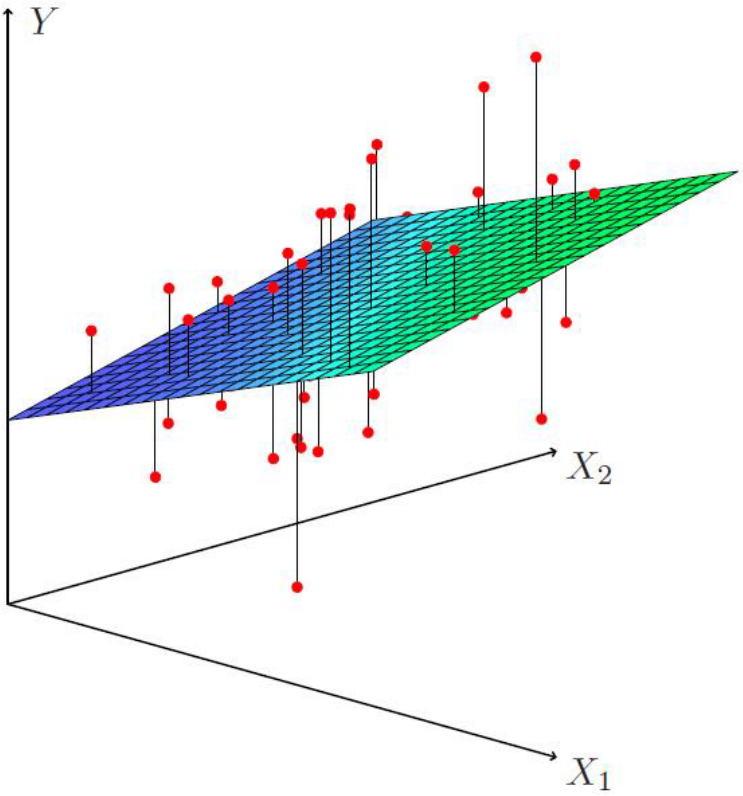
Le due malattie sono indicate da valori anomali di frequenza cardiaca, colesterolo e pressione arteriosa. Quindi sono stati eliminati gli attributi non necessari.
Successivamente sono stati creati due dataset: per la predizione dell'aterosclerosi gli attributi sono pressione arteriosa e colesterolo, per la predizione dell'infarto tutti gli attributi conservati.
Si è proceduto a normalizzare i valori delle caratteristiche (in scala 100).
In ultimo, tutti i dati grezzi si sono ritenuti importanti per la predizione delle malattie, e sono stati utilizzati come feature del modello. Grazie alla distribuzione ottenuta dalla generazione di rilevazioni, non si è considerato necessario un'ulteriore fase di data balancing.

Capitolo 4

Soluzione proposta

4.1 Modeling

In un primo momento, si è pensato di utilizzare un modello di regressione lineare multiplo per la risoluzione del problema.



La regressione lineare multipla è un particolare task supervisionato quale obiettivo é predire una variabile numerica (target) sulla base di dataset. In particolare, la regressione lineare determina la funzione

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

utilizzando un insieme di valori $X = \{X_1, X_2, ...X_n\}$ dove per ogni x_i il valore di $f(x_i)$ è noto.
Il calcolo del residuo tra funzione stimata e funzione effettiva si ottiene:

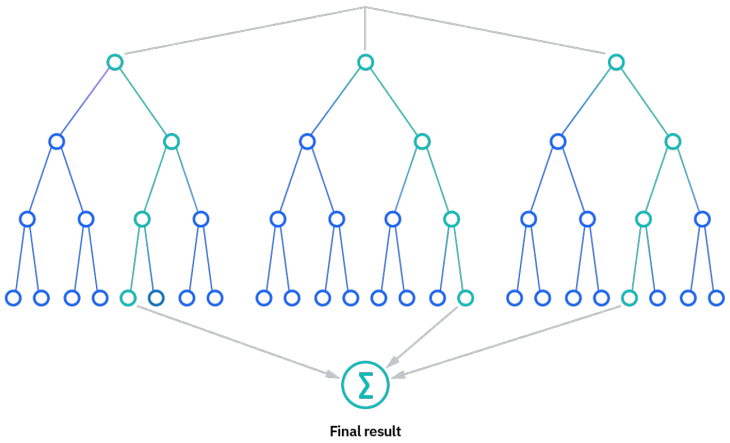
$$\epsilon = Y - f(x)$$

Il modello può essere impiegato solo se sono soddisfatte determinate condizioni:

- la relazione tra l'insieme $X = \{x_1, x_2, ...x_n\}$ e $Y = \{f(x_1), f(x_2), ...f(x_n)\}$ deve essere rappresentabile tramite una funzione lineare;
- gli errori devono essere normalmente distribuiti;

- gli errori devono avere una varianza costante;
- i residui devono essere indipendenti per ogni valore di X.

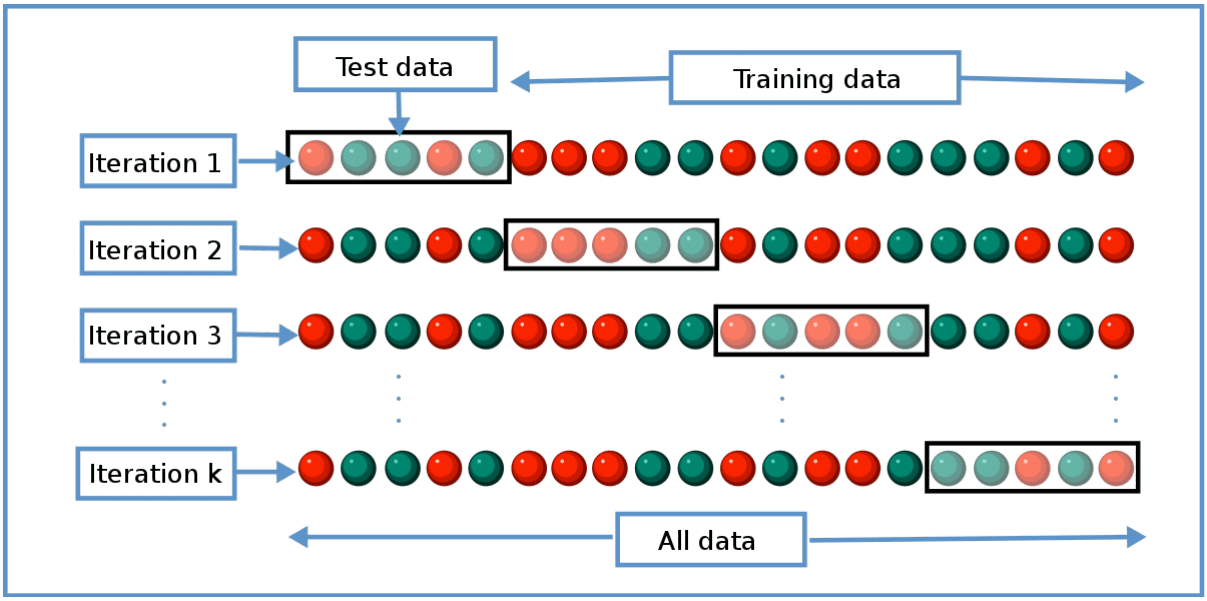
I dati a disposizione risultano essere non lineari. Si è proceduto quindi secondo un algoritmo di Random Forest.



Random Forest è un algoritmo di apprendimento supervisionato che utilizza un insieme di alberi decisionali, noto come "foresta", per prendere decisioni predittive.
Il Random Forest si basa sull'idea di creare diversi alberi decisionali, ognuno addestrato su un sottoinsieme casuale dei dati di addestramento. Durante la fase di previsione, ogni albero nella foresta fornisce una previsione e la classe o il valore di output più comune tra gli alberi viene selezionato come risultato finale.
La principale intuizione alla base del Random Forest è che la combinazione di più alberi decisionali può migliorare le capacità predittive rispetto a un singolo albero. Questo è possibile perché gli alberi all'interno del Random Forest sono diversi tra loro, grazie all'utilizzo di sottoinsiemi diversi dei dati di addestramento e delle caratteristiche.
L'utilizzo di un insieme di alberi decisionali nel Random Forest contribuisce a ridurre l'overfitting e a fornire previsioni più accurate e robuste.
Il singolo decision tree è costruito selezionando casualmente caratteristiche del problema, n decision tree sono combinati insieme selezionando il risultato più comune oppure selezionando il risultato medio.

4.2 Validazione del modello

La tecnica utilizzata per la validazione del modello è la 10-fold Cross-Validation. Questa tecnica consiste nel dividere i dati di input in fold, addestrare il modello su una parte di essi e testarlo sull'altra parte. Questo processo viene ripetuto più volte, utilizzando una diversa combinazione di fold come set di addestramento e di test ogni volta, e i risultati vengono quindi combinati per ottenere una stima più precisa della performance del modello.



4.3 Valutazione del modello

La libreria weka Java API contiene contiene la classe Evaluation che, oltre alla validazione, permette di calcolare varie metriche di valutazione del modello.

- Mean Absolute Error: differenza media (in valore assoluto) osservata tra i valori tra i valori previsti e i valori osservati

  $$\frac{\sum_{i=1}^{n} |\tilde{y} - y|}{n}$$

  il valore è circa 4.5;

- Root Mean Absolute Error: radice quadrata del mean squared error (MSE), che è la media dei quadrati degli errori tra i valori previsti e i valori osservati. La radice quadrata viene applicata per riportare l'errore alla stessa scala dei valori originali.

  $$\sqrt{\frac{\sum_{i=1}^{n} (\tilde{y} - y)^2}{n}}$$

  Il valore è circa 5.7.

- Correlation Coefficient:

  $$\rho = \frac{\sum((X_i - \overline{X})(Y_i - \overline{Y}))}{\sqrt{\sum((X_i - \overline{X})^2) \cdot \sum((Y_i - \overline{Y})^2)}}$$

Il valore è circa 0,2.

4.4 Implementazione

Per l'implementazione é stato fatto uso di "Weka", software di aprrendimento disponibile come libreria per Java.

- Viene generato il file ARFF, insieme delle misurazioni alle quali è associata una predizione già calcolata; questo file sarà il dataset di training.

```java
public static String getArff(String malattia, List<Device> rilevazione) {
    Instances dataset = null;
    String outputFilename = "";

    if (malattia.equals("infarto")) {
        dataset = PredizioniInfartoService.getAsInstanceInfarto(rilevazione, testing: null);
        outputFilename = "RilevazioniSetInfarto.arff";
    } else {
        dataset = PredizioniAteroService.getAsInstanceAtero(rilevazione, testing: null);
        outputFilename = "RilevazioniSetAtero.arff";
    }

    try {
        BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilename));
        writer.write(dataset.toString());
        writer.flush();
        writer.close();
    } catch (Exception e) {
        System.err.println("Failed to save data to: " + outputFilename);
        e.printStackTrace();
    }

    return outputFilename;

}
```

```java
public static Instances getAsInstanceAtero(List<Device> rilevazioni, String testing){
    @Deprecated
    FastVector deviceList = new FastVector<>();

    @Deprecated
    FastVector instance = new FastVector<>();
    //creare fastvector con gli attributi necessari alla predizione

    instance.addElement(new Attribute( attributeName: "pressione"));
    instance.addElement(new Attribute( attributeName: "pressione_due"));
    instance.addElement(new Attribute( attributeName: "colesterolo"));
    instance.addElement(new Attribute( attributeName: "predizione"));
    Instances data = new Instances( name: "dataSetAtero", instance, capacity: 0);

    for(int i=0;i<rilevazioni.size();i++){
        double[] vals = new double[instance.size()];  // important: needs NEW array!

        Device rilevazione = rilevazioni.get(i);
        vals[0] = MLModel.normalise(rilevazione.getPressione(), Device.minPresMas, Device.maxPresMas);
        vals[1] = MLModel.normalise(rilevazione.getPressione_due(), Device.minPresMin, Device.maxPresMin);
        vals[2] = MLModel.normalise(rilevazione.getColesterolo(), Device.minCol, Device.maxCol);
        if (testing == null)
            vals[3] = MLModel.calcolaPrediction(rilevazione, malattia: "atero");
        data.add(new DenseInstance( weight: 1.0,vals));
    }
    return data;
}
```

- Viene generato un modello di Random Forest tramite dataset di training letto dal file ARFF.

```java
public static RandomForest getModel(DataSource source) throws Exception {

    //get dataset from datasource
    Instances dataset = source.getDataSet();
    dataset.setClassIndex(dataset.attribute( name: "predizione").index());

    //Build model
    RandomForest model = new RandomForest();
    model.buildClassifier(dataset);
    return model;

}
```

- Viene effettuata una 10-fold Cross-Validation per testare il modello.

```java
public static Predizione classifyInstance(Instances instances, DataSource source) {

    try {
        //ottenimento modello dal dataset
        RandomForest model = getModel(source);
        Instances Trainingdataset = source.getDataSet();
        Trainingdataset.setClassIndex(Trainingdataset.attribute( name: "predizione").index());
        Evaluation eval= new Evaluation(Trainingdataset);

        instances.setClassIndex(instances.attribute( name: "predizione").index());
        eval.crossValidateModel(model,Trainingdataset, numFolds: 10,new Random());

        Predizione pr = new Predizione();
```

- Dato un dataset di rilevazioni, viene valutata l'istanza e restituisce in output una media delle predizioni.

```java
    Predizione pr = new Predizione();
    //classificazione instance, il metodo fa una media delle predizioni
    double percent = 0;
    int cont = 0;
    for (Instance i : instances) {
        percent += model.classifyInstance(i);
        cont++;
    }
    percent = percent / cont;
    pr.setPercentualeRischio(percent);
```

- L'agente mostra a schermo le predizioni.

Capitolo 5

Conclusioni

5.1 Considerazioni finali

Il progetto vuole essere una semplice soluzione al problema di indidualizzazione di possibili malattie cardio-vascolari.
É possibile sostituire il modello utilizzato con uno di classificazione, ma l'intento principale del progetto è di determinare la probabilità in percentuale del possibile rischio di infarto/ateriosclerosi.
Seppure ci sono stati apporfondimenti sulle due malattie, il modello potrebbe non rispettare i criteri della Medicina.
La lezione appresa è la seguente: la formulazione del problema procede qualsiasi altra azione, e con l'esperienza, costituisce la chiave del successo.