Villanueva - Comini

**ESAME R**

**Analisi descrittiva**

Per avere una prima visione delle variabili raccolte, abbiamo caricato il dataset “heart.csv” nella nuova variabile dataset con la funzione ‘read.csv’ e abbiamo, di conseguenza, analizzato la struttura dei dati raccolti. [SCREEN DATASET img1]

Osservando il dataset ci siamo accorti della presenza di valori NA dovuti, ad esempio, al data-entry manuale; quindi abbiamo creato un ciclo che controllasse la presenza di tali valori. Tutti i valori NA sono stati cancellati definitivamente dal dataset con la funzione ‘dataset <- na.omit (dataset)’.

Abbiamo proseguito l’analisi del dataset controllando e rimuovendo le colonne non ritenute necessarie. Infatti, attraverso la funzione ‘view(dataset)’, l’abbiamo visualizzato e, attraverso la funzione ‘subset ()’, abbiamo rimosso le colonne superflue; in questo specifico caso la colonna X perché ritenuta inutile.

Terminato quest’ultimo passaggio, abbiamo rinominato le colonne in maniera appropriata, descrivendone, di ciascuna, il tipo di attributo.

Abbiamo stampato il dataset con la funzione ‘str(dataset)’ e, con la funzione ‘names(dataset)[names(dataset) == "vecchio"] <- "nuovo"’, sono state rinominate le colonne in modo appropriato. Come ultima azione abbiamo assegnato ad ogni attributo il suo tipo.

names(dataset)[names(dataset) == "cp"] <- "chest\_pain"

names(dataset)[names(dataset) == "trestbps"] <- "rest\_bp"

names(dataset)[names(dataset) == "chol"] <- "cholesterol"

names(dataset)[names(dataset) == "thalach"] <- "max\_hr"

names(dataset)[names(dataset) == "exang"] <- "exercise\_angina"

names(dataset)[names(dataset) == "thal"] <- "thalassemia"

names(dataset)[names(dataset) == "target"] <- "heart\_disease"

names(dataset)[names(dataset) == "ca"] <- "n\_vessels"

names(dataset)[names(dataset) == "restecg"] <- "rest\_ecg"

age int ORDINALE

sex chr NOMINALE

chest\_pain int NOMINALE

rest\_bp int DI RAPPORTO

cholesterol chr DI INTERVALLO

fbs int DI RAPPORTO

rest\_ecg int NOMINALE

max\_hr int DI INTERVALLO

exercise\_angina int NOMINALE

oldpeak num ORDINALE

slope int NOMINALE

n\_vessels int ORDINALE

thalassemia int NOMINALE

heart\_disease int NOMINALE

Grazie alla libreria tidyverse() abbiamo potuto eliminare la colonna x e rinominare le colonne con i nuovi nomi.

dataset <- dataset %>%

select(-one\_of("x")) %>%

rename(

chest\_pain = cp,

rest\_bp = trestbps,

cholesterol = chol,

max\_hr= thalach,

exercise\_angina = exang,

thalessemia = thal,

heart\_disease = target,

n\_vessels = ca,

rest\_ecg = restecg

)

Successivamente, abbiamo eseguito un controllo per correggere la consistenza del tipo di dato per ogni variabile con la funzione ‘ [colonna del dataset] <- as. [tipo nel quale voglio cambiare i dati] (colonna del dataset)’ così come segue:

* Abbiamo trasformato, nella colonna sex, i semplici valori "0" e "1” in "F" per femmina e in "M" per maschio e poi abbiamo cambiato il tipo di dato da int a factor (quindi diviso in più livelli) per gli attributi “F” e “M”.
* Abbiamo cambiato il tipo di dato per la colonna chest pain da int a factor quindi diviso in più livelli (0 - 1 - 2 - 3).
* Per la colonna cholesterol abbiamo trasformato in primo luogo tutti i valori "undefined" nella mediana dei valori di tutta la mia colonna; in secondo luogo abbiamo trasformato il tipo di dato da char a integer.
* Abbiamo cambiato il tipo di dato per la colonna fbs da int a factor quindi diviso in più livelli (1 - 0).
* Abbiamo cambiato il tipo di dato per la colonna rest\_ecg da int a factor quindi diviso in più livelli (0 - 1 - 2).
* Abbiamo cambiato il tipo di dato per la colonna exercise\_angina da int a factor quindi diviso in più livelli (1 - 0).
* Abbiamo cambiato il tipo di dato per la colonna slope da num a factor quindi diviso in più livelli (0 - 1 - 2).
* Abbiamo cambiato il tipo di dato per la colonna thalassemia da int a factor quindi diviso in più livelli (0 - 1 - 2 - 3).
* Abbiamo cambiato il tipo di dato per la colonna heart\_disease da int a factor quindi diviso in più livelli (1 - 0).

Grazie alla libreria tidyverse() abbiamo potuto snellire il codice cambiando il tipo di dato delle variabili, togliendo eventuali valori undefined e rinominando i livelli per sex.

dataset <- dataset %>%

mutate(

age <- as.integer(age),

sex = ifelse(sex == "1", "M", "F"),

sex = as.factor(sex),

chest\_pain = as.factor(chest\_pain),

cholesterol =ifelse(cholesterol == "undefined", median(cholesterol), cholesterol),

cholesterol = as.integer(cholesterol),

fbs = as.factor(fbs),

rest\_ecg = as.factor(rest\_ecg),

exercise\_angina = as.factor(exercise\_angina),

slope = as.factor(slope),

thalassemia = as.factor(thalassemia),

thalessemia = as.factor(thalessemia),

heart\_disease = as.factor(heart\_disease)

)

Per vedere se le modifiche fossero avvenute con successo abbiamo stampato nuovamente il dataset, abbiamo rinominato i livelli dei fattori per ogni colonna, così da renderli più comprensibili.

Livelli per chest pain

0 = "asymptomatic"

1 = "nontypical\_angina"

2 = "nonanginal\_pain"

3 = "typical\_angina"

Livelli per fbs

0 = "False"

1 = "True"

Livelli per rest\_ecg

0 = "Ventricular\_hypertrophy"

1 = "Normal"

2 = "Anomaly"

Livelli per exercise\_angina

0 = "No"

1 = "Yes"

Livelli per slope

0 = "Descending"

1 = "Flat"

2 = "Ascending"

Livelli per thalessimia

0 = "non\_existent"

1 = "defect\_corrected"

2 = "normal\_blood"

3 = "reversible\_defect"

Livelli per heart disease

0 = "Yes"

1 = "No"

Anche in questo caso, con la libreria tidyverse, abbiamo potuto ridurre la lunghezza del codice da scrivere.

Per far emergere gli outlier e le anomalie, come prima cosa abbiamo estratto con la funzione ‘hist(dataset$max\_hr)’ un grafico a barre dei valori dei dati forniti in ingresso dal dataset. In ggplo la funzione diventa ‘hist\_ggplot(dataset, dataset$max\_hr)’. [GRAFICO HIST img 2]

Abbiamo stabilito che il numero maggiore di battiti cardiaci non sia superiore a 222 e che il numero minore di battiti cardiaci sia il valore medio della variabile. [GRAFICO CAMBIATO HIST img3 ]

[Hist prima img4]

[box plot prima 5]

Oltre ad un istogramma, abbiamo utilizzato un altro grafico più esplicativo: il boxplot. Abbiamo rappresentato gli outliers relativi alla variabile rest\_bp’, abbiamo calcolato il 3° e il 1° quantile e li abbiamo sostituiti nella formula IQR<-(Q3-Q1) per trovare lo scarto interquartile ed infine abbiamo rilevato il range della differenza interquartile per creare il nuovo boxplot.

[HIST DOPO img6]

[BOXPLOT DOPO img7]

Ristampando a console il dataset, aggiornato e modificato, abbiamo notato che è più consistente rispetto alla prima volta che l’avevamo visualizzato; tuttavia ci sono ancora delle modifiche da apportare. Abbiamo quindi impostato sull’attributo age un controllo che non permettesse di inserire valori inferiori a 0 o superiori di 120; mentre sull’attributo rest\_bp, abbiamo inserito un range che va da 70 a 150 per la pressione sanguigna a riposo. Al termine di queste modifiche abbiamo ristampato il dataset e abbiamo stabilito che non ci fossero più modifiche da apportare poiché, per noi, ritenuto consistente. [SCREN DATASET img8]

Procedendo con l’analisi del dataset ci siamo confrontati con la regressione lineare semplice e quindi abbiamo messo in relazione due variabili per vedere, tramite gli appositi grafici di regressione lineare, se ci fosse o meno una correlazione.

Per prima cosa abbiamo scelto i due attributi da mettere in relazione, nel mio caso age e rest\_pb, e abbiamo stampato a video con la funzione “summary()” il loro contenuti suddivisi in quantili. Abbiamo continuato disegnando la retta di regressione e abbiamo notato, dal grafico, che tra age e rest\_bp non c’era correlazione. [GRAFICO DI REGRESSIONE img 9]

Per poter continuare con il metodo della regressione abbiamo dovuto, perciò, cambiare attributi. Abbiamo scelto di mettere a confronto gli attributi age e max\_hr; infatti, arrivati allo stesso punto di prima, abbiamo visto che c’era una forte correlazione tra i due. Infine, abbiamo potuto rappresentare il tutto tramite una rappresentazione grafica. [GRAFICO NUOVA REGRESSIONE img 10]

Continuando la procedura, abbiamo calcolato il coefficiente di correlazione lineare e il coefficiente di determinazione per i due attributi. Per ultimare la fase di regressione lineare abbiamo analizzato i residui con apposito grafico e, con un altro grafico, abbiamo potuto confrontare la distribuzione in quantili rispetto ad una distribuzione normale standard.

[GRAFICO DEI RESIDUI CON RETTA img 11]

[Distribuzione in quantili confrontabile con quella di una normale img 12]

Come ultima consegna ci è stato chiesto di creare un data frame contente 10 osservazioni, non presenti nel dataset, effettuando delle previsioni. [SCREN NEW DATASET img 13]

Tramite python abbiamo creato uno script che ci permettesse di creare 10 osservazioni casuali per il nostro nuovo file “osservazioni.csv” e, grazie alla funzione ‘predict ()’, abbiamo potuto predire l’intervallo di confidenza per il mio attributo. [SCREN SCRIPT PYTHON img 14]

Infine, abbiamo applicato due modelli di machine learning (k-Nearest Neighbors, Multi-Layer Perceptron) per misurare l’accuratezza sul test set e, di conseguenza, è stato creato un dot plot () del risultato modelli utilizzati. [GRAFICO DEI tre MODELLI img 15]

Il modello K-Nearest Neighbors (KNN) è uno degli altri algoritmi diffusi nel machine learning. Può essere utilizzato sia per problemi di classificazione che di regressione, anche se è più utilizzato nei primi. La forza di quest’algoritmo è che permette di memorizzare tutte le istanze disponibili e di classificarle valutando la distanza rispetto ai suoi vicini. L’istanza verrà assegnata alla classe che include il data point più vicino all’istanza stessa.

Il modello perceptron multistrato (MLP) è una rete neurale artificiale feedforward che genera un insieme di output da un insieme di input. Un MLP è caratterizzato da diversi livelli di nodi di input collegati come un grafico diretto tra i livelli di input e output. MLP utilizza la backpropogation per addestrare la rete. MLP è un metodo di apprendimento profondo.

Il modello Random Forest (RF)è un metodo versatile di machine learning, capace di affrontare sia compiti di classificazione che di regressione. Con le foreste casuali è anche possibile applicare metodi per la riduzione della dimensionalità, gestire dati mancanti, valori degli outlier ed altri passaggi essenziali di esplorazione dei dati, producendo buoni risultati.

Per facilitare la stesura del codice abbiamo utilizzato le librerie:

library(tidyverse)

library(caret)

library(ggthemes)

Per avere una condivisione più efficiente di tutti i file è stato usato GitHub.

Link di GitHub: <https://github.com/r-vil/heart_in_r>