



UNIVERSITÀ DI PAVIA
Facoltà di Ingegneria
Dipartimento di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Bioingegneria.

Docente:
Prof. **Riccardo Bellazzi**

Relazione a cura di:
Vittoria Bianchi
Mat. 529352

« La Saggezza della Folla »

ANALISI DEL DATASET:
“DIABETES”- Pima Indian Diabetes dataset

Corso di Apprendimento Computazionale in Biomedicina
A.A. 2022/2023

Indice

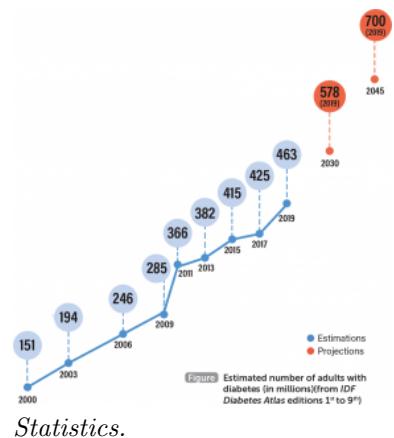
1 Definizione del problema e degli obiettivi dell'analisi dei dati	2
1.1 Principali Tipologie di Diabete (DM)	2
1.2 Work Flow	3
2 Comprensione e preparazione dei dati	5
2.1 Dataset	5
2.2 Pre-Processing	8
2.2.1 Gestione degli <i>Outliers</i>	8
2.2.2 Feature Elimination	9
2.2.3 <i>Imputing</i> dei dati Mancanti/Nulli	10
2.2.4 Normalizzazione	10
2.2.5 Divisione Trainig Set - Test Set	11
2.2.6 Bilanciamento delle Classi, SMOTE & <i>Data Augmentation</i>	12
2.2.7 Feature Selection	14
2.2.8 Discretizzazione	16
3 Modellizzazione e valutazione	17
3.1 I diversi Metodi considerati: scelta degli iperparametri	18
3.1.1 Naïve Bayes - (NB)	18
3.1.2 Support Vector Machines - (SVM)	18
3.1.3 Decision Tree - (DT)	19
3.1.4 Random Forest - (RF)	20
3.1.5 Logistic Regression - (LR)	21
3.1.6 k-Nearest-Neighbour - (KNN)	21
3.2 Tecniche di Valutazione	22
4 Scelta e raffinamento del modello	24
4.1 10-Fold Cross Validation	24
4.1.1 Training set Originale	24
4.1.2 Training set SMOTE	25
4.1.3 Training set <i>Data Augmentation</i>	26
4.2 Scelta del Training set e del Classificatore migliori	27
4.3 Test & Score	28
4.3.1 Paragone tra tutti i Classificatori: una mia curiosità	29
5 Utilizzo e disseminazione del modello	31
5.1 <i>Pipeline</i> Complessiva - Orange	32
Bibliografia	33

Capitolo 1

Definizione del problema e degli obiettivi dell'analisi dei dati

Il **diabete mellito (DM)** [1] è un disordine metabolico cronico a progressione multifattoriale. È una malattia 'Aliment Triggered' scatenata da livelli eccessivi di zucchero. Secondo l'International Diabetes Federation, nel **2019** circa **463 milioni di adulti (20–79 anni)** convivevano con il diabete. Analizzando la crescente **morbilità** degli ultimi anni, si prevede che entro il **2045** questa cifra salirà a **700 milioni**; il che significa che più di un adulto su dieci in futuro sarà affetto da diabete. [2]

Il DM è considerato anche una malattia autoimmune, poiché non può essere identificato un unico motivo principale come causa di tale patologia. Potrebbero esserci **molte ragioni**, alcune delle quali sono l'età, la storia familiare, la produzione di insulina, l'indice di massa corporea, lo stress, la gravidanza, ecc. Fattori analizzati all'interno dello studio.



1.1 Principali Tipologie di Diabete (DM)

Diabete di tipo 1. Viene anche chiamato diabete giovanile. Dipende dall'insulina: il sistema immunitario danneggia le cellule adibite al rilascio dell'insulina, riducendo la produzione di insulina nel corpo. Si verifica soprattutto nell'adolescenza. Il trattamento mira alla regolazione del livello di zucchero nel sangue con una terapia insulinica, dieta ed esercizio fisico. Alcune delle complicazioni per il tipo 1 includono danni ai reni, flusso sanguigno che si indebolisce, problemi in gravidanza e alla pelle.

Diabete di tipo 2. In questo caso la produzione di insulina da parte del corpo umano è insufficiente o esso resiste alla produzione di insulina. È più lieve rispetto al tipo 1 e può essere trattato con terapia insulinica, farmaci, dieta ed esercizio fisico. Le complicazioni che può causare colpiscono reni, nervi e retina; inoltre può causare malattie cardiache e ictus. La diagnosi prevede test come il test A1C, il test FPG (glucosio plasmatico a digiuno), il test RPG (glucosio plasmatico casuale) e il test di tolleranza al glucosio orale (OGTT).

Diabete gestazionale. Viene diagnosticato nei primi giorni di gravidanza. Alcune delle complicazioni includono fluttuazioni della pressione sanguigna, difficoltà respiratorie, complessità del peso alla nascita, parto precoce e diabete in futuro. Per la diagnosi viene effettuato il test di tolleranza al glucosio. Il trattamento per il diabete comprende dieta, iniezioni per l'equilibrio dell'insulina, esercizio fisico e monitoraggio della glicemia. Si tratta di una tipologia per la quale le probabilità che la malattia si manifesti dopo la nascita sono ridotte o nulle nella maggior parte dei casi; tuttavia, se non curato, potrebbe insorgere anche in età adulta.

Predibete. Viene anche chiamata alterata tolleranza al glucosio. È un caso in cui [1] la glicemia è alta rispetto al diabete di tipo 2. I fattori di rischio sono analoghi a quelli del diabete di tipo 2, e anche in questo caso le complicazioni includono malattie cardiache e l'ictus.

1.2 Work Flow

Quello che viene affrontato è dunque un problema di **classificazione** che mira a diagnosticare la malattia ad uno stadio precoce per poter intervenire repentinamente sugli effetti e sulle cause di essa. In tale contesto il Machine Learning gioca un ruolo chiave per la scoperta e la predizione delle diverse patologie: la classificazione nell'apprendimento automatico è infatti una delle principali tecniche decisionali utilizzate per l'analisi dei dati.

Viene riportato il Work-Flow seguito durante l'intero processo 1.1.

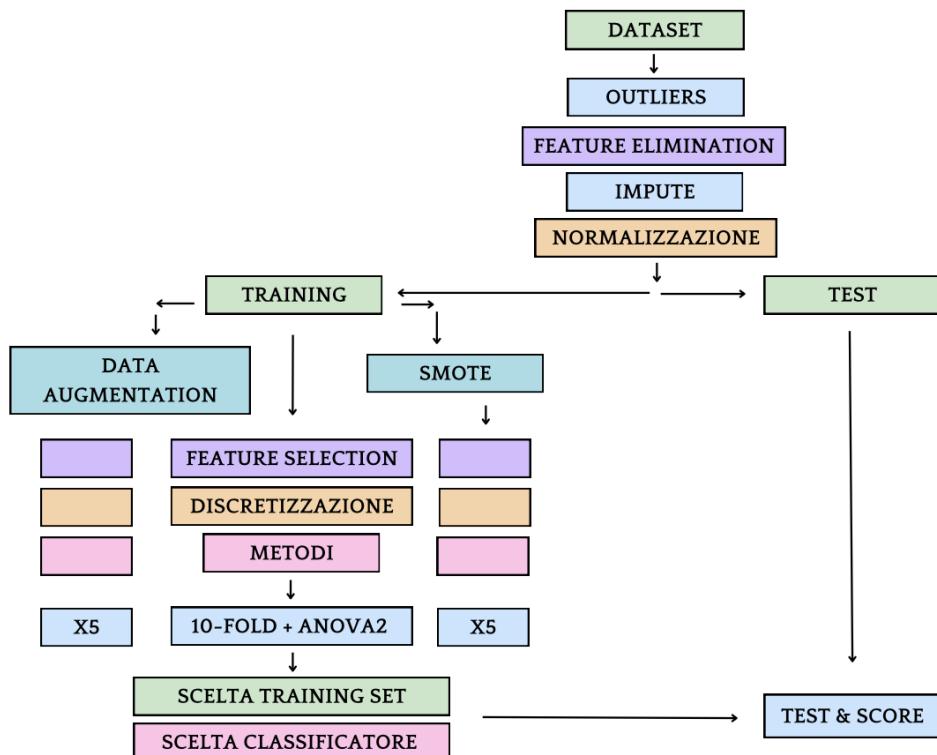


Figura 1.1: *Work Flow*.

Viene inizialmente eseguita un'analisi sull'intero **Dataset** che viene in seguito ripulito dagli **Outliers** e sul quale viene eseguita la **Feature Elimination** dell'attributo '*Insuline*'. Il nuovo *dataset*, ora ridotto, viene migliorato tramite **Imputing** dei dati mancanti, **Normalizzato** e suddiviso in **Training set** e **Test set**.

Si procede dunque parallelamente utilizzando il **Training set** definito **Originale** (che presenta classi sbilanciate), il **Training set ribilanciato** tramite l'algoritmo **SMOTE** e uno frutto della **Data Augmentation**. Su tutti e tre vengono provate tecniche di **Feature Selection** e attuando poi una **Discretizzazione** al fine di usare metodi di classificazione che la richiedono.

I **Classificatori** vengono inizialmente validati a livello di **Training set** tramite **10-Fold Cross Validation** analizzata in seguito tramite **ANOVA a 2 vie**. La validazione viene eseguita una sola volta per il set 'Originale'.

nale' e 5 volte per i set '*SMOTE*' e '*Data Augmentation*' per verificare la solidità dei risultati.

Viene quindi scelto su quale 'filone' di Training procedere (**Scelta Training set**) e quale modello (**Scelta Classificatore**) testare infine sul Test set tramite **Test & Score**.

Capitolo 2

Comprendere e preparazione dei dati

L'analisi esplorativa dei dati è il processo critico di esecuzione delle indagini iniziali sui dati per scoprire eventuali *patterns* e individuare anomalie. Una volta ottenuti i dati ed eseguita l'analisi preliminare, segue il *Pre-Processing*. I dati opportunamente processati entreranno nel *flow* della modellizzazione. Il risultato verrà valutato confrontando opportuni indicatori per la validazione del modello.

2.1 Dataset

I dati utilizzati constano di un *dataset* pubblico proveniente da *Kaggle* e avente il nome **Pima Indians Diabetes Database** (PIDD) [<https://www.kaggle.com/uciml/pima-indians-diabetes-database>]).

Il PIDD si riferisce ai risultati della diagnosi di diabete effettuata su pazienti di sesso femminile di almeno 21 anni. Sono presenti 9 attributi di cui 8 predittivi e un *Outcome Target* (0 or 1). Tali attributi sono: Pregnancy, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age and outcome, come riportato in Figura 2.1.

Number	Attribute Name	Description
1	Pregnancies	Number of time pregnant
2	Glucose	Plasma glucose concentration a 2h in an oral glucose test
3	Blood Pressure	Diastolic blood pressure (mmHg)
4	Skin Thickness	Triceps skin fold thickness (mm)
5	Insulin	2h Serum Insulin (mu U/ml)
6	BMI	Body Mass Index (weight kg / height m^2)
7	Diabetes Pedigree Function	Diabetes Pedigree Function
8	Age	Age (years)
9	Outcome	Class Variable (0 or 1)

Figura 2.1: *Dataset*.

In particolare con '**Skin Thickness**' si intende la piega cutanea del tricipite; data la relazione fra il grasso sottocutaneo e il grasso corporeo totale, si ritiene che il risultato della misura delle pliche sia un buon indicatore della densità corporea totale. La '**Diabetes Pedigree Function**' invece fornisce alcuni dati sulla storia del diabete mellito nei parenti e sulla relazione genetica di questi parenti con il paziente. Questa misura dell'influenza genetica dà una stima del rischio ereditario che si potrebbe avere con l'insorgenza del diabete mellito.

Le diverse Features vengono 'graficate' tramite **Istogrammi** per un'analisi iniziale. Ci si rende subito conto che sebbene il dataset PIDD non presenti colonne contenenti valori mancanti, tuttavia alcune delle misurazioni (Glucose, Blood Pressure, Skin Thickness, Insuline e BMI) mostrano **valori pari a 0**, il che non è possibile per un organismo umano vivente. Tale considerazione vale anche per il valore dell'Insulina Sierica ('Insuline'), che nei pazienti diabetici risulta essere bassa ma che non può assumere fisiologicamente il valore di 'zero tondo'.

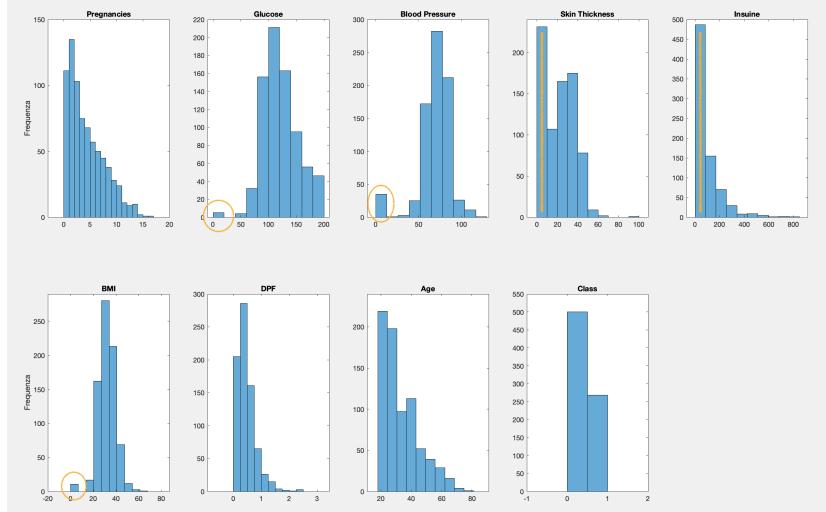
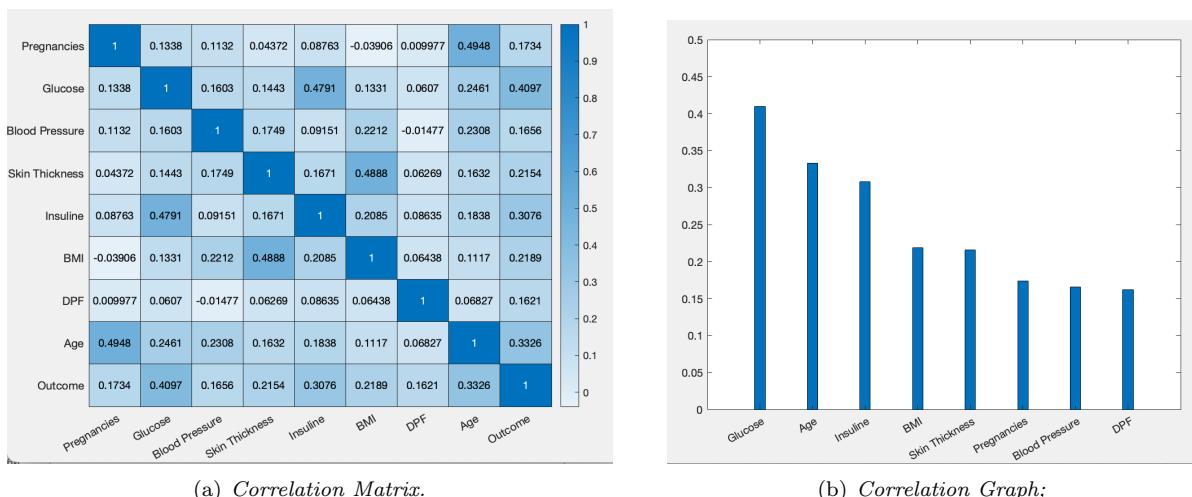


Figura 2.2: *Histogram Original Dataset*.

Pertanto, tali valori vengono considerati come mancanti e i calcoli successivi vengono momentaneamente eseguiti sostituendo tali valori nulli con dei '**NaN**', che non vengono automaticamente considerati all'interno della computazione (sia in ambiente Orange, sia in ambiente Matlab).

La correlazione dei diversi attributi viene presa in considerazione per scoprire quali di essi svolgono un ruolo importante nella previsione della malattia. Il **Coefficiente di Correlazione**, e quindi la **Matrice di Correlazione**, viene calcolato tramite la formula di 'Kendall' che si pensa essere la più opportuna per lo specifico *dataset*.



(a) *Correlation Matrix*.

(b) *Correlation Graph*;

Figura 2.3:

Il valore tra due attributi nella Matrice di Correlazione 2.3 (a), denota in che modo sono dipendenti l'uno dall'altro. Il **Grafico di Correlazione** indica invece la correlazione con la variabile di 'Outcome' ed è riportato

in Figura 2.3 (b), dove sull'asse y è indicato il valore del '*Feature Importance Score*'.

Le correlazioni osservate non sono da considerarsi elevate ma tuttavia rispecchiano il **significato fisico e fisiologico** delle variabili in esame. In particolare appare evidente come vi sia una correlazione tra Glucosio e Insulina: infatti l'ingresso del glucosio nelle cellule è dipendente dall'insulina stessa. L'insulina promuove l'entrata dello zucchero nelle cellule e lo toglie dal sangue (abbassa la glicemia), mentre il glucosio regola la secrezione pancreatica di insulina.

Il Glucosio influenza particolarmente anche la variabile di 'Outcome' e risulta essere la Feature più significativa all'interno del *dataset*, seguita dall'età della paziente e dalla concentrazione di Insulina Sierica. Tale risultato è confermato anche dall'analisi tramite ***Information Gain***, ***Information Gain Ratio***, ***Indice di Gini*** e analisi del χ^2 , analisi eseguita attraverso il Widget 'Rank' disponibile in Orange (Fig. 2.4).

#	Info.gain	Gain ratio	Gini	χ^2
1	0.170	0.085	0.101	139.901
2	0.081	0.041	0.048	62.029
3	0.079	0.039	0.044	53.744
4	0.055	0.030	0.031	8.780
5	0.043	0.021	0.028	34.316
6	0.036	0.018	0.022	5.262
7	0.022	0.011	0.014	16.143
8	0.015	0.007	0.009	12.918

Figura 2.4: *Output 'Rank' - Orange*.

Dall'immagine 2.5, che rappresenta gli **Istogrammi evidenziando la Classe** di appartenenza, già si può notare come 'Outcome' presenti differenze notevoli al variare della variabile 'Classe'. Discorso analogo può essere fatto per 'Insuline', mentre non appare così netto per 'BMI'.

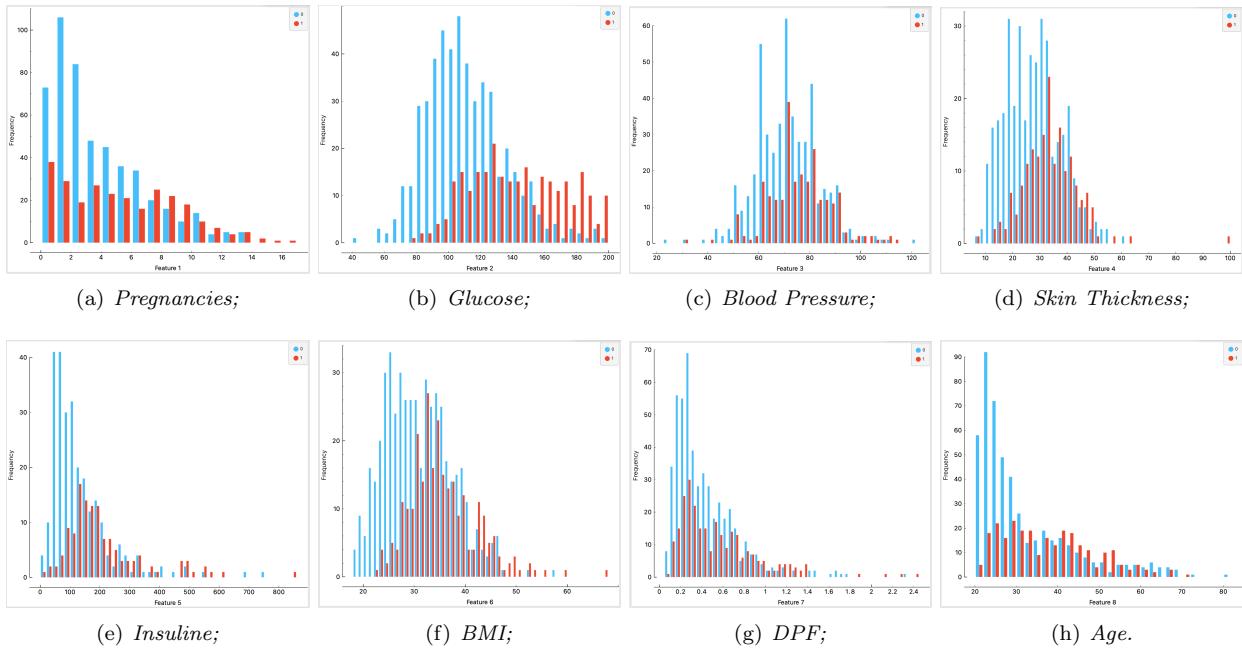


Figura 2.5: *Iistogrammi per Classe*

2.2 Pre-Processing

Il **Pre-Processing** è il processo iniziale che trasformerà i dati di input in dati con il formato appropriato e pronto per essere elaborato. Può anche essere utilizzato per modificare i dati in modo tale da poter considerare più di un tipo di algoritmo durante l'elaborazione e l'implementazione.

2.2.1 Gestione degli *Outliers*

A seguito del processo appena descritto, graficando i nuovi istogrammi, che quindi non considerano ora i valori nulli convertiti precedentemente in 'NaN', è possibile notare la presenza di alcuni *outliers*.

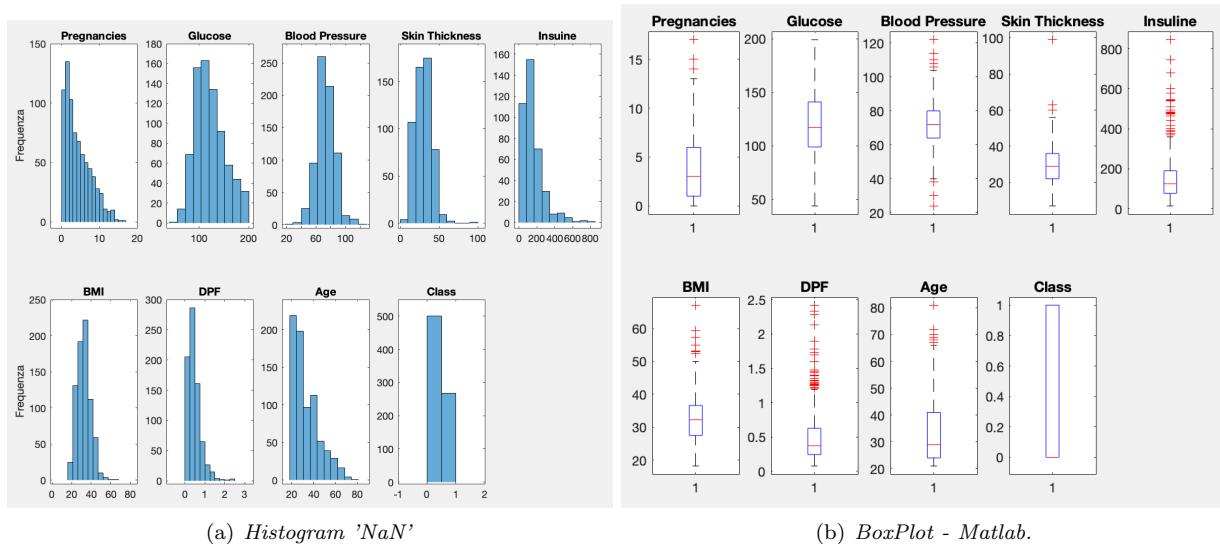


Figura 2.6:

Il 'boxplot' di Matlab ne segnala numerosi (indicati con il segno '+' in rosso all'interno del 'BoxPlot'): un numero troppo elevato, non compatibile con la rimozione di essi.

Il Widget 'Outliers' di Orange seleziona invece 6 dati fuori range, come riportato in Figura 2.7. Il Widget è stato impostato con 'Contamination' al 1% e 'Metric' 'Euclidian', utilizzando come 'Method' il 'Local Outlier Factor' che calcola un *Fattore di Outlier Locale* per ogni punto del dataset; tale fattore rappresenta la deviazione della densità locale del punto rispetto alla densità locale dei suoi vicini (una sorta di KNN). Questo fattore può essere utilizzato per identificare appunto i diversi *outliers*.

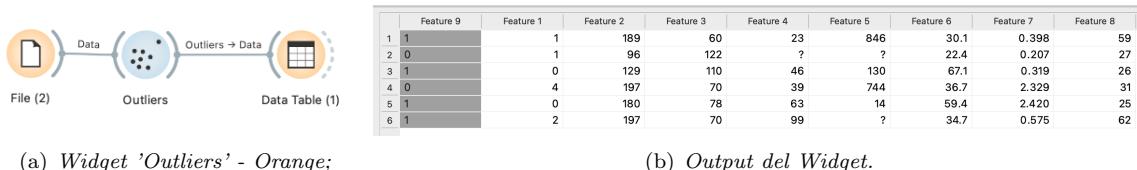


Figura 2.7:

Di questi 6 esempi del *dataset*, 5 mostrano un chiaro riscontro con l'ispezione visiva degli istogrammi. Per quanto riguarda il rimanente, non è possibile determinare per quale Feature il Software di Orange lo consideri come *outlier*, e pertanto è stato scartato, considerando in ultima istanza come tali solamente i primi 5 citati.

Possono essere infatti impostate delle **soglie** sui diversi parametri che debbano rispettare alcune caratteristiche fisiologiche. Ad esempio, le tabelle relative al **BMI**, considerano come 'pazienti gravemente obesi' già pazienti con un BMI di 42, pertanto un valore di 60 come riscontrato all'interno del nostro *dataset* viene considerato come un errore di misura.

Anche per quanto riguarda valori della **Plica Cutanea del Tricipite** superiori a 70mm viene fatto lo stesso ragionamento: infatti un Plicometro standard ha un range di funzionamento con un valore massimo di 70mm: un valore di 70mm è pertanto considerato come estremo all'interno del *dataset* e viene eliminato da esso.

ALTEZZA IN CM	PESO IN KG			SOTTOPESO			NORMOPESO			SOVRAPPESO			OBESITÀ			OBESITÀ ESTREMA								
	45.5	47.7	50.0	52.3	54.5	56.8	59.1	61.4	63.6	65.9	68.2	70.5	72.7	75.0	77.3	79.5	81.8	84.1	86.4	88.6	90.9	93.2	95.5	97.7
152.4	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
154.9	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
157.4	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
160.0	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
162.5	17	18	18	19	20	21	22	23	24	24	25	26	27	28	29	30	31	31	32	33	34	35	36	37
165.1	16	17	18	19	20	21	22	23	24	25	25	26	27	28	29	30	30	31	31	32	33	34	35	35
167.6	16	17	17	18	19	20	21	21	22	23	24	25	25	26	27	28	29	29	30	31	32	33	34	34
170.1	15	16	17	18	18	19	20	21	22	22	23	24	25	25	26	27	28	29	29	30	31	32	33	33
172.7	15	16	16	17	18	19	19	20	21	22	22	23	24	25	25	26	27	28	28	29	30	31	32	32
175.2	14	15	16	17	17	18	19	20	20	21	22	23	24	25	25	26	27	28	28	29	30	31	31	
177.8	14	15	15	16	17	18	18	19	20	20	21	22	23	23	24	25	25	26	27	28	29	30	30	
180.3	14	14	15	16	16	17	17	18	19	20	21	21	22	23	23	24	25	25	26	27	28	29	30	
182.8	13	14	14	15	16	17	17	18	19	19	20	21	21	22	23	23	24	25	25	26	27	28	29	
185.4	13	13	14	15	16	17	17	18	19	19	20	21	21	22	23	23	24	25	25	26	27	28		
187.9	12	13	14	14	15	16	16	17	18	18	19	19	20	21	21	22	23	23	24	25	25	26	27	
190.5	12	13	13	14	15	15	16	16	17	18	18	19	20	20	21	22	23	23	24	25	25	26	26	
193.0	12	12	13	14	14	15	15	16	17	17	18	18	19	20	20	21	22	23	23	24	25	25	26	

(a) Curve BMI



(b) Plicometro standard.

Figura 2.8:

Per quanto riguarda l'**insulina sierica**, sono considerati valori normali quelli compresi tra 65 e 100 mg/dl. Nel nostro caso le misurazioni sono espresse in **μ U/ml**, per confrontarle con le **tabelle standard** i valori vanno quindi convertiti considerando il fattore di conversione tipico della sostanza, che nel caso dell'insulina è pari a 6 mg/dl. Pertanto valori di 'Insuline' pari a 700 μ U/ml corrispondono a 4200 mg/dl: valori molto maggiori del 200 mg/dl etichettato in letteratura come 'paziente diabetico'.

Column	Threshold	Count outliers
Blood Pressure	> 120	1
Skin Thickness	> 70	1
Insulin	> 700	2
	< 15	1
BMI	> 60	1
Age	> 80	1

Figura 2.9: Thresholds.

Siccome si tratta di pochi dati, in percentuale inferiore al 5%, è quindi possibile eliminarli.

2.2.2 Feature Elimination

Come sottolineato in precedenza, sebbene in questo *dataset* nessuna colonna contenga valori mancanti, alcune delle misurazioni (Glucosio, Pressione Sanguigna, Skin Thickness, Insulina e BMI) hanno valori pari a 0, il che non è compatibile con un organismo umano vivente.

In particolare la Feature '**Insuline**' presenta quasi il **50%** dei valori pari a **zero**. Per questo motivo, nonostante l'intero *dataset* sia composto solamente di 8 Features (più la classe), essendo una percentuale così alta, si decide di **eliminare** direttamente la colonna della Feature per tutto il *dataset*. Fare Imputing su questa Feature polarizzerebbe troppo il *dataset*, non portando comunque una quantità di informazione rilevante per la classificazione.

Feature	Count(0)	Percentuale
Glucose	5	0.657030 %
BloodPR	35	4.599212 %
SkinTck	226	29.697766 %
Insulin	372	48.883049 %
BMI ind	11	1.445466 %

Figura 2.10: *Zeros*.

Anche la Feature 'Skin Thickness' presenta numerosi dati nulli (circa il 30% del totale), tuttavia per non ridurre troppo le dimensioni del *dataset* si decide di mantenerla.

2.2.3 *Imputing* dei dati Mancanti/Nulli

Al posto di eliminare i dati nulli, essendo essi particolarmente numerosi soprattutto nelle colonne di Skin Thickness e Insuline, si è preferito eseguire un ***Impute*** di essi, considerandoli come errori di inserimento nel *dataset*, e sostituendoli con il valore mediano e non medio della colonna, che appare più robusto rispetto alla presenza di eventuali altri *outliers*.

```
% rimozione dei NaN e Imputing con mediana

index = find(isnan(out_Glucose) == 1);
out_Glucose(index) = median(out_Glucose,'omitNaN');

clear index
index = find(isnan(out_Blood) == 1);
out_Blood(index) = median(out_Blood,'omitNaN');

clear index
index = find(isnan(out_Skin) == 1);
out_Skin(index) = median(out_Skin,'omitNaN');

clear index
index = find(isnan(out_BMI) == 1);
out_BMI(index) = median(out_BMI,'omitNaN');
```

Figura 2.11: *Impute*.

2.2.4 Normalizzazione

La Figura 2.12 mostra la **Media** e la **Deviazione Standard** di ciascun attributo all'interno del PIDD; come si può vedere, l'intervallo di valori tra gli attributi è elevato. I risultati sono coerenti con quanto riscontrato nel documento [3].

Feature	Media	StDev
Pregnancies	3.858081	3.370279
Glucose	121.300920	30.139240
BloodPR	72.285151	11.929129
SkinTck	28.939553	8.334079
BMI ind	32.390670	6.703540
DPF fun	0.467402	0.317952
Age	33.137976	11.592563

Figura 2.12: Mean & StDev Before Normalization.

Si decide dunque di **trasformare** i valori degli attributi in un nuovo **range compreso tra 0 e 1**. A tale scopo si utilizza l'apposito Widget di Orange denominato 'Preprocess' → 'Normalize Features' → 'Normalize to interval [0,1]'; ottenendo come Medie e Deviazioni standard i risultati riportati in Figura 2.13.

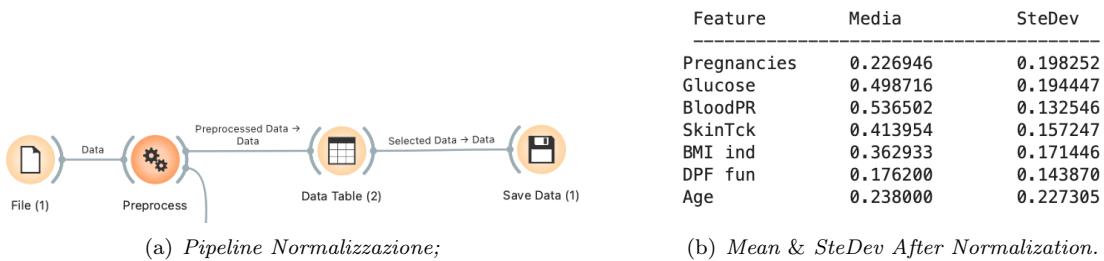


Figura 2.13:

La Normalizzazione viene eseguita su tutto il dataset, prima della divisione in *Training set* e *Test set* e la formula usata per la Normalizzazione è definita '**Metodo Min-Max**' ed è la seguente:

$$Norm = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.1)$$

2.2.5 Divisione Trainig Set - Test Set

Una volta ripulito dagli *outliers*, migliorato con la tecnica di *Imputing*, e in fine Normalizzato, il *dataset* viene dunque suddiviso in una parte costituente il **Training Set** (70% dei dati), e in una parte comprendente il **Test Set** (30% del dataset). La divisione viene fatta con l'ausilio del Widget 'Data Sampler' di Orange come mostrato in Figura 2.23.

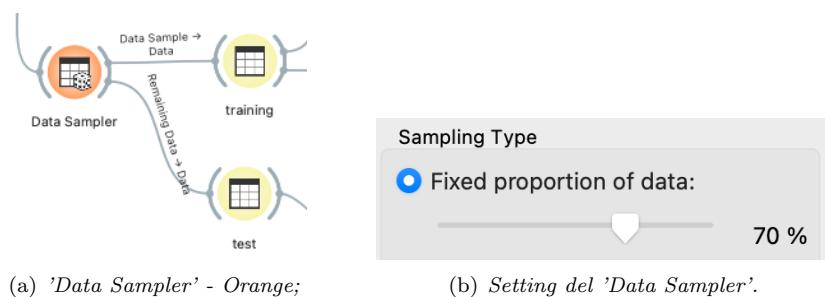


Figura 2.14:

Il Training set subirà ulteriori modifiche, mentre il Test set arriverà inalterato alla fine del processo, quando sarà usato per testare i differenti Metodi di Classificazione.

2.2.6 Bilanciamento delle Classi, SMOTE & *Data Augmentation*

Si può notare come all'interno del nostro *dataset* le due **classi** siano leggermente **sbilanciate**. La percentuale della classe 0 è del **65%**, mentre della classe 1 è del **35%**. Si tratta di un caso ***borderline*** di sbilanciamento, nel quale dunque non risulta strettamente necessario un intervento di ribilanciamento delle classi.

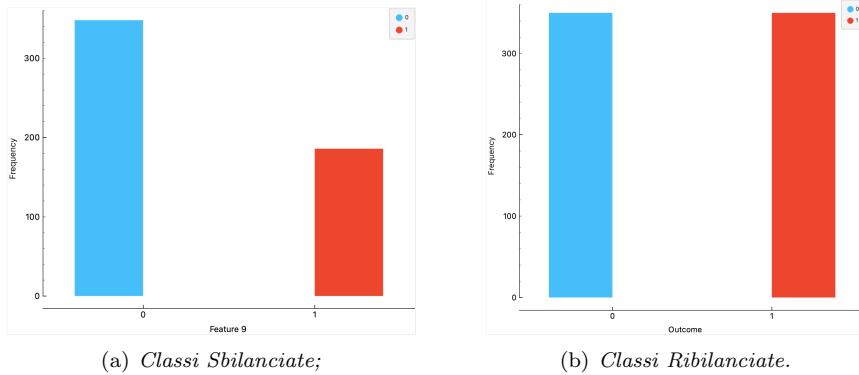


Figura 2.15:

Si decide dunque di procedere testando i Metodi di Classificazione su **3 Training set parallelamente**, per poi confrontare i risultati ottenuti:

- Training set **Originale** 'sbilanciato',
 - Training set 'ribilanciato' con algoritmo **SMOTE**,
 - Training set 'ribilanciato' con **Data Augmentation**.

La tecnica SMOTE [4] (Synthetic Minority Oversampling Technique) è una tecnica statistica per aumentare il numero di casi nel set di dati in modo bilanciato. L'algoritmo funziona generando nuove istanze a partire da casi di minoranza esistenti forniti come input. Il vantaggio di SMOTE è che non agisce generando duplicati (come avviene nell'oversampling), che renderebbero i dati fortemente dipendenti, ma piuttosto creando nuovi dati sintetici leggermente diversi dai dati originali. Viene riportato lo **pseudocodice** (Fig. 2.17).

```

%% -----
% SMOTE - Synthetic Minority Over-sampling Technique
clc
clear all

% lo faccio solo sul training set quindi devo importare il training set
% dopo la divisione in orange

x = [1 2 3 4 5 6 7];
Training = readtable("Training.xlsx");
A = Training(:,x);
Class_A = table2array(Training(:,end));
T = table2array(A);

% magheggio xk orange è mongo
Training.Properties.VariableNames=["Pregnancies","Glucose","Blood Pressure","Skin Thickness","BMI","DPF","Age","Outcome"];
writetable(Training,"TrainingOriginal.csv");

% richiamo l'algoritmo SMOTE come implementato nella funzione apposita che
[B,C,Xn,Cn] = smote(T,[],3, 'Class',Class_A);

```

Figura 2.16: *Bilanciamento - SMOTE*.

L'algoritmo prende dunque in *input* i dati appartenenti al Training set e che sono già stati Normalizzati in precedenza. L'implementazione consta di una funzione Matlab riadattata a partire da una **funzione MyFun** precedentemente creata da alcuni utenti del Software.

```

Algorithm SMOTE(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest
neighbors k
Output: (N/100) * T synthetic minority class samples
1. (* If N is less than 100%, randomize the minority class samples as only a random
percent of them will be SMOTEd. *)
2. if N < 100
3.   then Randomize the T minority class samples
4.   T = (N/100) * T
5.   N = 100
6. endif
7. N = (int)(N/100) (* The amount of SMOTE is assumed to be in integral multiples of
100. *)
8. k = Number of nearest neighbors
9. numattrs = Number of attributes
10. Sample[ ][ ]: array for original minority class samples
11. newindex: keeps a count of number of synthetic samples generated, initialized to 0
12. Synthetic[ ][ ]: array for synthetic samples
(* Compute k nearest neighbors for each minority class sample only. *)
13. for i ← 1 to T
14.   Compute k nearest neighbors for i, and save the indices in the nnarray
15.   Populate(N, i, nnarray)
16. endfor

Populate(N, i, nnarray) (* Function to generate the synthetic samples. *)
17. while N ≠ 0
18.   Choose a random number between 1 and k, call it nn. This step chooses one of
the k nearest neighbors of i.
19.   for attr ← 1 to numattrs
20.     Compute: dif = Sample[nnarray[nn]][attr] – Sample[i][attr]
21.     Compute: gap = random number between 0 and 1
22.     Synthetic[newindex][attr] = Sample[i][attr] + gap * dif
23.   endfor
24.   newindex++
25.   N = N – 1
26. endwhile
27. return (* End of Populate. *)
End of Pseudo-Code.

```

Figura 2.17: *Pseudo Codice SMOTE*.

L'algoritmo viene usato quindi inizialmente per il **ribilanciamento** delle classi senza cambiare il numero di dati appartenente alla classe di maggioranza, ma generando solamente nuovi dati appartenenti a quella di minoranza. In seguito si decide di provare anche la tecnica della *Data Augmentation*, sempre attraverso l'algoritmo SMOTE e **generando** un numero maggiore di dati di Training, sempre bilanciato ma che quindi coinvolge in questo caso non solo la classe di minoranza ma anche quella di maggioranza.

'Giocando' con i parametri dell'algoritmo è possibile decidere **di quanto aumentare** il set di dati per la creazione del Training set 'Aumentato'.

```

% quante volte ogni osservazione verrà usata come base di sintesi
obs_0 = 2*(strcmp(Class_A,'0'));
obs_1 = 5*(strcmp(Class_A,'1'));
obs = obs_0 + obs_1;

% richiamo l'algoritmo SMOTE come implementato nella funzione apposita che
[B1,C1,Xn1,Cn1] = smote(T,[],3, 'Class',Class_A,'SynthObs',obs);
% [B1,C1,Xn1,Cn1] = smote(T,[],3, 'Class',Class_A);

```

Inizialmente si ottenevano circa 7000 **dati totali di Training 'Aumentato'**, ma sotto consiglio della Professoressa Sacchi si decide di diminuire tale quantità a **circa 2000**. Infatti generando troppi dati si rischia di distorcere quelle che sono le informazioni intrinseche originali del *dataset*.

Viene eseguito un '*kstest*' per verificare che i risultati di SMOTE appartengano alla stessa distribuzione 'naturale' dei dati iniziali. Il **test di Kolmogorov Smirnov** conferma che sia i dati 'ribilanciati', sia quelli frutto della *Data Augmentation* appartengono alla stessa distribuzione iniziale. Viene riportato il codice Matlab utilizzato (Fig. 2.18).

```
% ks test
for i = 1:8

    H_SMOTE(i) = kstest2(sottomatrice(:,i),sottomSMOTE(:,i));
    H_DataAug(i) = kstest2(sottomatrice(:,i),sottomDataAug(:,i));

end
```

Figura 2.18: *kstest*

Infine, a seguito del processo descritto vengono **riscalati** i valori ottenuti per le Features 'Pregnancies' e 'Age' che sono attributi discreti. I dati in *output* dall'algoritmo SMOTE vengono arrotondati al valore più simile tra le osservazioni pre-SMOTE. Viene riportato il codice Matlab utilizzato (Fig.2.19).

```
% -----
% i dati di Pregnancies e Age devono essere sistemati
% vettore 1 da arrotondare
% vettore 2 riferimento per arrotondarli

vettore1Preg = B(:,1);
vettore2Preg = unique(T(:,1));

% Arrotondamento dei dati di vettore1 al valore più simile di vettore2
vettArrPreg = zeros(size(vettore1Preg));

for i = 1:length(vettore1Preg)
    dato = vettore1Preg(i);
    [~, indice_simile] = min(abs(vettore2Preg - dato));
    vettArrPreg(i) = vettore2Preg(indice_simile);
end

% Faccio la stessa cosa con Age
vettore1Age = B(:,7);
vettore2Age = unique(T(:,7));

clear dato indice_simile i
vettArrAge = zeros(size(vettore1Age));

for i = 1:length(vettore1Age)
    dato = vettore1Age(i);
    [~, indice_simile] = min(abs(vettore2Age - dato));
    vettArrAge(i) = vettore2Age(indice_simile);
end

B(:,1) = vettArrPreg;
B(:,7) = vettArrAge;
```

Figura 2.19: *Arrotondamento.*

2.2.7 Feature Selection

Usando le Widget 'Scatter plot' e 'Correlation' di Orange è possibile considerare se tra le variabili dei diversi Training set ve ne siano di correlate. L'analisi tramite *Feature Selection* viene infatti eseguita su tutti e tre i set parallelamente ma vengono riportati soltanto gli **output** che riguardano il Training set 'Originale' sbilanciato, perché gli altri due mostrano sostanzialmente i medesimi risultati qualitativi.

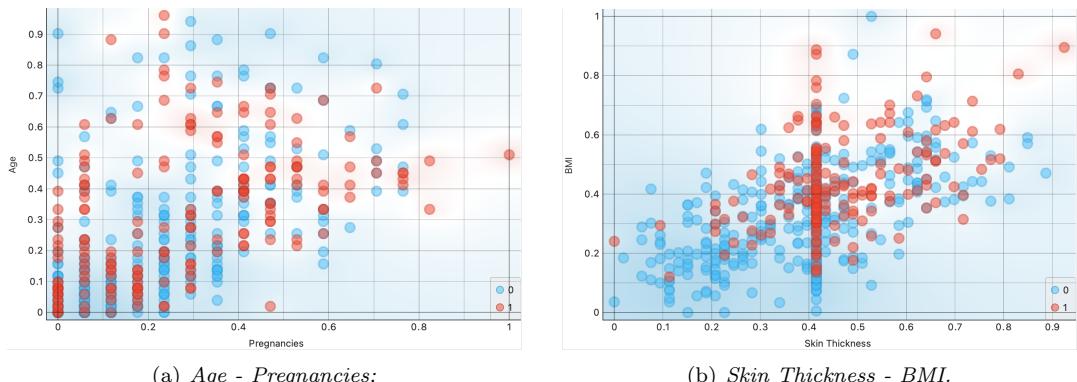
1	+0.563	Age	Pregnancies
2	+0.541	BMI	Skin Thickness
3	+0.362	Age	Blood Pressure
4	+0.264	BMI	Blood Pressure
5	+0.255	BMI	Glucose
6	+0.247	Age	Glucose
7	+0.234	Blood Pressure	Glucose
8	+0.225	Blood Pressure	Pregnancies
9	+0.192	Glucose	Skin Thickness
10	+0.177	BMI	DPF
11	+0.175	Blood Pressure	Skin Thickness
12	+0.123	DPF	Glucose
13	+0.112	Age	Skin Thickness
14	+0.107	Glucose	Pregnancies
15	+0.100	DPF	Skin Thickness
16	+0.086	Pregnancies	Skin Thickness

Figura 2.20: *Output di 'Correlation' - Orange.*

L'output di 'Correlation' dunque suggerisce in particolare che potrebbe esserci correlazione tra:

- Pregnancies e Age
- BMI e Skin Thickness

Per quanto riguarda '**Age**' e '**Pregnancies**', è evidente che all'aumentare dell'età della donna, si ha più probabilità che essa abbia avuto più gravidanze. Questo sia per un fattore di tempistiche di gestazione ma anche perché un tempo si tendeva a procreare più figli che oggi. Tuttavia risulta essere una correlazione non sufficiente da pensare di eliminare una delle due Features. La decisione in tal senso è anche motivata dalla scelta precedente di eliminare la Feature 'Insuline': il *dataset* ora consta di sole 7 Features e sarebbe eccessivo eliminarne di ulteriori.

Figura 2.21: *'Scatter Plot' - Orange*

Osservando lo 'Scatter Plot' di '**Skin Thickness**' e '**BMI**' sembra esserci correlazione: infatti la plica del tricipite sarà più spessa quanto più la paziente avrà una componente adiposa elevata (e quindi quanto più sarà elevato il suo BMI). Tuttavia, anche in questo caso, si tratta di una correlazione non sufficiente per eliminare uno dei due attributi.

Come suggerito dalla Professoressa Sacchi, si provano anche diverse altre tecniche per la Feature Selection. Tra esse possono essere consultate nella 'Pipeline' di Orange (Fig. 2.22) le Widget 'Rank', 't-SNE', 'MDS' e 'PCA' .

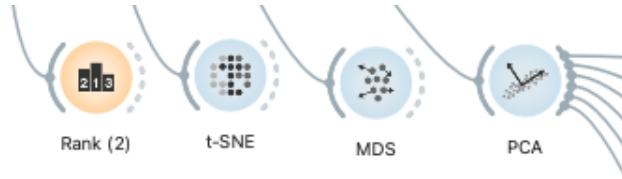


Figura 2.22: Caption

L'output di 'Rank' mostra gli stessi risultati qualitativi ottenuti inizialmente durante l'ispezione del *dataset*. Gli output di 't-SNE' e di 'MDS' variano leggermente nei diversi casi di Training set, ma mostrano sostanzialmente le stesse informazioni. **T-SNE** in particolare è un algoritmo di riduzione dimensionale utilizzato per visualizzare dati ad alta dimensionalità in uno spazio a dimensioni ridotte; il suo output in questo caso ci suggerisce che si tratta di un **problema di difficile classificazione non separabile linearmente**.

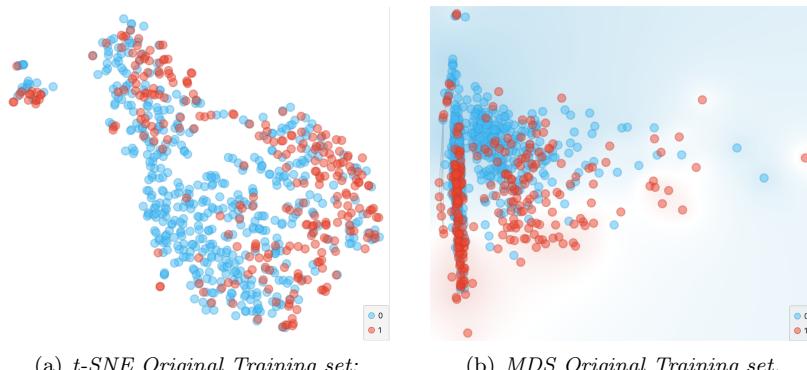


Figura 2.23: Feature Selection

2.2.8 Discretizzazione

Il processo di Discretizzazione viene eseguito **solo per quei metodi che richiedono dati discreti** e riguarda tutte le Features del *dataset*. Viene usato il Widget di Orange 'Preprocess' → 'Discretize Continuous Variables' → 'Entropy-MDL Discretization'. L'*Entropy-MDL discretization* risulta essere il metodo migliore poiché consente di trovare il bilanciamento ottimale tra la riduzione dell'entropia e la complessità del modello.

Capitolo 3

Modellizzazione e valutazione

”Il Machine Learning è una branca della scienza che consente ai computer di essere intelligenti come gli umani, migliorando automaticamente la loro comprensione attraverso l’esperienza, in modo che possano prendere decisioni senza essere ripetutamente programmati”[5].

Il Machine learning si divide in 4 categorie: 1. Supervised Learning - 2. Semi-Supervised Learning - 3. Unsupervised Learning - 4. Reinforcement Learning. In particolare l'**apprendimento supervisionato** è una tecnica di apprendimento automatico che utilizza set di dati etichettati al fine di effettuare previsioni e classificazioni.

Per quanto riguarda quali metodi utilizzare per lo specifico problema di classificazione del Diabete Mellitus, la scelta è stata basata su un'**analisi della letteratura** di 20 Articoli [[1] [2] [6] [5] [7] [3] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21]] trovati tramite **PubMed** che facessero uso del dataset PIDD. È stata contata la **frequenza di utilizzo** di ogni Classificatore incontrato: il risultato è osservabile nel diagramma seguente 3.1.

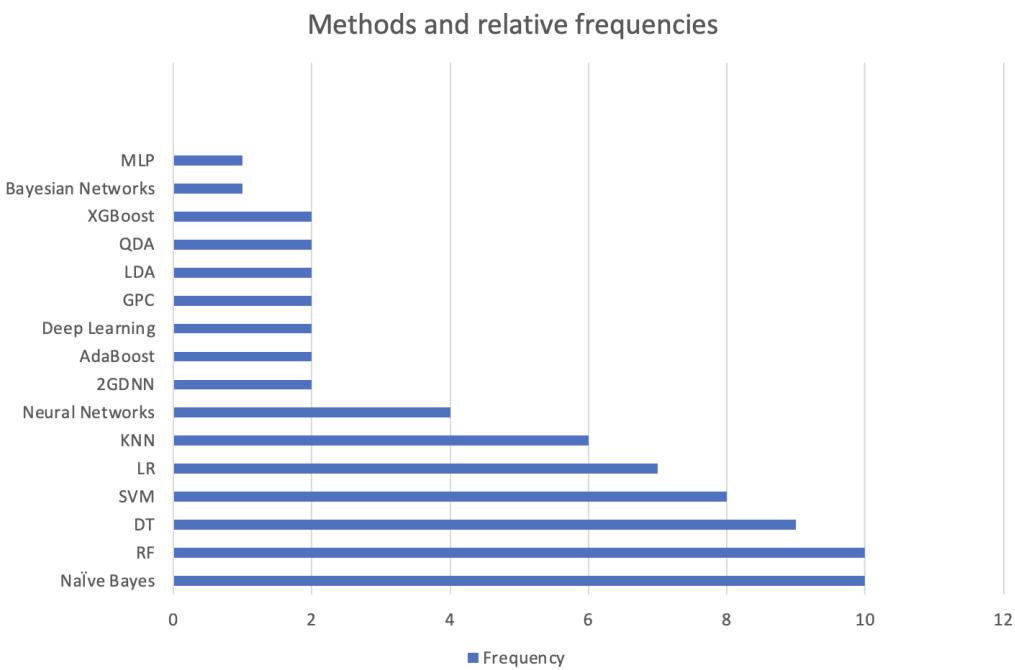


Figura 3.1: *Methods and relative Frequencies.*

Sono stati scelti i 6 metodi più frequenti.

3.1 I diversi Metodi considerati: scelta degli iperparametri

Vengono brevemente descritti i Modelli utilizzati per la classificazione sottolineando alcune loro caratteristiche peculiari. Per quanto riguarda il *set* degli iperparametri di modello, esistono alcune tecniche di *Tuning* dei parametri che consentono di provare iterativamente diversi valori di parametrizzazione per scegliere poi quelli che massimizzano gli indici decisionali. Tuttavia si teme che in tal modo vi sia il rischio di ricadere nell'*overfitting*. Si preferisce dunque tener conto delle **parametrizzazioni** proposte dal ***Tuning*** per quanto riguarda un 'range' entro il quale stare, ma prediligere in ultima istanza il ragionamento logico e il '**settaggio manuale**'.

3.1.1 Naïve Bayes - (NB)



Naïve Bayes

Figura 3.2: Widget Naïve Bayes - Orange

Naïve Bayes è un algoritmo di classificazione per problemi di classificazione binaria (a due classi) e multiclasse. Si tratta di un classificatore statistico che funziona secondo il teorema di Bayes, classificando i dati in categorie predeterminate utilizzando la probabilità condizionale. Una **Regola Bayesiana** è un approccio utilizzato per stimare la possibilità di un attributo dato un set di dati come input.

Viene chiamato **Naïve** perché il calcolo della probabilità per ogni ipotesi è semplificato per rendere il calcolo praticabile. Si basa dunque su un presupposto molto forte (e anche molto improbabile nei dati reali) di indipendenza tra attributi.

Tale metodo ha diversi meriti: è facile da usare e la quantità di dati di Training di cui necessita per la classificazione non è necessariamente grande. Inoltre, sebbene il classificatore sia progettato in modo 'Naïve' e la sua assunzione sembri troppo semplice, si comporta bene in una serie di complicate situazioni del mondo reale.

3.1.2 Support Vector Machines - (SVM)



SVM

Figura 3.3: Widget SVM - Orange

Quello delle Support Vector Machines è l'algoritmo principalmente preferito per l'apprendimento automatico supervisionato grazie alla sua semplicità, elevata precisione e affidabilità. Può gestire in modo efficace sia set di dati separabili linearmente che non separabili linearmente (come nel nostro caso).

Tale metodo mira a trovare il **confine ottimale (iperpiano)** tra i punti dati nello spazio delle caratteristiche, confine che massimizza il margine di separazione tra due classi. Questo può essere costruito con l'**aiuto di vettori di supporto** per avere un buon margine e per migliorare la classificazione.

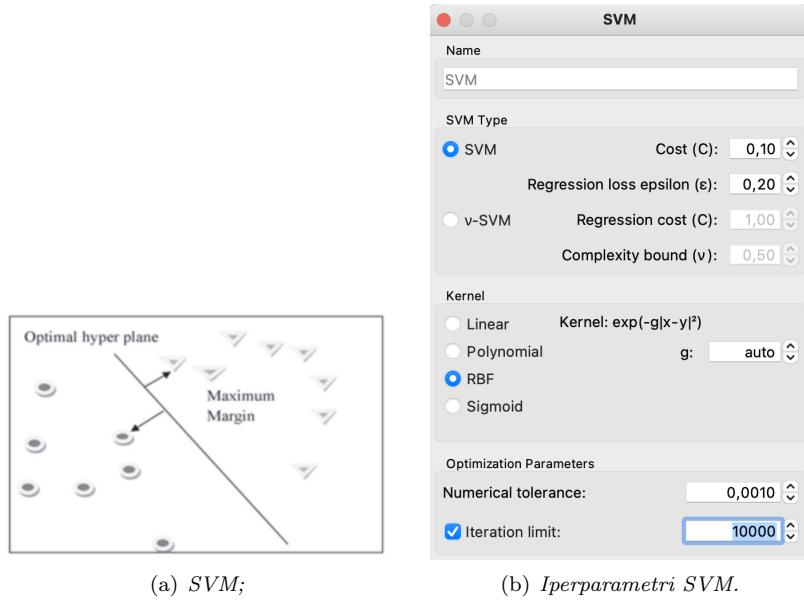


Figura 3.4:

La maggior parte dei dati del mondo reale è per lo più non lineare. I problemi non lineari in SVM vengono risolti mappando lo spazio di input n-dimensionale in uno spazio di caratteristiche ad alta dimensione in cui SVM può ancora operare in modo lineare.

3.1.3 Decision Tree - (DT)

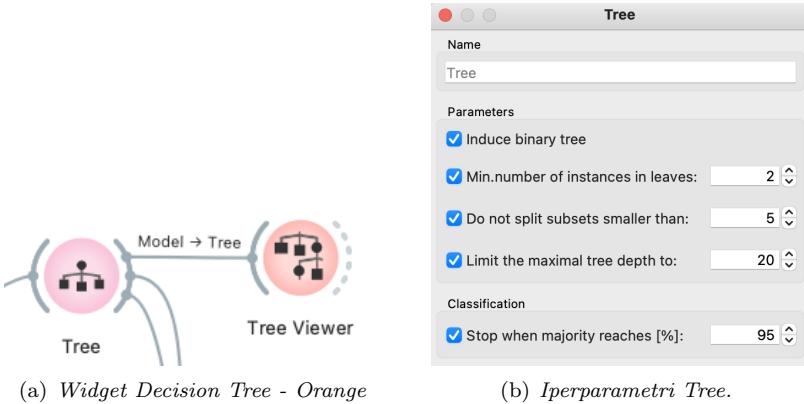


Figura 3.5:

L'albero decisionale segue l'approccio '**Divide et Impera**' in cui il *dataset* viene diviso in sottoinsiemi e rappresentato in una struttura ad albero. Un nodo foglia in un albero decisionale rappresenta i risultati della classificazione e un nodo interno rappresenta il giudizio degli attributi.

Il primo passo è selezionare l'attributo più appropriato per il nodo radice; la suddivisione inizia con una suddivisione binaria e procede fino a quando non è possibile effettuare più partizioni. La struttura è formata utilizzando il concetto di **Information Gain** e, a seconda dell'entità delle informazioni, il classificatore esegue

la classificazione. L'obiettivo di questo modello è quello di incapsulare i dati nel più piccolo albero possibile. Questo perché i piccoli alberi prendono decisioni più rapidamente rispetto agli alberi più grandi e sono molto più facili da interpretare.

3.1.4 Random Forest - (RF)

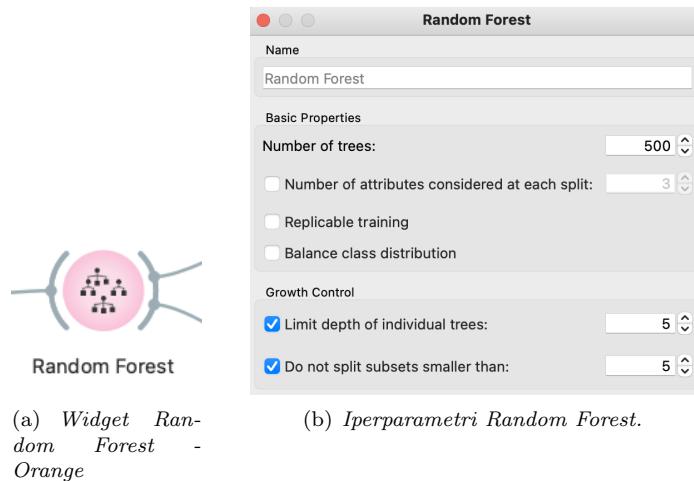
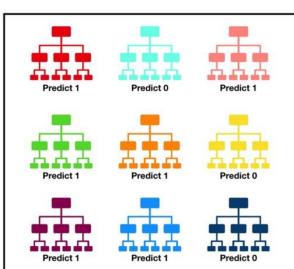


Figura 3.6:

Il classificatore Random Forest viene utilizzato principalmente per classificare i set di dati con valori mancanti. Attraverso RF vengono estratte le migliori caratteristiche per classificare il set di dati. La previsione del modello può assumere due forme: se l'output è un valore medio, allora RF risolve un problema di regressione, mentre se l'output è un modo delle classi, allora RF risolve un problema di classificazione, come nel nostro caso.

Random Forest utilizza un concetto di base semplice e potente, chiamato '***La Saggezza della Folla***': è un'estensione di un albero decisionale ed è composto da numerosi alberi decisionali singoli, ciascuno dei quali produce una categoria di risultati di previsione. Per un certo *dataset* vengono estratti campioni casuali e questi vengono inviati all'albero decisionale per ottenere l'output. Ogni albero utilizza il proprio meccanismo per classificare il campione di dati in una classe. La **classe con il maggior numero di voti** nella foresta costituisce il risultato di previsione finale e viene utilizzato per la classificazione.



Ad esempio, come mostrato affianco, tra nove alberi decisionali singoli nella foresta, i risultati della previsione di sei alberi sono 1 e quelli dei restanti tre alberi sono 0. Pertanto, il risultato della previsione della RF è 1.

La chiave per il buon funzionamento di questo classificatore è che gli alberi nella foresta siano relativamente indipendenti l'uno dall'altro, assicurando che la decisione che prendono nel loro insieme sia migliore delle decisioni prese da ciascuno di loro individualmente. In altre parole, il modello Random Forest funziona bene perché numerosi modelli relativamente non correlati che operano nel loro insieme hanno prestazioni migliori rispetto a qualsiasi singolo modello costituente.

3.1.5 Logistic Regression - (LR)

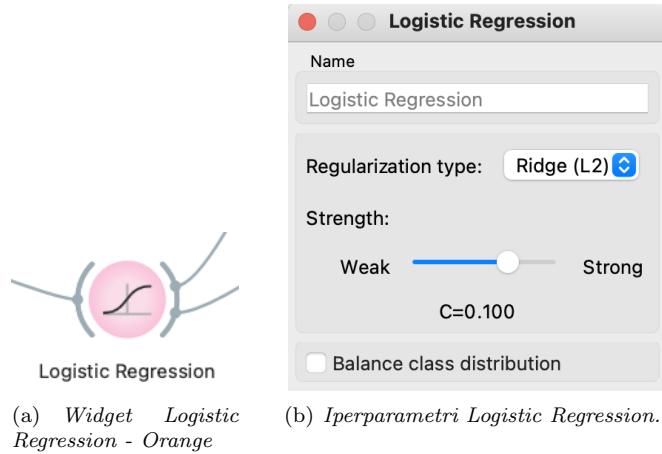


Figura 3.7:

Il modello di regressione logistica è un algoritmo di apprendimento supervisionato utilizzato per affrontare **soltamente problemi di classificazione binaria** (ed è il nostro caso).

La regressione logistica è un algoritmo di classificazione che può gestire sia variabili continue che variabili categoriche senza richiedere la discretizzazione del *dataset*. Questo modello prevede la probabilità che un'istanza appartenga a una determinata classe utilizzando una funzione logistica, che mappa l'input in un valore compreso tra 0 e 1.

Il modello di regressione logistica viene addestrato utilizzando un Training set etichettato. Durante il processo di addestramento, i coefficienti del modello ($b_0, b_1, b_2, \dots, b_n$) vengono ottimizzati per minimizzare una funzione di perdita, come ad esempio la **Log Loss**.

3.1.6 k-Nearest-Neighbour - (KNN)

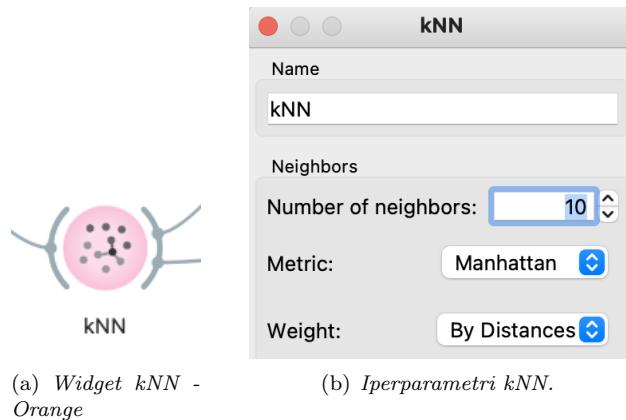


Figura 3.8:

k-Nearest-Neighbour (KNN) è un semplice algoritmo di apprendimento supervisionato per classificare un determinato set di dati. Il **Neighbour** più vicino è misurato dalla funzione di distanza (che può essere: Euclidea,

Manhattan, Hamming o Minkowski).

Si selezionano i K punti di addestramento più vicini (i "vicini") all'istanza di test sulla base della distanza calcolata. Il valore di K è un iperparametro del modello e deve essere specificato in anticipo. La scelta del valore di K è cruciale. Valori **K troppo piccoli** possono essere sensibili al rumore e causare **overfitting**, mentre valori **K troppo grandi** possono portare a una perdita di informazioni e causare **underfitting**.

Si determina la classe di appartenenza dell'istanza di test sulla base delle classi dei K vicini. Ciò può essere fatto utilizzando una regola di voto di maggioranza, '*Rule of Thumb*', in cui la classe più frequente tra i vicini determina la classe predetta per l'istanza di test. Quando vengono visualizzati nuovi dati, possono essere facilmente classificati mettendoli nella categoria con la minima distanza dal dato.

K-NN ha alcune considerazioni importanti da prendere in esame:

- La normalizzazione delle caratteristiche può essere necessaria per garantire che le diverse caratteristiche abbiano un impatto equilibrato sul calcolo delle distanze.
- K-NN può essere computazionalmente costoso, poiché richiede il calcolo delle distanze per tutte le istanze di addestramento.

3.2 Tecniche di Valutazione

Vengono utilizzate diverse tecniche per effettuare la valutazione dei Classificatori:

1) Specificity

È la proporzione di pazienti senza diabete, i casi negativi, identificati come non diabetici ed è calcolata come rapporto tra veri negativi (TN) e somma di (TN) e falsi positivi (FP).

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (3.1)$$

2) Sensitivity - Recall

Questa è la proporzione di pazienti con diabete, i casi positivi, correttamente identificati come diabetici ed è calcolata come rapporto tra veri positivi (TP) e somma di (TP) e falsi negativi (FN).

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (3.2)$$

3) Precision

Si tratta della proporzione di pazienti con diabete, i casi positivi, correttamente identificati come diabetici su tutti i pazienti diabetici ed è calcolata come rapporto tra (TP) e somma di (TP) e falsi positivi (FP).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.3)$$

4) F1-Score

Questa è la media pesata di precisione e richiamo. Di conseguenza, questo punteggio considera sia i falsi positivi che i falsi negativi.

$$F1 - Score = 2 * \left[\frac{(Recall * Precision)}{(Recall + Precision)} \right] \quad (3.4)$$

5) Accuracy

È il rapporto tra il numero totale di previsioni corrette e il numero totale di previsioni.

$$Precision = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.5)$$

6) Matthews Correlation Coefficient - MCC

L'MCC è un coefficiente di correlazione tra la classificazione effettiva e la classificazione prevista. Il suo intervallo di valori è [-1, 1]. Quando l'MCC è uguale a uno, indica una previsione perfetta per il soggetto. Quando il valore MCC è 0 indica che il risultato previsto è al pari del risultato di una previsione casuale e -1 significa che la classificazione prevista è completamente incoerente con la classificazione effettiva.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FN) * (TN + FP) * (TP + FP) * (TN + FN)}} \quad (3.6)$$

Capitolo 4

Scelta e raffinamento del modello

4.1 10-Fold Cross Validation

Si decide di eseguire una **10-Fold Cross Validation** per testare i diversi modelli scelti inizialmente a livello di Training set. A tale scopo viene utilizzato il Widget 'Data Sampler' di Orange, spuntando 'Replicable' e 'Stratify Sample' per poter ripetere il test per le 10 volte di cui consta l'esperimento e per ottenere Fold bilanciati a livello di classe. La pipeline viene collegata in modo da considerare 10 Fold di cui 9 di Training e 1 di Test.

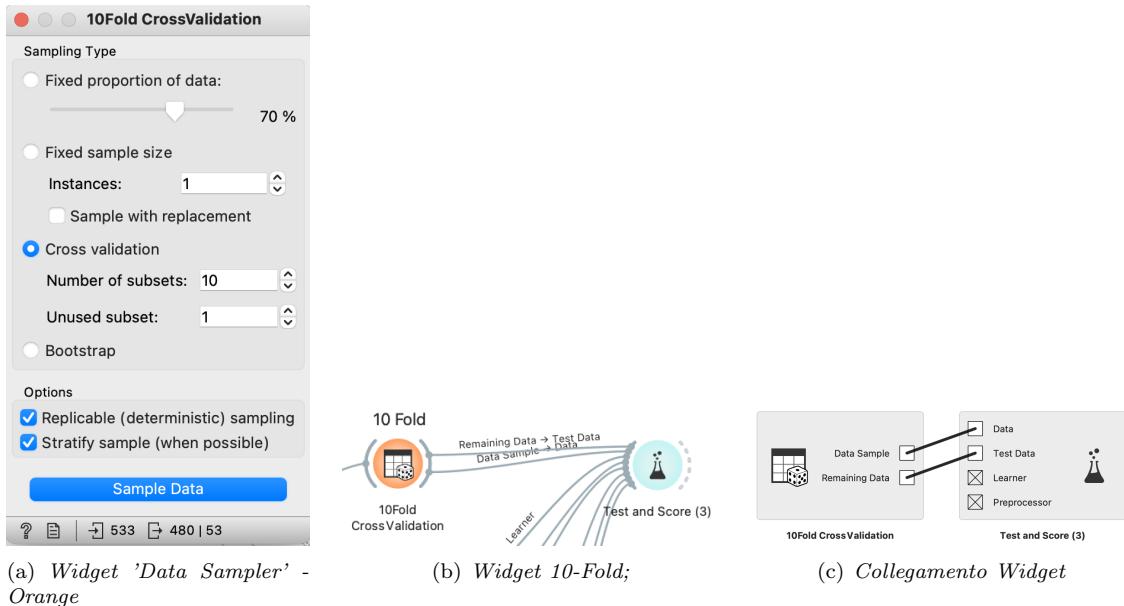


Figura 4.1:

4.1.1 Training set Originale

Per quanto riguarda il **Training set Originale**, viene eseguita una sola 10-Fold analizzata poi tramite **Test Two-Way ANOVA** previo **'kstest'** per confermare l'**ipotesi di normalità delle accuratezze**. Viene usata ANOVA2 per testare il contributo dei Classificatori al variare della composizione dei Fold (le due vie sono quindi il Classificatore e il Fold).

```

for i = 1:6
    H(i)=kstest((A(i,:)-mean(A(i,:)))./std(A(i,:)));
end

[p,tbl,stats] = anova2(A');
c1 = multcompare(stats,'Estimate','column');

```

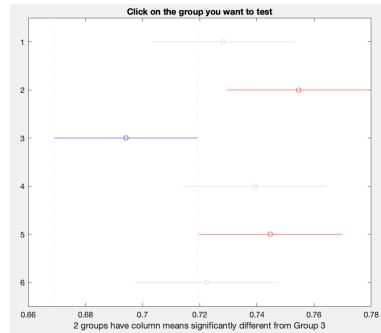
(a) *kstest*;(b) *Multiple Comparison - ANOVA2*.

Figura 4.2:

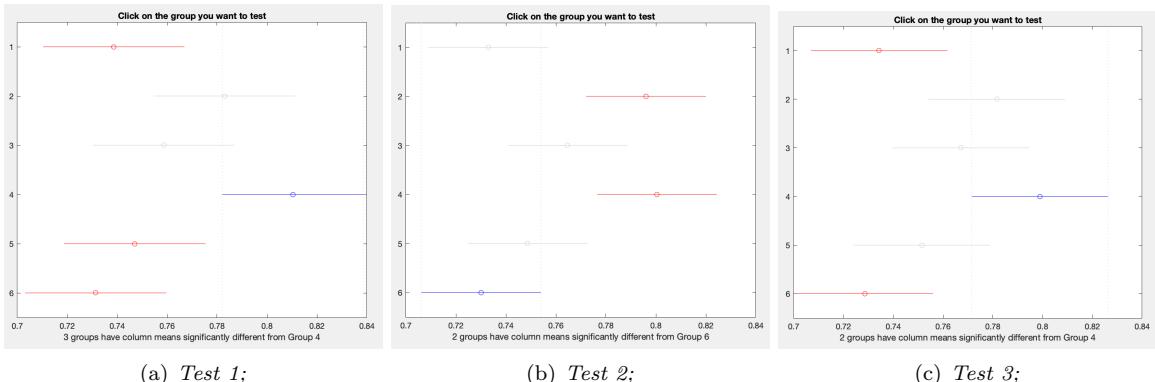
Come si può notare dalla Figura 4.2, si riscontra una differenza statisticamente significativa tra il Classificatore Decision Tree [3] e i Classificatori **Random Forest** [2] e **SVM** [5], che presentano una accuratezza migliore; questi ultimi due non mostrano una differenza significativa tra loro e possono essere considerati equivalentemente performanti. Non si riscontra una differenza significativa, invece, con i Classificatori Naïve Bayes [1], kNN [4] e Logistic Regression [6].

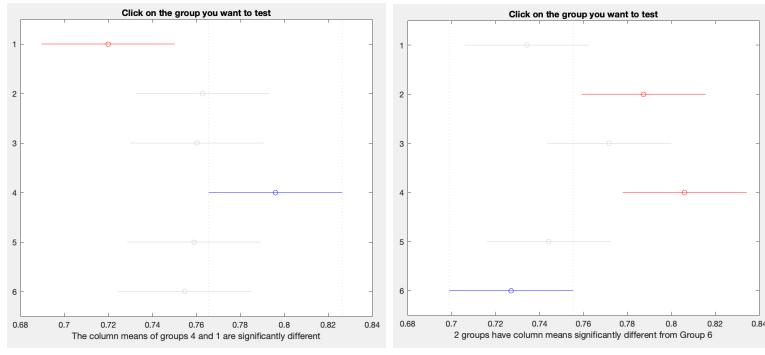
4.1.2 Training set SMOTE

Si vuole verificare la **solidità** dei risultati ottenuti dai diversi Classificatori quando vengono testati sul Training set bilanciato con algoritmo SMOTE. Ovvero si vuole capire se la generazione di nuovi dati al fine di bilanciare il *dataset* possa portare, nel caso dell'algoritmo SMOTE, a risultati randomici o se i diversi Classificatori si comportano allo stesso modo nonostante siano allenati con **diversi dataset contenenti dati generati**.

A tale scopo vengono generati **5 set 'SMOTE'** che vengono testati su **5 diverse 10-Fold Cross Validation** tramite ANOVA2, come nel caso precedente. I risultati ottenuti sono comparabili al caso Originale a livello di significatività statistica. Non risultano nuovamente particolarmente performanti i classificatori Naïve Bayes [1] e Logistic Regression [6]. Non ottiene accuratezze elevate nemmeno il Classificatore SVM [5] che invece performava bene nel caso di Training Originale.

Le migliori accuratezze variano al variare del test svolto ma mostrano una vittoria, seppur non schiacciatrice, del Classificatore **kNN** [4] (che nel caso di Trainin Originale non spiccava), e da **Random Forest** [2] (che otteneva buoni risultati già in precedenza). Mentre vi è significatività di differenza tra questi due Classificatori e i rimanenti, non vi è invece una differenza significativa tra di loro: possono essere dunque considerati al pari.





4.1.3 Training set *Data Augmentation*

Vengono generati 5 diversi Training set anche per quanto riguarda la *Data Augmentation*. In questo caso i risultati ottenuti sono tutti uguali nonostante varino i Training set e le composizioni dei 10 Fold. Il Training Aumentato enfatizza sostanzialmente le caratteristiche già viste nel Training SMOTE, portandole all'estremo: infatti le significatività statistiche non sono più paragonabili ai casi precedenti e risultano essere molto più marcate.

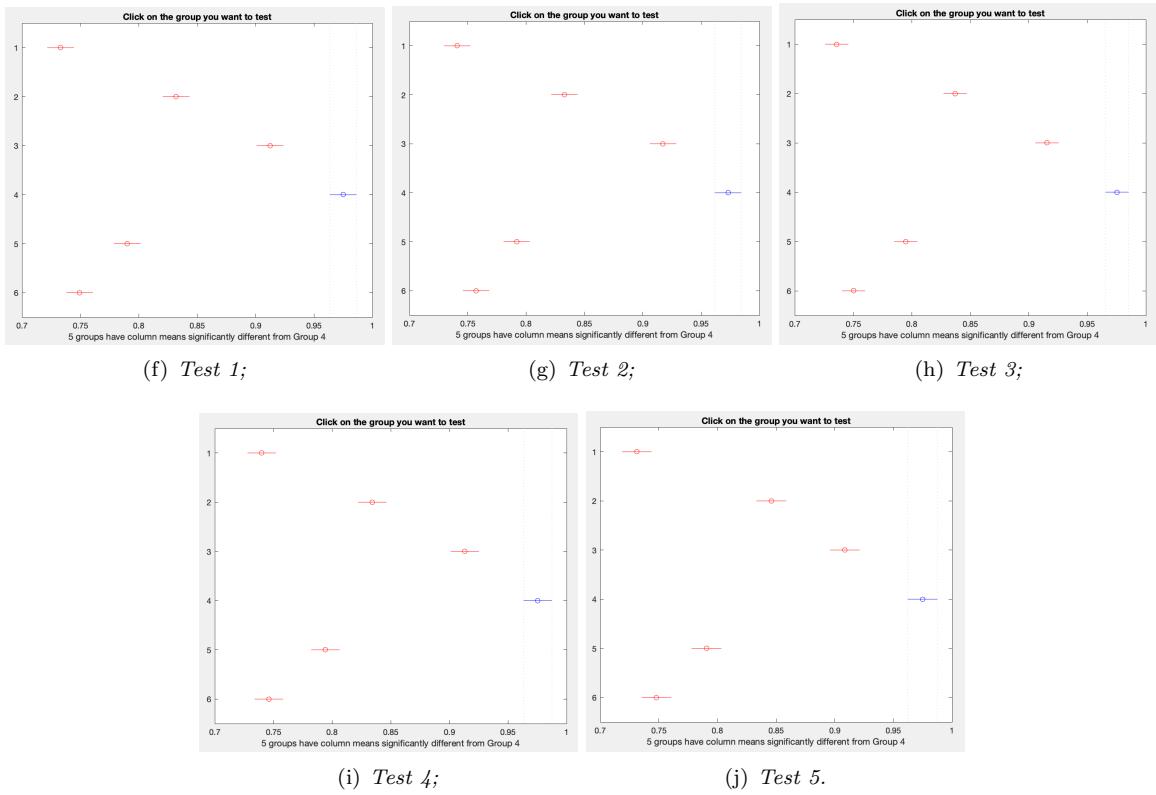


Figura 4.3: *Multcompare ANOVA - Data Augmentation*

Analogamente a prima, Naïve Bayes [1] e Logistic Regression [6] hanno differenza maggiore dal Classificatore migliore, segue il Classificatore SVM [5] che anche in questo caso mostra prestazioni peggiorate rispetto al caso di Training Originale. Peggiorano notevolmente le capacità predittive del Classificatore Random Forest [2],

mentre sembrano migliorare quelle del Decision Tree [3] che nel Training Originale era in fondo alla classifica. In tutti e 5 i test il Classificatore **kNN** [4] presenta delle performance statisticamente superiori agli altri Metodi di Classificazione.

In Figura 4.4 vengono riportate le accuratezze ottenute in per dimostrare che non si tratta di un unico esperimento 'camuffato' per ripetuto: come si può notare, il Classificatore kNN raggiunge delle accuratezze molto elevate, talvolta anche pari a 1.

```
% Prova 1
% run Prova1DataAug.m
%
% Prova 2
% Naive = [0.7268518518518519 0.7316111111111112 0.78703703703703 0.7534683739380232 0.7289302325581395409 0.7302325581395409 0.7441860465116279]
% Random = [0.8378939392952629 0.8467322222222222 0.8564914814814815 0.8186046511627907 0.8418604651162791 0.81329393225581394 0.8418604651162791 0.788946511627907]
% SVM = [0.914814814814814815 0.9222222222222222 0.914814814814814815 0.8861111111111112 0.9627906976744186 0.95813953488721 0.9728930232558139 0.9728930232558139]
% Tree = [0.9385555555555556 0.9120370370370371 0.9120370370370371 0.9120370370370371 0.9627906976744186 0.9893023255813954 0.9488372093023256 0.916279069767441411]
% Logistic = [0.7685185185185185 0.8148148148148148 0.810851851851852 0.8055555555555556 0.7488372093023256 0.8893023255813954 0.8846511627906977 0.7953488372093023256]
% Prova 3
% Naive = [0.6997047407470471 0.7592592592592593 0.7129629629629629 0.7731481481481481 0.7228930232558184 0.7488372093023256 0.7348837209302326 0.6976744186046512]
% Logistic = [0.7314814814814815 0.7407407407407407 0.7268518518518519 0.7546296296296297 0.7534883720930232 0.7767441860465116 0.7534883720930232 0.725581395348837];
% SVM = [0.7888888888888888 0.8148148148148148 0.8055555555555556 0.8148148148148148 0.7674418604651163 0.8846511627906977 0.78064511627907 0.8 0.893823255813954 0.7813953488372093]
% Random = [0.824074074074074074 0.8425925925925926 0.8287037037037037 0.8657407407407407 0.827906976744186 0.8372093023255814 0.8465116279069768 0.8325581395348837 0.8418604651162791 0.8232558139534883];
% Tree = [0.8277777777777777 0.9212962962962963 0.8981481481481481 0.9259259259259259 0.909674418604651 0.9289302325581395 0.9255813953488372 0.9441860465116279 0.8837209302325582];
% KNN = [0.9708518518518519 0.9814814814814815 0.9861111111111112 0.9627906976744186 0.97674418604651163 0.9813953488372092 0.9767441860465116 0.9627906976744186]
%
% Prova 4
% Naive = [0.7037037037037037 0.7453703703703703 0.7175925925925926 0.7638888888888888 0.7627906976744186 0.772093023255814 0.7348837209302326 0.7209302325581395 0.7488372093023256 0.730232558139549];
% Logistic = [0.7037037037037037 0.7314814814814815 0.7037037037037037 0.7361111111111112 0.7441860465116279 0.739534883720930232 0.77674418604651163 0.7395348837209302];
% SVM = [0.7685185185185185 0.8148148148148148 0.7916666666666666 0.7962962962962963 0.7581395348837209 0.8325581395348837 0.7627906976744186 0.8046511627906976 0.8232558139534883 0.7906976744186046];
% Random = [0.8089259259259259 0.8333333333333334 0.8333333333333334 0.8379629629629629 0.8186046511627907 0.8697674418604651 0.846511627906976 0.846511627906976 0.827906976744186 0.827906976744186];
% Tree = [0.981481481481481 0.9305555555555556 0.9212962962962963 0.875 0.9023255813953488 0.9395348837209302 0.9823255813953488 0.8976744186046511];
% KNN = [0.9722222222222222 0.9768518518518519 0.9953703703703703 0.9907407407407407 0.9953703703703703 0.9627906976744186 0.9581395348837209 0.9627906976744186 0.9627906976744186];
%
% Prova 5
Naive = [0.7268518518518519 0.7129629629629629 0.7037037037037037 0.7917592592592592 0.7916666666666666 0.7534883720930232 0.71627906976744181];
Logistic = [0.7037037037037037 0.7361111111111112 0.7129629629629629 0.7546296296296297 0.7534883720930232 0.8 0.7441860465116279 0.7488372093023256];
SVM = [0.7638888888888888 0.7916666666666666 0.7870370370370371 0.772093023255814 0.8046511627906977 0.7813953488372093 0.8186046511627907 0.8046511627906977 0.7906976744186046];
Random = [0.8148148148148148 0.8333333333333334 0.8425925925925926 0.8796296296296297 0.8418604651162791 0.8418604651162791 0.8744186046511628 0.827906976744186 0.8418604651162791];
Tree = [0.981481481481481 0.9305555555555556 0.9212962962962963 0.875 0.9023255813953488 0.9395348837209302 0.9823255813953488 0.8976744186046511];
KNN = [0.9629629629629629 0.9953703703703703 0.9907407407407407 0.9953703703703703 0.9627906976744186 0.9581395348837209 0.9627906976744186 0.9627906976744186 0.9627906976744186];
```

Figura 4.4: Accuratezze con Data Augmentation.

Appare evidente che la **Data Augmentation** crei dei dataset che inducono il Classificatore kNN all'*overfitting*: Data Augmentation genera dei dati che 'assecondano' di più il Clasificatore kNN rispetto agli altri.

Inoltre inizialmente si è pensato che la tendenza all'*overfitting* potesse essere dovuta anche alla scelta dell'iperparametro $k=10$, ipotizzando che fosse un valore troppo piccolo e che dunque portasse il Classificatore ad essere particolarmente soggetto alla componente di rumore dei dati; tuttavia si è visto che anche modificando tale valore in un range [1-100] il risultato non cambia (100 come massimo poiché è il massimo numero inseribile come valore in Orange). Il modello sembra apprendere troppo dai dati di Training: potrebbe generalizzare poco e potrebbe non ottenere buoni risultati a livello di Test set.

4.2 Scelta del Training set e del Classificatore migliori

Date le considerazioni fatte nella sezione precedente, il Training set Aumentato sembra essere il peggioro tra i tre, tendendo a polarizzare i risultati dei Classificatori. La decisione si riduce al caso Originale o ribilanciato con SMOTE. Il **Training set ribilanciato con SMOTE** pare una buona via di mezzo tra l'Originale e l'Aumentato: presenta parità di dati appartenenti alle due classi ma non porta all'*overfitting* da parte dei Modelli e pertanto viene scelto come Training set definitivo.

Per quanto riguarda la scelta del Modello, si è visto dall'analisi *Multcompare* come il **Metodo Random Forest** [2] e **kNN** [4] ottengano buoni risultati a livello di Training SMOTE (e nel vaso Random Forest anche a livello di Training Originale). Pertanto si decide di testare questi due sul Test set.

4.3 Test & Score

I Modelli scelti per la classificazione vengono ora testati a livello di Test set utilizzando il Widget 'Test & Score' disponibile in Orange, selezionando l'opzione 'Test on Test Data'. I risultati ottenuti vengono riportati in Figura 4.5.

Model	AUC	CA	F1	Prec	Recall	MCC
Random Forest SMOTE	0.850	0.789	0.793	0.805	0.789	0.562
kNN	0.812	0.737	0.743	0.765	0.737	0.469

Figura 4.5: *Test & Score.*

Da notare anche le **Matrici di Confusione** 4.6.

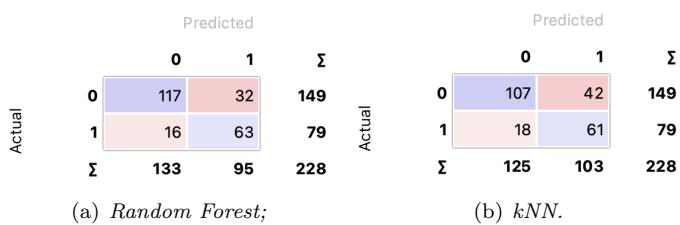


Figura 4.6: *Confusion Matrix*.

I Classificatori mostrano entrambi una buona accuratezza. Vengono riportate anche le curve ROC in Figura 4.7: in verde quella del Modello Random Forest, in arancione del kNN. Come si può notare il Classificatore Random Forest ottiene risultati leggermente migliori a livello di **Area Sotto la Curva**.

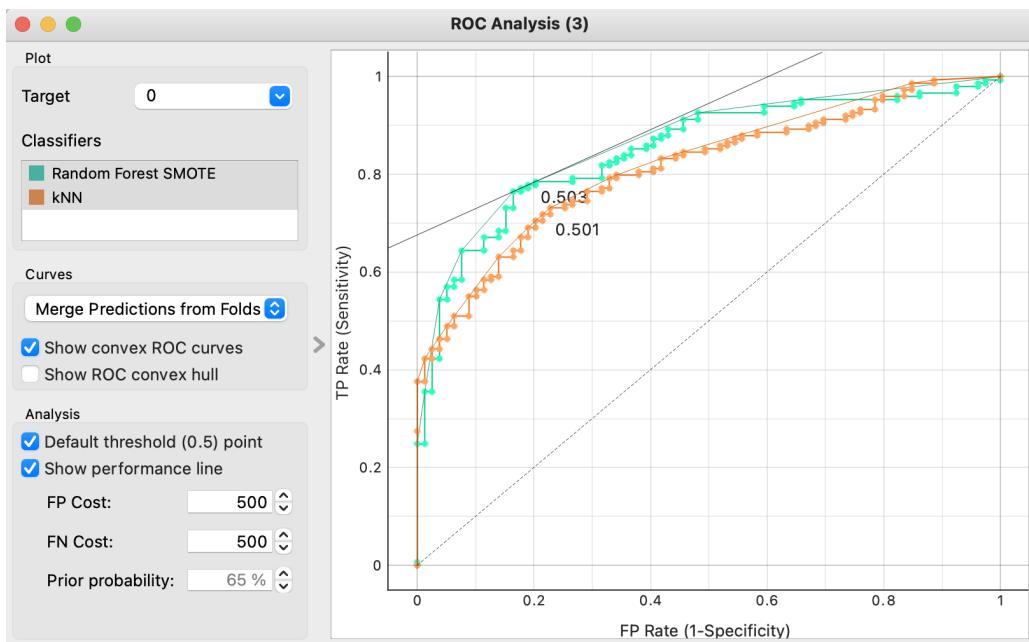


Figura 4.7: Curve ROC

A fronte dei risultati ottenuti (soprattutto se si considera AUC e MCC) il miglior Classificatore sembra essere il Random Forrest. Il risultato non ci stupisce:

- Si tratta di un Modello che funziona bene su set di dati con **tanti attributi**: nel nostro caso si hanno 7 Features che non sono tantissime ma nemmeno poche;
- Le performance del Classificatore sono buone per **variabili continue**: nel Pima Indians Diabetes Dataset vi sono 5 variabili continue e le altre due divengono tali a seguito del processo di Normalizzazione (la sesta è Insuline che seppur continua non viene considerata per *Feature Elimination*).
- Il Metodo classifica particolarmente bene **variabili** di *Outcome binarie*, come nel nostro caso.

4.3.1 Paragone tra tutti i Classificatori: una mia curiosità

Il modello finale Random Forest è stato scelto e si sono viste le sue prestazioni che appaiono soddisfacenti, la curiosità tuttavia ci porta a voler comunque fare un paragone sul Test set con gli altri modelli analizzati in precedenza. Le prestazioni sarebbero state migliori se avessi scelto un altro dei Classificatori?

Per visualizzare meglio analogie e differenze, i risultati di *Test & Score* vengono visualizzati attraverso 'bar plot' (Matlab, Fig.4.8) (I nomi vengono riportati abbreviati per migliore comprensione). Tutti i Classificatori sono stati allenati su Training set SMOTE come scelto nel paragrafo precedente.

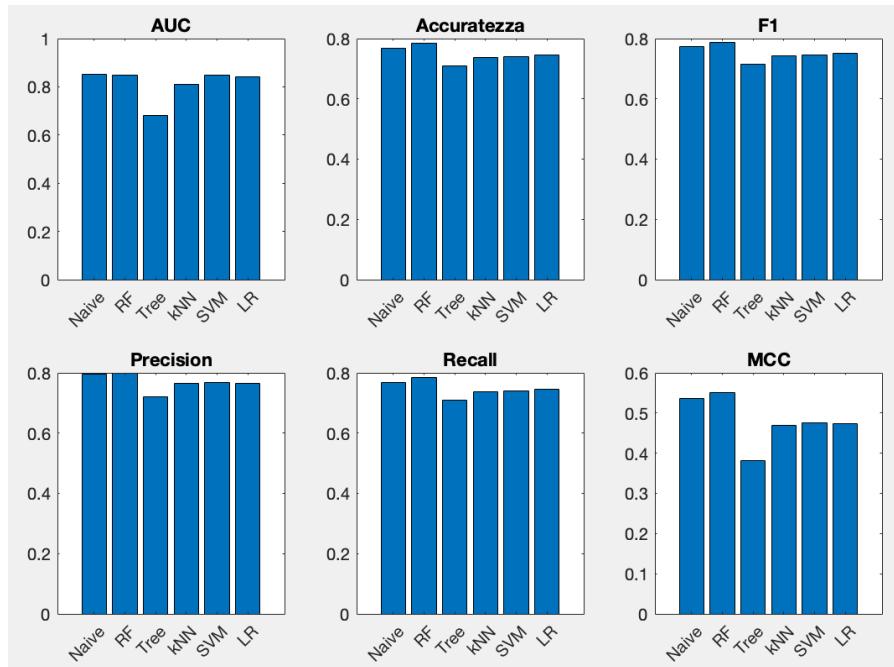


Figura 4.8: Bar Plot

Il Modello Random Forest avrebbe classificato bene anche in paragone con gli altri modelli. Il Classificatore ad Albero non ottiene risultati brillanti, cosa che già era intuibile dall'analisi sul Training set. Risulta dunque essere confermato il fatto che gli Alberi all'interno della Foresta ottengano risultati decisamente migliori di quelli singoli.

Stupiscono invece i Classificatori Naïve Bayes e Logistic Regression che non performavano particolarmente bene in precedenza, ma anche kNN che 'perde terreno' e non sembra affatto dare i risultati eclatanti che si ottenevano invece sul Training set. Vengono graficate anche le diverse Curve ROC (Fig.4.9).

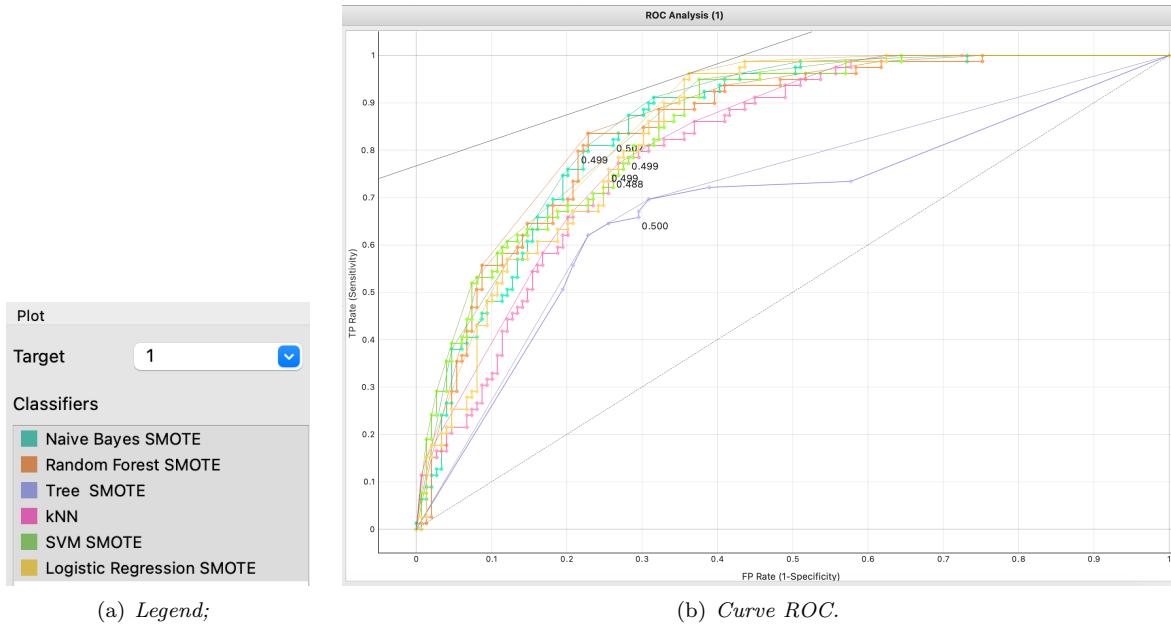


Figura 4.9: Paragone tra Classificatori.

Capitolo 5

Utilizzo e disseminazione del modello

Utente: Medico e non Paziente



Il modello ottenuto potrebbe essere implementato all'interno di una App. Tuttavia si pensa che l'utente al quale tale applicazione debba essere destinata sia un medico e non un paziente. Il modello può essere utile come supporto al medico per la diagnosi: considerando infatti l'accuratezza non elevata che presenta il nostro classificatore, si crede che sia comunque necessario il parere esperto di un professionista. Sarebbe rischioso invece mettere questo strumento nelle mani del paziente: sarebbe come se il paziente 'cercasse i sintomi online', distorcendo quindi la classificazione con un'opinione a priori non competente e ricca di componente emotiva.

Estensione al Sesso Maschile

Il *dataset* analizzato riporta esempi solamente di pazienti di sesso femminile, ci si chiede dunque: il modello ottenuto sarebbe estendibile anche al caso di pazienti maschi? (previa eliminazione della Feature 'Pregnancies'). Sarebbe interessante raccogliere dati di pazienti di sesso maschile e testare le prestazioni del modello pure su di loro.

Gestione della Storia Familiare

Come sottolineato nel capitolo iniziale, la 'Diabetes Pedigree Function' fornisce alcuni dati sulla storia del diabete mellito nei parenti e sulla relazione genetica di tali parenti con il paziente. Questa misura dell'influenza genetica dà una stima del rischio ereditario che si potrebbe avere con l'insorgenza del diabete mellito. Tuttavia non è detto che il dato sia conosciuto per tutti i pazienti, ci si chiede dunque se i risultati del Modello Random Forest cambierebbero o se rimarrebbero ugualmente predittivi se si eliminasse anche la Feature 'DPF'. Quello appena descritto potrebbe essere uno spunto di analisi per il futuro.

5.1 Pipeline Complessiva - Orange

Vengono lasciati tutti i *Widget* utilizzati, anche quelli non riportati all'interno del discorso, ma che in qualche modo sono stati utili per l'analisi del *dataset* PIDD.

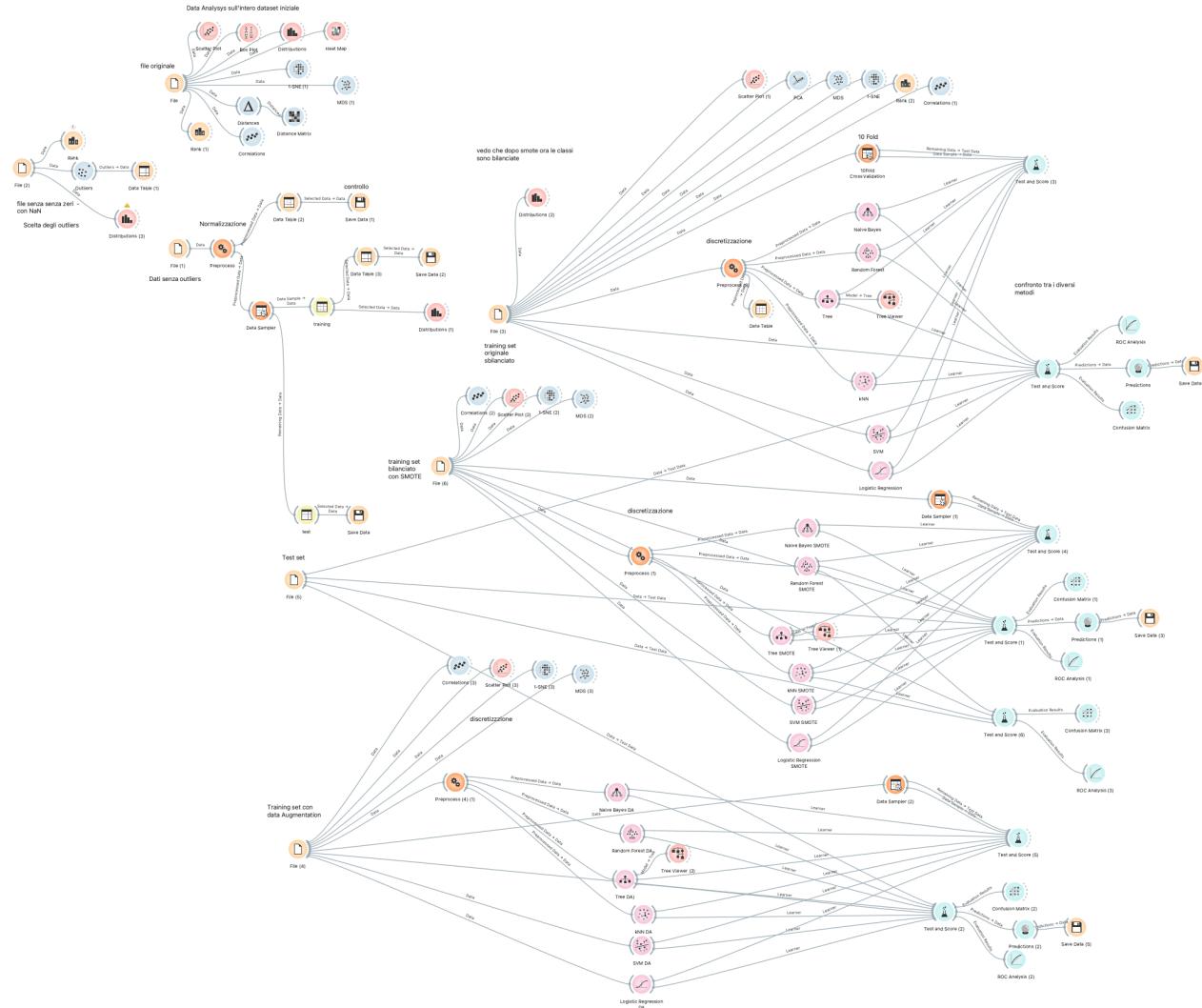


Figura 5.1: Pipeline Complessiva - Orange

Bibliografia

- [1] M. S. A. B. Shamreen Ahamed and A. O. V. Nancy, "Diabetes mellitus disease prediction using machine learning classifiers with oversampling and feature augmentation," *Hindawi - Advances in Human-Computer Interaction*, vol. Vol. 2022, p. 14 pages, 19 September 2022.
- [2] S. L. Priyanka Rajendra, "Prediction of diabetes using logistic regression and ensemble techniques," *Computer Methods and Programs in Biomedicine*, p. 1–8, 2021.
- [3] Y. H. Yang Guo, Guohua Bai, "Using bayes network for prediction of type-2 diabetes," *7th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2012.
- [4] L. O. H. W. P. K. Nitesh V. Chawla, Kevin W. Bowyer, "Smote: Synthetic minority over-sampling technique," *ournal of Artificial Intelligence Research*, vol. Vol. 16, p. 321–357, September 2002.
- [5] G. P. S. K. M. S. R. Y. Muhammad Exell Febrian, Fransiskus Xaverius Ferdinand, "Diabetes prediction using supervised machine learning," *Procedia Computer Science*, vol. Vol. 216, p. 21–30, 2023.
- [6] Y. L. D. Y. Y. J. Quan Zou, Kaiyang Qu and H. Tang, "Predicting diabetes mellitus with machine learning techniques," *Frontiers in Genetics*, vol. Vol. 9 n°515, p. 1–10, 06 November 2018).
- [7] H. S. Morteza Mohammad Shabtari, Vinodkumar Shukla and I. Nanda, "Analyzing pima indian diabetes dataset through data mining tool 'rapidminer,'" *International Conference on Advance Computing and Innovative Tecnologies in Engineering (ICACITE)*, p. 560–574, 2021).
- [8] J. Sun, "The study of pima indian diabetes," *ResearchGate*, 04 November 2016.
- [9] R. Patra and B. Khuntia, "Analysis and prediction of pima indian diabetes dataset using sdknn classifier technique," *IOP Conf. Series: Materials Science and Engineering*, p. 1–14, 2021.
- [10] H. N. . S. Ahuja, "Deep learning approach for diabetes prediction using pima indian dataset," *Journal of Diabetes Metabolic Disorders*, vol. Vol. 19, p. 391–403, 14 April 2020.
- [11] Q. A. X. Victor Chang, Jozeene Bailey and Z. Sun, "Pima indians diabetes mellitus classification based on machine learning (ml) algorithms," *ResearchGate- Neural Computing and Applications*, March 2022.
- [12] M.-L. L. Y.-P. F. D.-C. M. Jiang WU, Yuan-Bo DIAO, "A semi-supervised learning based method: Laplacian support vector machine used in diabetes disease diagnosis," *Interdiscip Sci Comput Life Sci*, vol. Vol. 1, p. 151–155, 7 January 2009.
- [13] M. A.-M. H. S. S. M. M. A. A. E.-B. . J. S. S. Md. Maniruzzaman, Md. Jahanur Rahman, "Accurate diabetes risk stratification using machine learning: Role of missing value and outliers," *Journal of Medical Systems*, vol. Vol. 42 n°92, p. 1–17, 10 April 2018.
- [14] M. M. A.-M. S. I. H. S. S. A. S. E.-B. J. S. S. Md. Maniruzzaman, Nishith Kumar, "Comparative approaches for classification of diabetes mellitus data: Machine learning paradigm," *Computer Methods and Programs in Biomedicine*, vol. Vol. 152, p. 23–34, 2017.
- [15] S. S. B. J. B. Bassam Abdo Al-Hameli, AbdulRahman A. Alsewari and M. A. H. Ali, "Diabetes disease prediction system using hnb classifier based on discretization method," *Journal of Integrative Bioinformatics*, vol. Vol. 20 n°0037, p. 1–13, 2023.

- [16] M. S. Chollette C. Olisah, Lyndon Smith, "Diabetes mellitus prediction and diagnosis from a data pre-processing and machine learning perspective," *Computer Methods and Programs in Biomedicine*, vol. Vol. 220, p. 1–12, 2022.
- [17] A. S. Francesco Mercaldo, Vittoria Nardone, "Diabetes mellitus affected patients classification and diagnosis through machine learning techniques," *Procedia Computer Science*, vol. Vol. 112, p. 2519–2528, 2017.
- [18] G. P. S. K. M. S. R. Y. Muhammad Exell Febrian, Fransiskus Xaverius Ferdinand, "Diabetes prediction using supervised machine learning," *Procedia Computer Science*, vol. Vol. 210, p. 21–30, 2023.
- [19] M. A. F. H. H. A. B. Umair Muneer Butt, Sukumar Letchmunan and H. H. R. Sherazi, "Machine learning based diabetes classification and prediction for healthcare applications," *Hindawi - Journal of Healthcare Engineering*, vol. Vol. 2021, p. 1–17, 2021.
- [20] M. A. A. M. R. I. S. M. S. I.-J. M. W. Q. Koushik Chandra Howlader, Md. Shahriare Satu and M. A. Moni, "Machine learning models for classification and identification of significant attributes to detect type 2 diabetes," *Health Information Science and Systems*, vol. Vol. 10 n°2, p. 1–13, 2022.
- [21] N. G. Subhash Chandra Gupta, "Predictive modeling and analytics for diabetes using hyperparameter tuned machine learning techniques," *Procedia Computer Science*, vol. Vol. 218, p. 1257–1269, 2023.