ENI Data Challange

Andrea Maioli

September 26, 2022

Contents

1 Data challange

Sulla base dei dati presenti al seguente link, costruisci un modello per prevedere la classe relativa allo stato fetale (N=normal; S=suspect; P=pathologic) a partire dalle features estratte dalle cardiotocografie.

Risolvi il problema al meglio delle tue capacità utilizzando codice Python, costruendo la tua soluzione tramite Jupyter Notebook o Scripts.

Prepara una presentazione dei risultati ottenuti (si avranno a disposizione circa 15 minuti per presentarla) focalizzandoti su questi tre punti:

- Analisi della soluzione scelta con giustificazione tecnica dell'approccio seguito
- Analisi dei risultati raggiunti (performance, possibili utilizzi dell'algoritmo)
- Cosa hai imparato del problema a partire dal modello

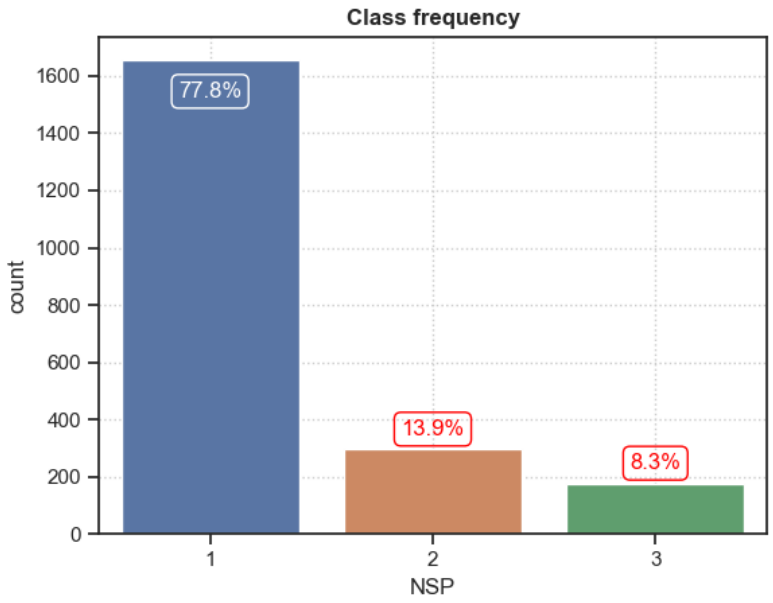
2 Analisi dati e *feature selection*

Durante tutto il corso del documento adotto la seguente convenzione per comodita':

$$N = 1$$
$$S = 2$$
$$P = 3$$

2.1 Classi

Prima ancora di analizzare i regressori osservo che la frequenza delle classi e' asimmetrica. In particolare i samples della classe **P** sono solo l' 8.3%.



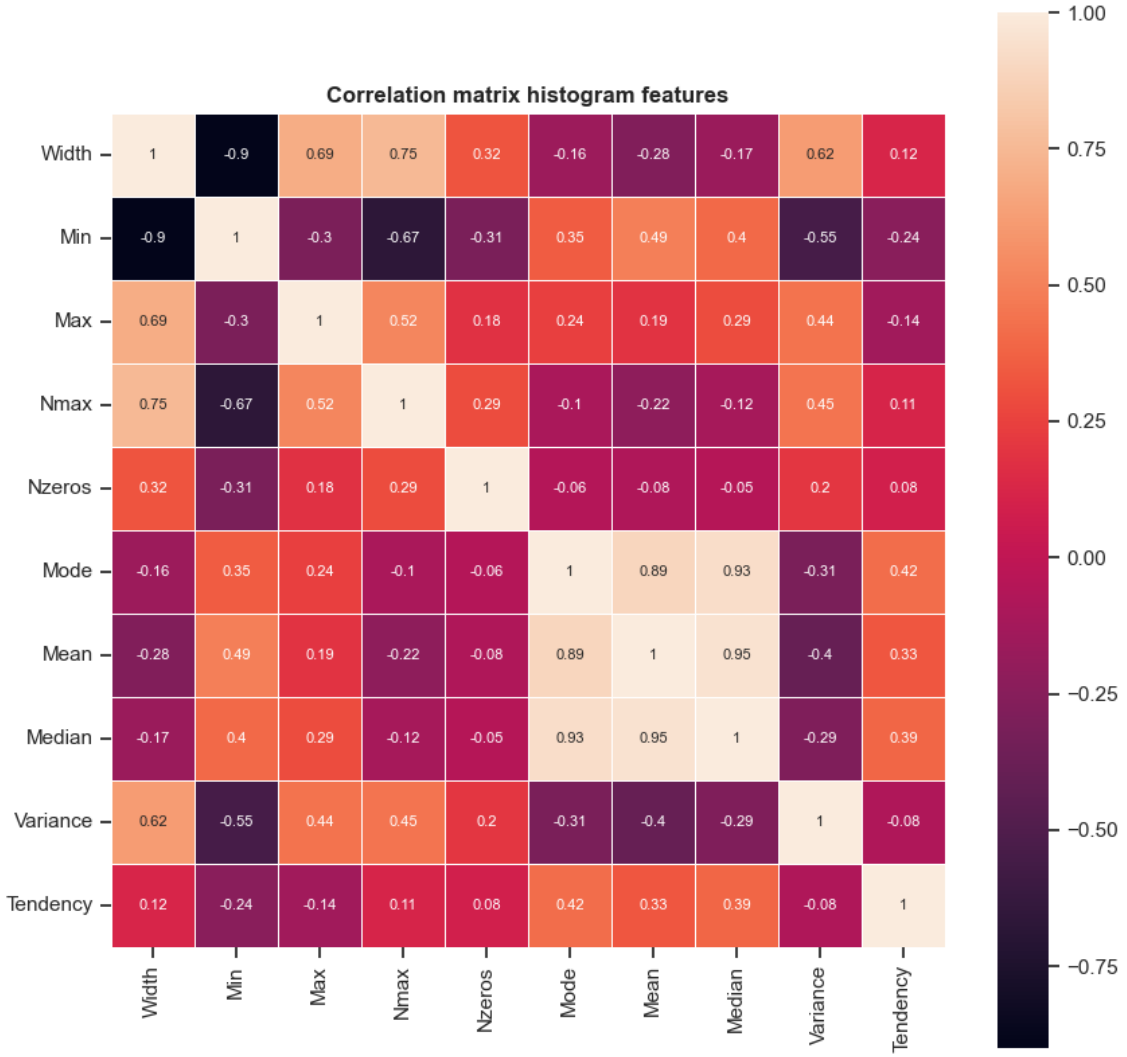
Segue che non avrà senso utilizzare l'**accuracy** come metrica in fase di valutazione e che, in fase di modellizzazione, sara' fondamentale usare tecniche per diminuire il rischio di overfitting.

2.2 Regressori

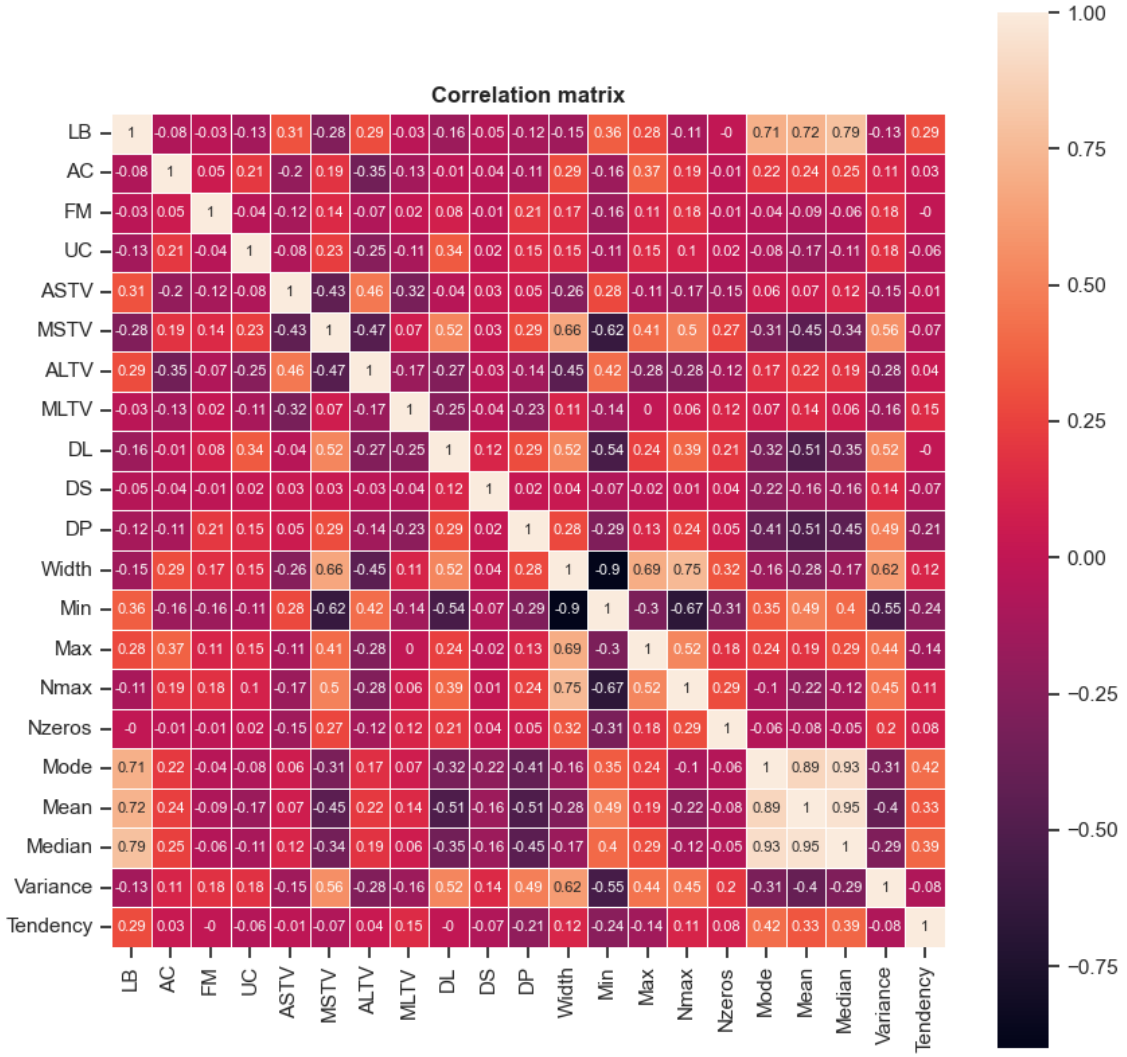
Noto che per definizione i regressori `Width, Min, Max, Nmax, Nzeros, Mode, Mean, Median, Variance, Tendency` sono tutte proprieta' dell'istogramma delle osservazioni eseguite durante l'analisi cardiografica. E' quindi probabile che ci siano informazioni ridondanti o variabili collineari e quindi che si possa diminuire la dimensione del problema.

La matrice di correlazione mi suggerisce che `Mode, Mean, Median` come prevedibile sono altamente correlati e quindi ha senso usare solo una di queste tre variabili.

Anche `Width` e `Min` hanno una correlazione molto alta senza pero' un motivo evidente. Per ora quindi non scartero' nessuno delle due variabili.

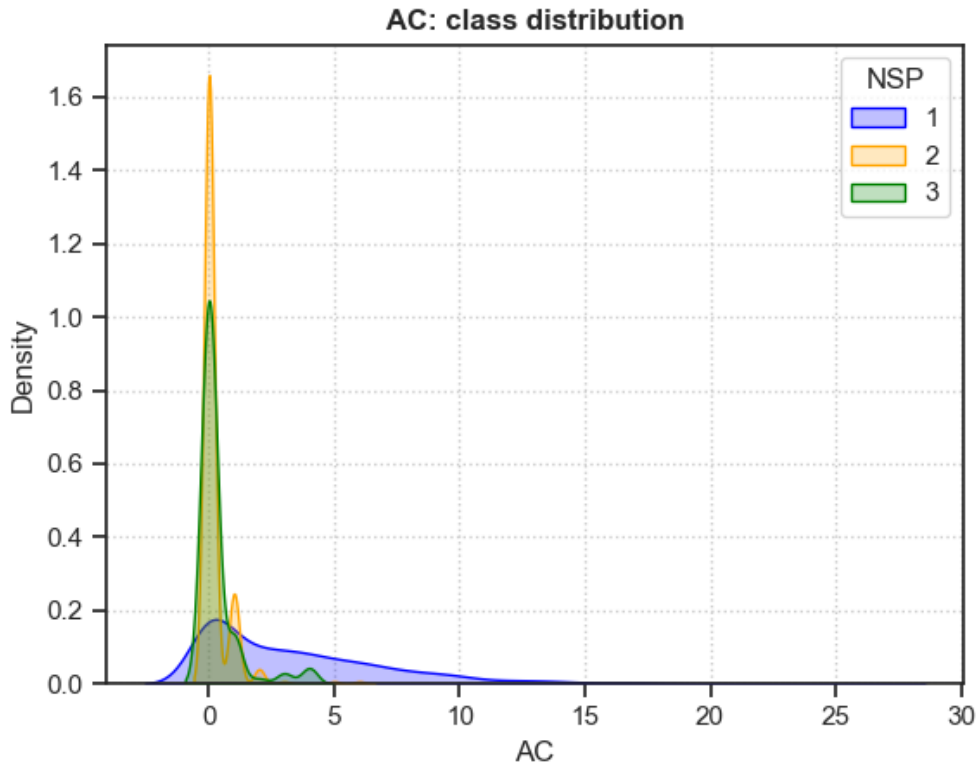


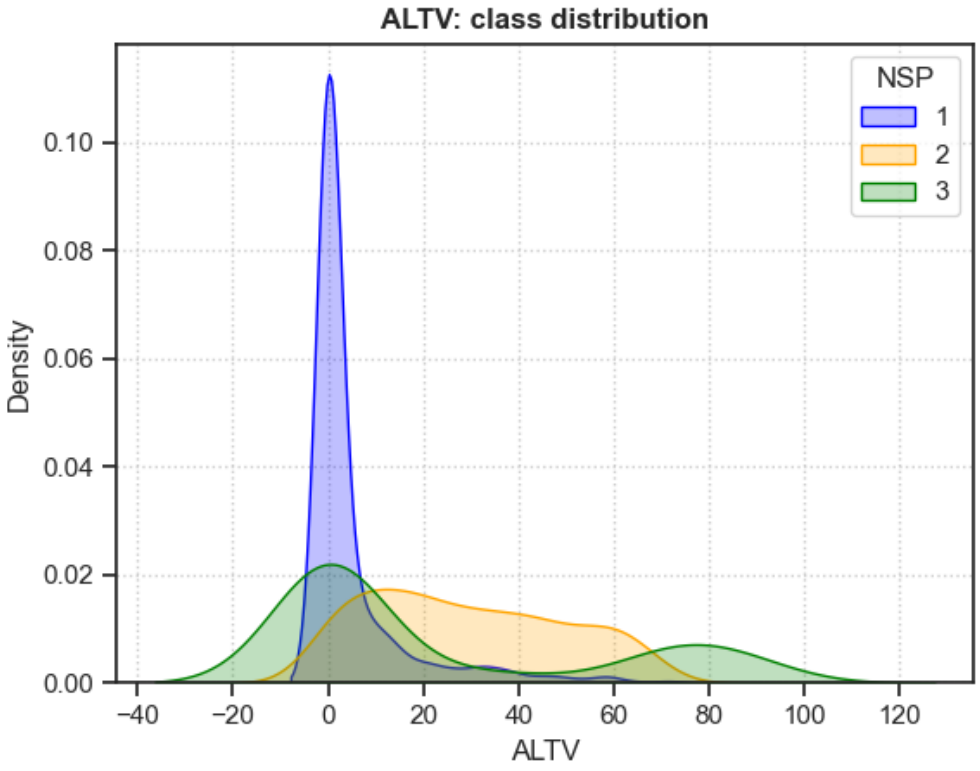
La matrice di correlazione globale non ci suggerisce altre semplificazioni ovvie.

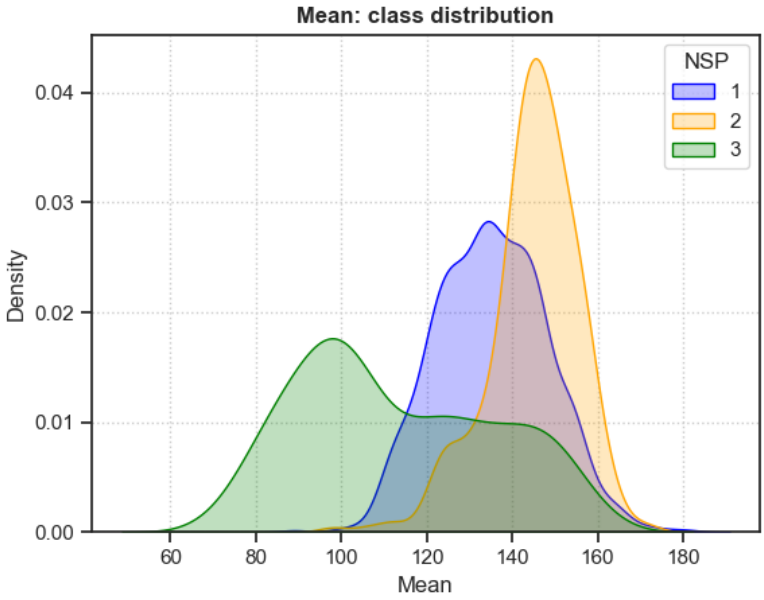


Visto il numero contenuto di variabili possiamo visualizzare come ogni feature si distribuisce rispetto alle classi.

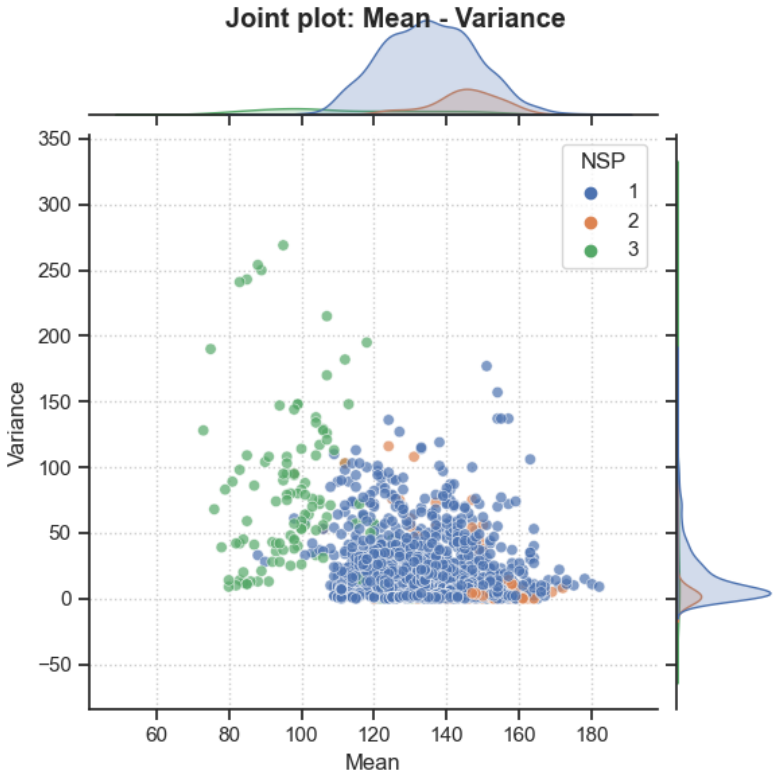
Feature con distribuzioni molto distanti tra loro forniranno molta informazione ai modelli e saranno i regressori principali.
Ho individuato `AC`, `ALTV`, `ASTV`, `Mean`, `DP` tra le migliori con questa proprieta'.


AC: class distribution
NSP
1
2
3
Density
AC
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
5
10
15
20
25
30


ALTV: class distribution
NSP
1
2
3
Density
ALTV
0.00
0.02
0.04
0.06
0.08
0.10
−40
−20
0
20
40
60
80
100
120



Replico questo approccio in due dimensioni e osservo che la distribuzione congiunta di alcune coppie di variabili e' estramamente esplicativa. Tra tutte individuo la coppia `(Mean, Variance)` che e' in grado di separare in modo soddisfacente buona parte dei samples della classe **P**.



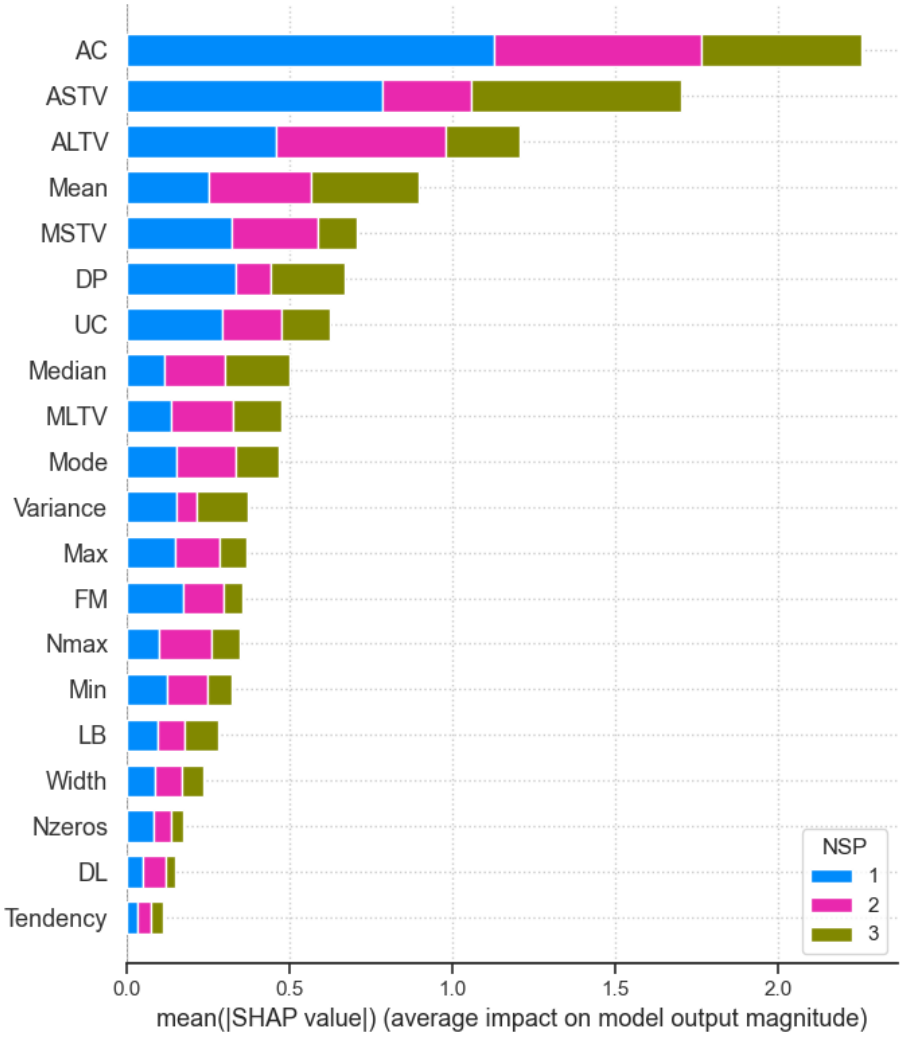
In conclusione mi aspetto che le variabili `AC, ALTV, ASTV, Mean, Variance, DP` siano tra i regressori principali.

2.3 Feature importance: valore di Shapley

Per avere anche un riferimento numerico riguardo alla *feature importance* delle singole variabili ho usato il **valore di Shapley**.

E' stato calcolato eseguendo il training di un `BoostedTreeClassifier` (il validation set e' stato usato per impostare un `early stopping` e evitare overfitting) a cui poi e' stato applicato il `TreeExplainer`

Il risultato e' il seguente:



Vista l'analisi grafica e i valori di Shapley seleziono come regressori: **AC, ASTV, ALTV, DP, MSTV, Mean, Variance**.

3 Modellizzazione

3.1 Benchmark

Per avere un punto di riferimento ho scritto due semplici modelli: `MostFreq` e `SmartRandom`.

- `MostFreq`: predice sempre la classe piu' frequente del training set.
- `SmartRandom`: assegna a ogni classe una probabilita' proporzionale alla frequenza della classe nel training set. Predice estraendo una classe random. L'estrazione viene fatta con le probabilita assegnate in fase di training.

3.2 Bilanciamento delle classi

Per la natura del problema la classe **P** e' la piu importante ed e' fondamentale cercare di diminuire i Falsi Negativi senza aumentare a dismisura i Falsi Positivi.

Per sopperire alla scarsita' delle classi **S** e **P** ho utilizzato due approcci:

- **Sample weight**: eseguire il training assegnando un peso diverso ai samples che compensi l'asimettria delle classi nella funzione di costo.
- **SMOTE**: utilizzare un algoritmo di oversampling per generare dati sintetici e ricondurmi a un problema di classificazione con classi bilanciate.

3.2.1 Sample weight

Voglio assegnare un peso ad ognuna delle classi. Sia k_i il peso assegnato alla classe i.

Devono essere pesi che soddisfino le seguenti proprieta':

- Il peso deve essere antiproporzionale alla frequenza della classe nel training set.
- $\sum_i k_i = 1$ per una piu' facile interpretazione.

Definisco allora i pesi k_i nel seguente modo; sia c_i = |samples della classe i|.

Allora:

$$K = \frac{1}{c_1} + \frac{1}{c_2} + \frac{1}{c_3}$$

$$k_i = \frac{\frac{1}{c_1}}{K} = \frac{K}{c_1}$$

Ottengo i seguenti pesi:

$$k_1 = 0.06 \pm 0.002$$
$$k_2 = 0.35 \pm 0.0016$$
$$k_3 = 0.59 \pm 0.0013$$

3.2.2 SMOTE

Invece di modificare i pesi dei sample e intervenire sulla funzione di costo posso fare oversampling dei samples, cioe' generare dati sintetici che rispettino la struttura e le relazioni tra regressori.

L'ideale e' generare un numero di dati tale da eliminare l'asimettria e ottenere quindi un training set con la stessa frequenza per ogni classe.

Vista la semplice struttra dei dati ho utilizzato un algoritmo SMOTE. Viene fatto oversampling di una classe tramite estrazione casuale di sample nell'inviluppo convesso dei samples appartenenti alla classe di interesse.

In fase di modellizzazione provero' diversi modelli con e senza **Sample weight** e **SMOTE** per verificare l'effettiva utilita'.

3.3 Semplice → Complesso

A livello modelisstico, oltre ai benchmark, ho deciso di utilizzare anche i modelli:

- LogisticRegression
- BoostedTreeClassifier (CatBoostClassifier)

In particolare sono interessato a stimare la differenza, in termini di performance, tra un modello lineare e un modello non lineare.

Ho scelto un modello `BoostedTree` poiche' i regressori sono vincolati da limiti fisici. Non e' quindi necessario un modello che sia in grado estrapolare valori estremi.

Questa e' la lista dei modelli utilizzati che competeranno tra di loro:

- `MostFrequent`[1]
- `SmartRandom` [1]
- `Logistic`
- `LogisticWeight`
- `CatBoost`
- `CatBoostWeight`
- `CatBoostSMOTE`

3.4 Note tecniche

Di seguito alcune note tecniche riguardo i modelli:

1. Nel caso dei modelli `Logistic` viene applicato anche una *penalty term* $L1$, per questo viene applicato uno `StandardScaler` ai regressori.

[1] Questo modello ha in realta' solo lo scopo di stabilire un benchmark.

2. Nel caso del modello `SMOTE`, come suggerito nell' articolo originale, viene applicato anche un undersampler (tramite `RandomUnderSample`) della classe piu frequente **N**.
3. Nel caso dei modelli `BoostedTreeClassifier` e' stato impostato un *early stopping* con `rounds=50` per evitare *overfitting*.

4 Performance

4.1 Scelta della metrica

Per i problemi di classificazione sono disponibili decine di metriche. E' quindi fondamentale stabilire che tipo di risultato si vuole ottenere.

Nel caso di questo dataset voglio una metrica prediliga il modello che soddisfa le seguenti proprieta':

- Dara la priorita' alla massimizzazione della **Recall** della classe **P**
- Minimizzare **Falsi Negativi**, sopratutto nel caso di classi **P** erroneamente predette come **N** (e' infatti molto piu grave classificare un caso **Patologico** come **Normale** che viceversa).
- Mantere **Precision** ragionevolmente alte per le classi **N** e **S** (voglio cioe' evitare troppi **Falsi Positivi** della classe **P**).

Per ottenere queste proprieta' ho deciso di stabilire il modello migliore tramite il seguente criterio:

1. Selezionare i 3 modelli con **F2 score** medio maggiore.[a]
2. Tra i 3 modelli selezionati scegliere quello con **Recall** della classe **P** maggiore.

[a] $F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$

4.2 Risultati

I risultati che seguono sono stati ottenuti tramite `StratifiedCrossValidation` con `cv=5`.

Model	F2 Score	Recall class P
MostFrequent	0.315	0
SmartRandom	0.322	0.094
Logistic	0.712	0.678
LogisticWeight	0.755	0.785
CatBoost	**0.801**	0.795
CatBoostWeight	**0.797**	0.812
CatBoostSMOTE	**0.797**	**0.818**

Utilizzando il criterio precedentemente descritto, `CatBoostSMOTE` e' il modello che meglio soddisfa le proprieta' richieste. Poiche' pero' `CatBoostWeight` ha performance analoghe ma senza la componente randomica dello `SMOTE` (e quindi maggiore stabilita') **individuo CatBoostWeight come modello di riferimento**.

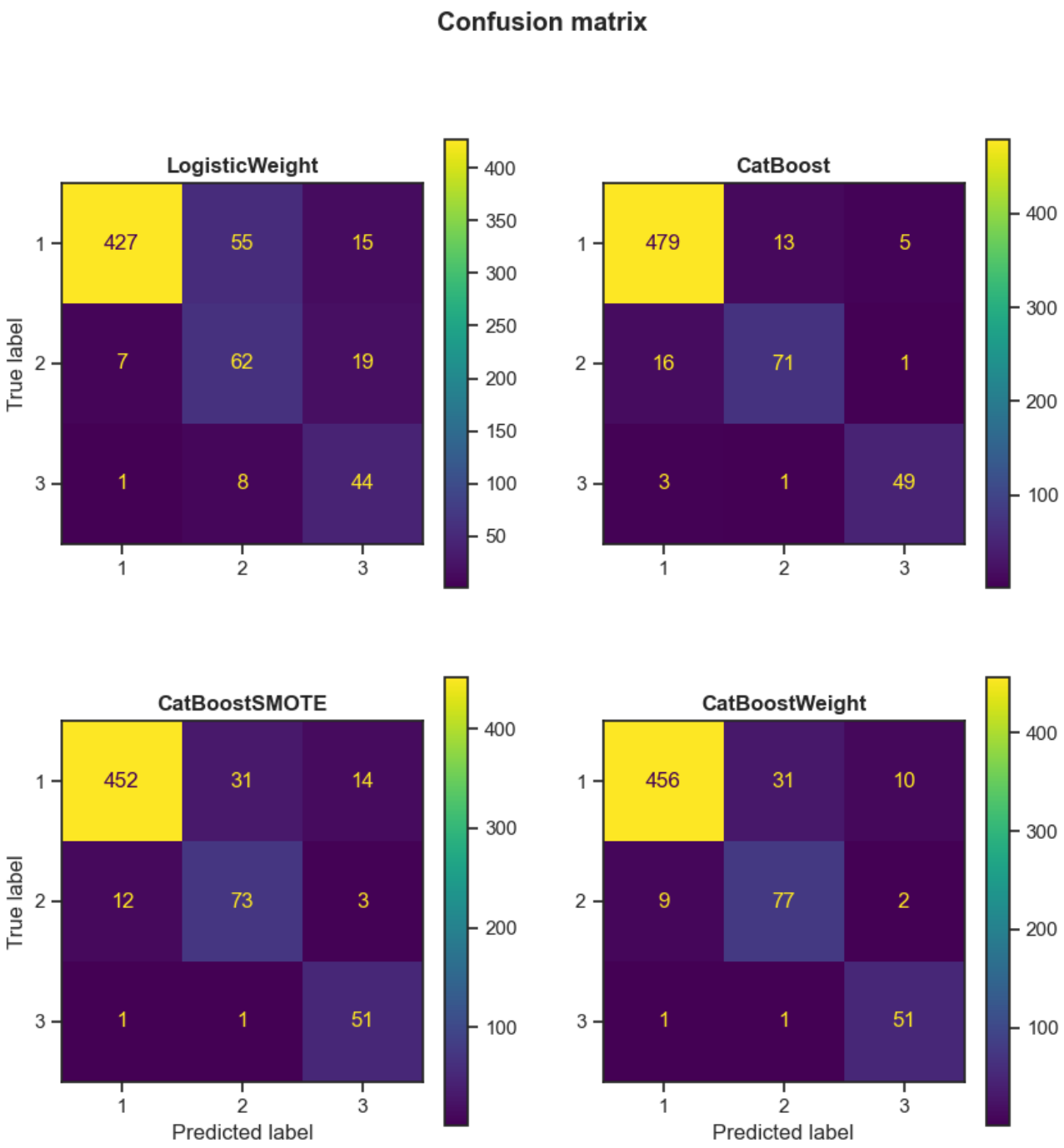


Figure 1: Confusion matrix del principali modelli calcolata su una singola run

5 Conclusione

Il problema di classificazione che ho analizzato e' sicuramente un problema particolare.

Sebbene infatti possa essere considerato standard per quanto riguarda la modelizzazione dei classificatori, e' stato molto costruttivo in particolare per:

- Il processo di **feature selection** in cui abbiamo verificato che un approccio qualitativo (visuale) ha portato agli stessi risultati di un approccio numerico (valori di Shapley). Questo e' stato possibile anche alla facile interpretabilita' delle variabile.
- La costruzione di un criterio di valutazione non standard. Questo e' stato il punto fondamentale poiche' ha costretto a porsi domande sulla natura del problema e ad assegnare un significato preciso all' output.

In generale il modello finale e' stato pensato per funzionare come un filtro e segnalare quei casi che richiedono una verifica *"manuale"* da parte di medici specializzati.