

# Modelli di Ottimizzazione per Big Data

## Classificazione multi-classe

**Alfredo Milani - 0255826**  
**Mauro Trulli - 0277665**

A.A. 2019/2020



# Contents

<b>1 Problem definition</b>	<b>4</b>
1.1 Obiettivo . . . . .	4
<b>2 Computational model</b>	<b>5</b>
2.1 Data preparation . . . . .	5
2.1.1 Data splitting . . . . .	5
2.1.2 Bad values management . . . . .	5
2.1.3 Data normalization . . . . .	6
2.1.4 Feature selection . . . . .	11
2.1.5 Data sampling . . . . .	13
2.2 Hyper-parameters tuning . . . . .	13
2.3 Evaluation . . . . .	15
<b>3 Output analysis</b>	<b>16</b>
<b>4 Conclusions</b>	<b>17</b>



## List of Figures

1	Pair plot representing <i>features</i> .	12
---	--	----



## List of Tables

1	<i>F1-score</i> con sampler SMOTE e scaler MinMaxScaler . . . . .	7
2	<i>F1-score</i> con sampler SMOTE e scaler MaxAbsScaler . . . . .	8
3	<i>F1-score</i> con sampler SMOTE e scaler QuantileTransformer . . . . .	9
4	<i>F1-score</i> con sampler SMOTE e scaler PowerTransformer (Yeo-Johnson)	10
5	<i>F1-score</i> con sampler SMOTE e scaler PowerTransformer (Box-Cox) . .	11
6	Lista degli iperparametri testati per ogni modello di <i>machine learning</i> testati	14
7	<i>F1-score</i> per i vari modelli di <i>machine learning</i> analizzati . . . . .	16



# 1 Problem definition

## 1.1 Obiettivo

L'obiettivo del progetto è risolvere un problema di classificazione multi-classe con 4 classi utilizzando algoritmi di machine learning. Il classificatore deve prevedere la classe corretta sulla base delle 20 caratteristiche di ciascuna istanza.

Il dataset considerato, *training\_set.csv*, consta di 21 attributi: i primi 20 (F1-F20) rappresentano le features mentre l'ultimo (CLASS) rappresenta la classe. Il dataset completo presenta 10000 istanze.

I modelli di *machine learning* analizzati sono stati valutati utilizzando *f1-score*.

## 2 Computational model

Il modello migliore è stato ottenuto dopo una prima fase di *data preparation*, ed una successiva fase di *hyper-parameter tuning*.

### 2.1 Data preparation

#### 2.1.1 Data splitting

Il *dataset* è stato diviso in due sezioni differenti:

- *training set*: contiene 80% dell'intero dataset originale (6400 records)
- *test set*: contiene il restante 20% del dataset originale (1600 records)

#### 2.1.2 Bad values management

Il *dataset* originale (*training\_set.csv*) contiene dati mancanti in corrispondenza di alcune delle *features*, per questo motivo, i valori relativi a tali campi sono stati sostituiti con la mediana relativa alla colonna della *feature* corrispondente all'interno del *dataset*.

In seguito, sono stati analizzati gli *outliers*. Un *outlier* è un'osservazione che devia marcatamente da altre osservazioni nel campione di dati. L'identificazione di potenziali *outliers* è importante dal momento che potrebbero indicare dati corrotti o mal codificati, anche se, in alcuni casi, potrebbero essere il risultato di variazioni casuali nel campione di dati.

L'individuazione degli *outliers* dipende dalla distribuzione dei dati sottostante. Il *dataset* considerato, come si evince dal pair-plot in figura 1, seguiva approssimativamente una distribuzione *Normale*, pertanto sono stati analizzati i risultati derivanti da tre metodi:

- *z-score*
- *modified z-score*
- *inter-quartile range*

Lo *Z-score* di una osservazione è definito come:

$$z_i = \frac{x_i - \mu_x}{\sigma_x}$$

essendo  $\mu_x$  e  $\sigma_x$ , rispettivamente, la media e la deviazione standard del campione di dati. Sono considerati *outliers* i campioni con il valore assoluto dello score maggiore di 3.

Lo *Z-score modificato* (*Iglewicz and Hoaglin*) è definito come:

$$m_i = \frac{0.6745 \cdot (x_i - \tilde{x})}{MAD}$$

essendo *MAD* la median absolute deviation e  $\tilde{x}$  la mediana. Sono considerati *outliers* i campioni con il valore assoluto dello score maggiore di 3.5.

L'*inter-quartile range* è definito come:

$$IQR = Q_3 - Q_1$$

essendo  $Q_1$  e  $Q_3$ , rispettivamente, il primo ed il terzo quartile del campione di dati. Sono considerati *outliers* i campioni con uno valore maggiore di  $Q_3 + IQR \cdot 1.5$  e minore di  $Q_1 - IQR \cdot 1.5$ .

Per non ridurre la numerosità del *training set* si è scelto di sostituire (e non eliminare) gli *outliers* con la mediana relativa ad una determinata *feature*. Si è ritenuto opportuno utilizzare la mediana in quanto è una misura più robusta per la rappresentazione della dispersione di valori rispetto alla media essendo meno soggetta alla presenza di *outliers*.

### 2.1.3 Data normalization

La normalizzazione è il processo di ridimensionamento dei singoli campioni per avere una norma unitaria. Questo processo può essere utile se si prevede di utilizzare una forma quadratica come il prodotto scalare o qualsiasi altro kernel per quantificare la somiglianza di qualsiasi coppia di campioni. Per effettuare la normalizzazione dei campioni sono stati utilizzati gli scalers: *MinMaxScaler*, *MaxAbsScaler*, *QuantileTransformer (uniform output)* il quale fornivano un output già normalizzato nell'intervallo [0, 1], mentre con gli scaler *PowerTransformer (Yeo-Johnson e Box-Cox transforms)* si è dovuto ulteriormente normalizzare l'output in modo "manuale" poichè l'output non era compreso nell'intervallo [0, 1].

La tabella 1 mostra i risultati ottenuti con MinMaxScaler

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (100, 50) learning_rate: adaptive learning_rate_init: 0.01 solver: sgd	<b>0.87906</b>
Support Vector Machine	C: 10 decision_function_shape: ovo gamma: 10 kernel: rbf	0.79652
Decision Tree	criterion: entropy max_depth: 90 max_features: None min_samples_leaf: 1 min_samples_split: 2 splitter: best	0.62407
Random Forest	criterion: entropy max_depth: 90 max_features: log2 min_samples_leaf: 2 min_samples_split: 2 n_estimators: 400	0.81243
K-Nearest Neighbors	metric: minkowski n_neighbors: 3 p: 4	0.75836
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 400	0.82467
Naive Bayes	priors: None var_smoothing: 10e-08	0.55340

Table 1: *F1-score* con sampler SMOTE e scaler MinMaxScaler

La tabella 2 mostra i risultati ottenuti con MaxAbsScaler

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (150, 100) learning_rate: adaptive learning_rate_init: 0.1 solver: sgd	<b>0.84048</b>
Support Vector Machine	C: 10 decision_function_shape: ovo gamma: 10 kernel: rbf	0.69239
Decision Tree	criterion: entropy max_depth: 90 max_features: None min_samples_leaf: 1 min_samples_split: 2	0.62502
Random Forest	criterion: entropy max_depth: 80 max_features: sqrt min_samples_leaf: 2 min_samples_split: 2 n_estimators: 400	0.81371
K-Nearest Neighbors	metric: minkowski n_neighbors: 3 p: 3	0.76017
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 300	0.81562
Naive Bayes	priors: None var_smoothing: 0.01	0.54984

Table 2: *F1-score* con sampler SMOTE e scaler MaxAbsScaler

La tabella 3 mostra i risultati ottenuti con QuantileTransformer

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (150, 100) learning_rate: adaptive learning_rate_init: 0.1 solver: sgd	<b>0.82885</b>
Support Vector Machine	C: 10 decision_function_shape: ovo gamma: 10 kernel: rbf	0.74399
Decision Tree	criterion: entropy max_depth: 90 max_features: None min_samples_leaf: 5 min_samples_split: 2	0.61674
Random Forest	criterion: entropy max_depth: 80 max_features: log2 min_samples_leaf: 2 min_samples_split: 2 n_estimators: 500	0.81326
K-Nearest Neighbors	metric: minkowski n_neighbors: 11 p: 3	0.80935
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 200	0.81836
Naive Bayes	priors: None var_smoothing: 0.01	0.49225

Table 3: *F1-score* con sampler SMOTE e scaler QuantileTransformer

La tabella 4 mostra i risultati ottenuti con PowerTransformer (Yeo-Johnson)

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (150, 100) learning_rate: adaptive learning_rate_init: 0.1 solver: sgd	<b>0.82103</b>
Support Vector Machine	C: 50 decision_function_shape: ovo gamma: 10 kernel: rbf	0.78082
Decision Tree	criterion: entropy max_depth: 80 max_features: None min_samples_leaf: 2 min_samples_split: 2	0.61677
Random Forest	criterion: entropy max_depth: 90 max_features: sqrt min_samples_leaf: 2 min_samples_split: 2 n_estimators: 200	0.81341
K-Nearest Neighbors	metric: minkowski n_neighbors: 3 p: 3	0.76799
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 200	0.82963
Naive Bayes	priors: None var_smoothing: 1e-08	0.55331

Table 4: *F1-score* con sampler SMOTE e scaler PowerTransformer (Yeo-Johnson)

La tabella 5 mostra i risultati ottenuti con PowerTransformer (Box-Box)

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (150, 100) learning_rate: adaptive learning_rate_init: 0.1 solver: sgd	<b>0.88298</b>
Support Vector Machine	C: 50 decision_function_shape: ovo gamma: 10 kernel: rbf	0.78082
Decision Tree	criterion: entropy max_depth: 80 max_features: None min_samples_leaf: 2 min_samples_split: 2	0.62280
Random Forest	criterion: entropy max_depth: 80 max_features: log2 min_samples_leaf: 2 min_samples_split: 2 n_estimators: 400	0.81267
K-Nearest Neighbors	metric: minkowski n_neighbors: 3 p: 3	0.76799
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 300	0.82458
Naive Bayes	priors: None var_smoothing: 1e-08	0.55331

Table 5: *F1-score* con sampler SMOTE e scaler PowerTransformer (Box-Cox)

#### 2.1.4 Feature selection

La feature selection è una tecnica in cui scegliamo quelle feature (caratteristiche) nei nostri dati che contribuiscono maggiormente alla variabile di destinazione. In altre parole, scegliamo i migliori predittori per la variabile target.

Vantaggi della feature selection:

- Riduce l'overfitting: meno dati ridondanti significa meno possibilità di prendere decisioni basate su dati / rumore ridondanti.

- Migliora la precisione: meno dati fuorvianti significa che la precisione della modellazione migliora.
- Riduce il tempo di addestramento: meno dati significa che gli algoritmi si addestrano più velocemente.

Come approccio alla feature selection è stata usata la *SelectKBest*, ovvero per "filtrare" le features viene assegnato un punteggio ad ogni feature utilizzando una funzione, infine vengono rimosse tutte feature tranne le k con il punteggio maggiore.

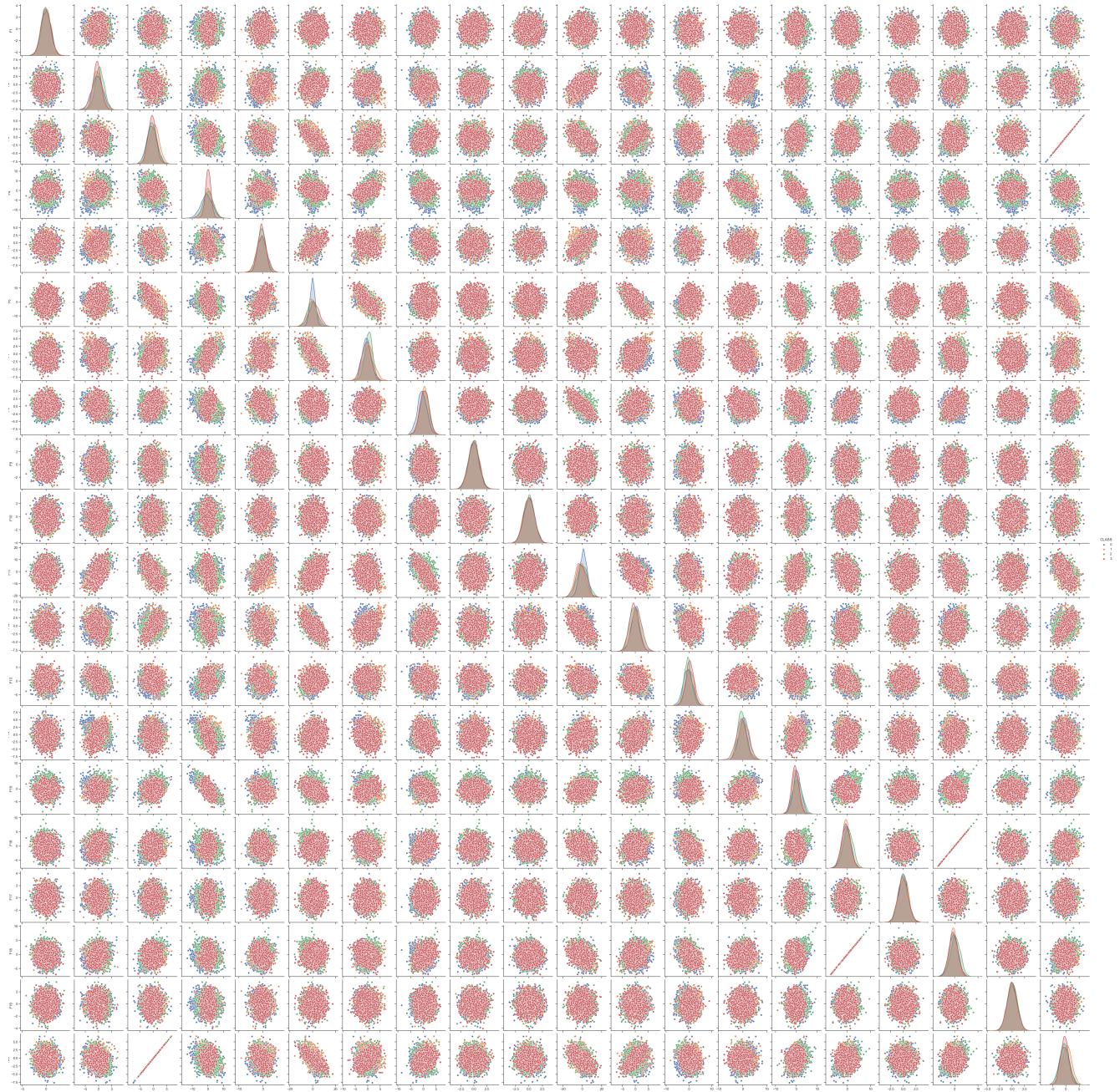


Figure 1: Pair plot representing *features*.

### 2.1.5 Data sampling

Il *dataset* considerato presenta elementi così distribuiti:

- *Class 1*: 33.67 %
- *Class 2*: 15.99 %
- *Class 3*: 20.66 %
- *Class 4*: 29.68 %

È stato quindi effettuato balancing delle classi per evitare che il modello finale pecchi nel riconoscimento di classi meno presenti nel *training set*.

La tecnica di balancing utilizzata è lo *SMOTE*.

**TODO**

## 2.2 Hyper-parameters tuning

I modelli utilizzati per la classificazione sono:

- *Multi-Layer Perceptron*
- *Support Vector Machine*
- *Decision Tree*
- *Random Forest*
- *K-Nearest Neighbors*
- *Stochastic Gradient Descent*
- *Ada Boost*
- *Naive Bayes*
- *K-Means*

In questa fase sono stati cercati gli *iper-parametri* migliori per i vari classificatori tra quelli elencati nella seguente tabella.

Model	Parameters
Multi-Layer Perceptron	activation: [tanh, relu] hidden_layer_sizes: [(150, 100), (120, 60), (60, 30), (75), (45)] learning_rate: [constant, adaptive] learning_rate_init: [1e-1, 1e-2, 1e-3, 1e-4] solver: [sgd, adam]
Support Vector Machine	kernel: linear C: [0.1, 1, 10] decision_function_shape: [ovo, ovr]
	kernel: rbf C: [0.1, 1, 10] decision_function_shape: [ovo, ovr] gamma: [1e-4, 1e-3, 1e-2, 1e-1, 1e+1, 1e+2, 1e+3, 1e+4]
Decision Tree	kernel: poly C: [0.1, 1, 10] degree: [2, 3, 4] decision_function_shape: [ovo, ovr] gamma: scale
Random Forest	criterion: [entropy, gini] max_depth: [80, 90] max_features: [log2, sqrt, None] min_samples_leaf: [2, 5, 10] splitter: [best, random]
K-Nearest Neighbors	criterion: [entropy, gini] max_depth: [80, 90] max_features: [log2, sqrt, None] min_samples_leaf: [2, 5, 10] n_estimators: [100, 200, 300, 400, 500]
Ada Boost	metric: [minkowski, euclidean, chebyshev] n_neighbors: [3, 5, 7, 11] p: [3, 4, 5]
Naive Bayes	criterion: [entropy, gini] class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: [100, 200, 300] splitter: [best, random]
K-Means	priors: [None, [0.25, 0.25, 0.25, 0.25]] var_smoothing: [10e-9, 10e-6, 10e-3, 10e-1]
	max_iter: 10000 n_clusters: 4

Table 6: Lista degli iperparametri testati per ogni modello di *machine learning* testati

### 2.3 Evaluation

La metrica utilizzata al fine della valutazione dei classificatori è la *f1-score* (media armonica), definita come segue:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

con,

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$

essendo  $TN$  il numero di veri negativi,  $FP$  il numero di falsi positivi,  $FN$ , falsi negativi e  $TP$  veri positivi.

### 3 Output analysis

Lo score migliore è stato ottenuto con samplig di tipo **SMOTE**

La tabella 7 mostra i risultati ottenuti.

Lo score migliore è stato ottenuto con samplig di tipo **SMOTE**

La tabella 7 mostra i risultati ottenuti.

Model	Parameters	F1-score
Multi-Layer Perceptron	activation: relu hidden_layer_sizes: (100, 50) learning_rate: adaptive learning_rate_init: 0.01 solver: sgd	<b>0.87906</b>
Support Vector Machine	C: 10 decision_function_shape: ovo gamma: 10 kernel: rbf	0.79652
Decision Tree	criterion: entropy max_depth: 90 max_features: None min_samples_leaf: 1 min_samples_split: 2 splitter: best	0.62407
Random Forest	criterion: entropy max_depth: 90 max_features: log2 min_samples_leaf: 2 min_samples_split: 2 n_estimators: 400	0.81243
K-Nearest Neighbors	metric: minkowski n_neighbors: 3 p: 4	0.75836
Ada Boost	class_weight: balanced max_depth: 90 max_features: 3 min_samples_leaf: 4 n_estimators: 400	0.82467
Naive Bayes	priors: None var_smoothing: 10e-08	0.55340

Table 7: *F1-score* per i vari modelli di *machine learning* analizzati

## 4 Conclusions

Dai vari test effettuati e dai risultati ottenuti si evince che il miglior modello è il Multi-Layer Perceptron con parametri:

Una possibile miglioria da applicare al modello potrebbe quessa d iaumentare il numero di percetroni in ogni livello, ma questo comporterebbe un significativo aumento della complessità computazionale portando conseguentemente ad un notevole aumento del tempo di esecuzione nel training del modello.