

Water Potability

Data Science Project

Edited by: Balestrieri Niccolò – 307905

SOMMARIO

-Introduzione al progetto.....	2
-Analisi dei dati iniziale(esplorazione e scelta).....	2
-Data visualization.....	4
-Gestione valori mancanti.....	6
-Feature selection.....	6
-Splitting dataset.....	7
-Feature scaling.....	7
-Selezione modello migliore(ml model comparison).....	8
-Fine tuning con il modello migliore.....	9
-Confusion matrix.....	10
-T-sne.....	11

INTRODUZIONE AL PROGETTO

In questa relazione, verrà mostrato un progetto che si occupa dell'analisi di un dataset che possiede informazioni riguardanti la potabilità dell'acqua.

In questo progetto, si sfrutta l'apprendimento supervisionato (supervised learning) con un canale di preferenza per il task di classificazione.

Il dataset che andremo ad utilizzare è leggermente sbilanciato, 61% non potabile (indicata con 0) e 39% potabile (indicata con 1), con questo tipo di informazioni è possibile andare a dire che si utilizzeranno precision, recall e f1 score per andare a valutare la correttezza che ci restituisce il nostro modello.

ANALISI DEI DATI INIZIALE (ESPLORAZIONE E SCELTA)

In questa fase è importante andare a studiare gli attributi del dataset e le sue caratteristiche.

Il dataset analizzato, possiede 10 features e 3276 esempi ed ha una singola variabile target.

Le feature sono le seguenti:

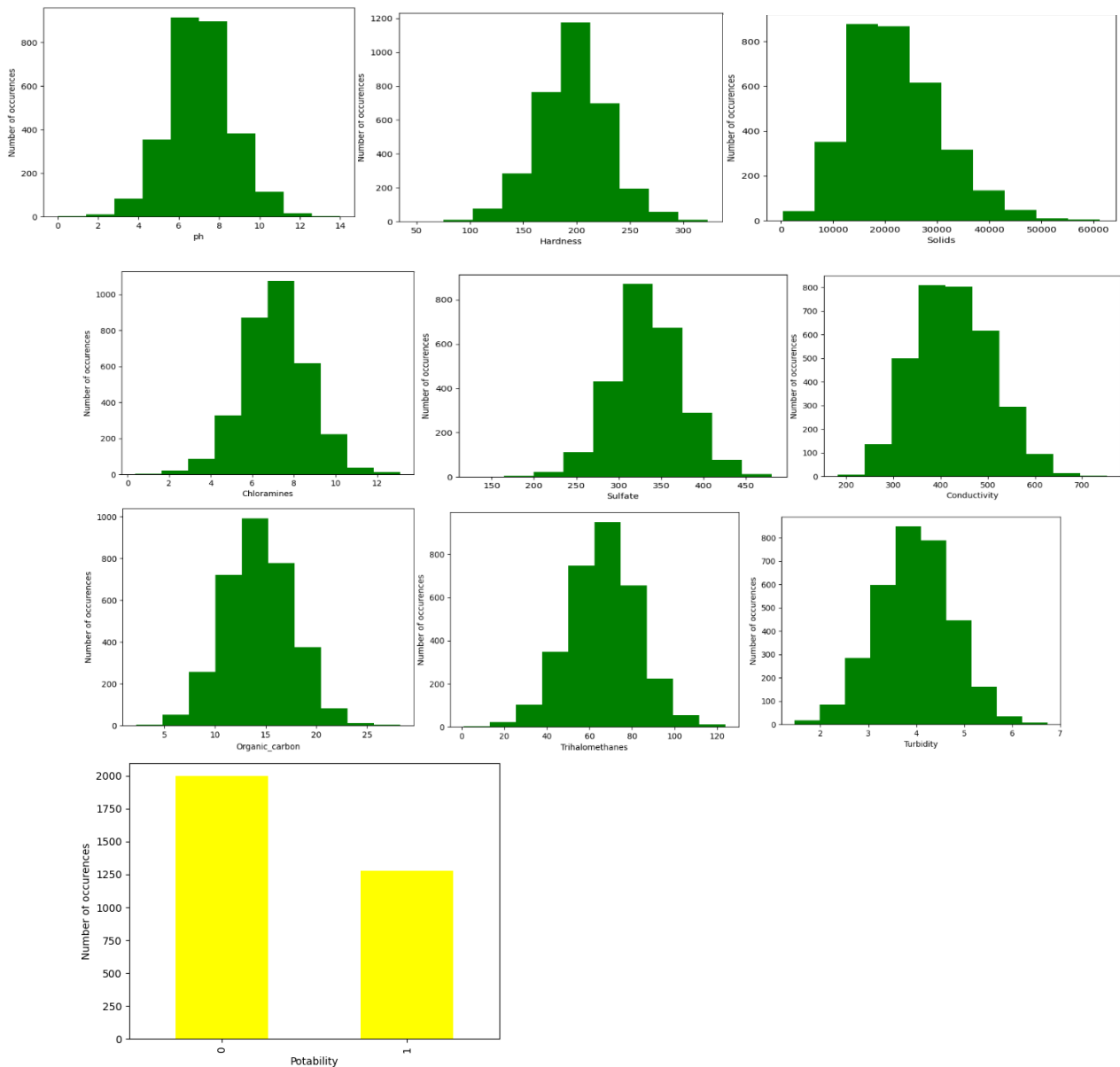
1. Ph
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 14.987790%
2. Hardness
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 0.0%
3. Solids
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 0.0%
4. Chloramines
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 0.0%

5. Sulfate
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 23.840049%
6. Conductivity
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 0.0%
7. Organic_carbon
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 14.987790%
8. Trihalomethanes
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 4.945055%
9. Turbidity
 - Variabile numerica continua
 - Illimitato
 - Valori mancanti: 0.0%
10. Potability
 - Variabile categorica nominale
 - Illimitato
 - Valori mancanti: 0.0%

Da questa analisi si comprende immediatamente che sono presenti numerosi NaN value (valori mancanti), per questo motivo sarà necessario fare con molta attenzione le operazioni di pre-processing.

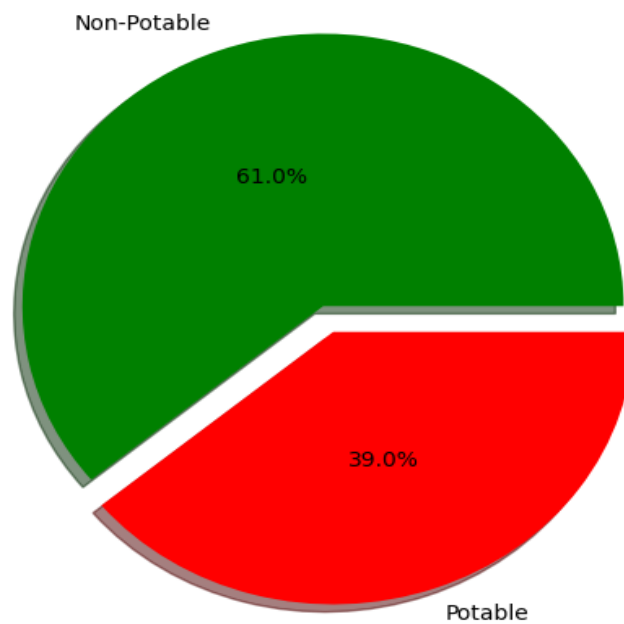
DATA VISUALIZATION

Grazie alla data visualization, è possibile mostrare gli istogrammi relativi alle variabili numeriche e i grafici a barre relativi alle variabili categoriche(solamente variabile target):

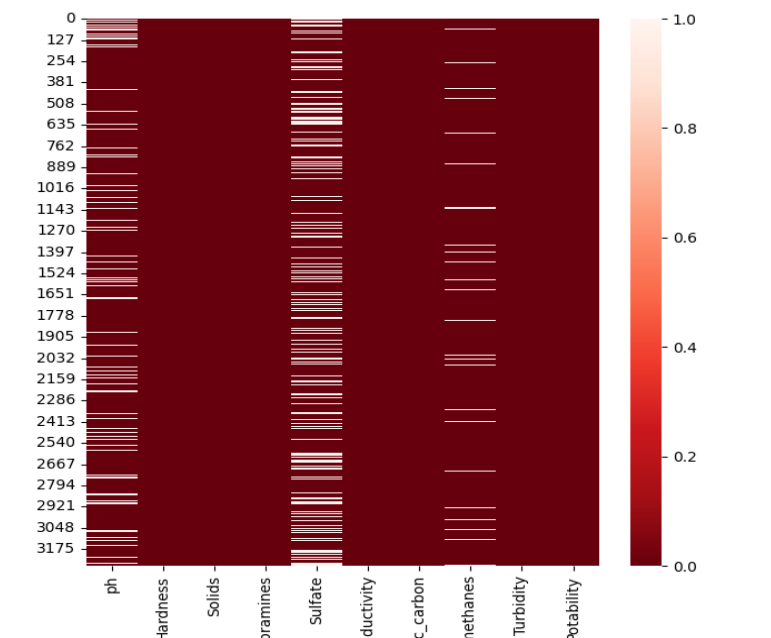


Inoltre, si può rappresentare le percentuali della variabile target con un grafico a torta:

Water Potability



Infine, viene plottata una heatmap che mostra la posizione dei NaN value. Questa visualizzazione è molto comoda per vedere se ci sono NaN distribuiti in particolari zone del dataset o meno.



GESTIONE VALORI MANCANTI

Come si può vedere, alcune feature contengono valori mancanti, per questo motivo bisogna andare a riempire o rimuovere tali valori. Le feature con valori mancanti sono 3: ph, sulfate e trihalomethanes.

Ho deciso quindi di andare a riempire questi valori con la media, visto che ho a che fare con variabili numeriche. Inoltre ho preso questa decisione perché ritenevo inopportuno andare ad eliminare i valori mancanti, perché avrei cancellato circa il 39% del dataset e avrei quindi ottenuto in seguito dei risultati non molto soddisfacenti.

FEATURE SELECTION

Il dataset, contiene quindi alcuni valori che risultano essere inutili, per questo motivo si esegue la feature selection.

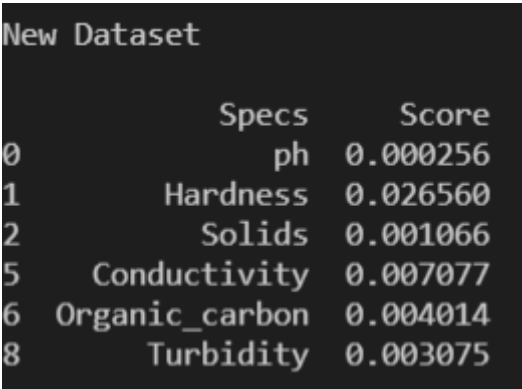
Con essa vado a eliminare quei valori che sono poco significativi per il task, in particolare andando ad utilizzare la mutua informazione.

Di seguito mostro quelli che sono gli score, ottenuti dalla mutua informazione, grazie all'utilizzo della funzione "SelectKBest":

```
Feature Selection
-----
(3276, 9)
-----
```

	Specs	Score
0	ph	0.000256
1	Hardness	0.026560
2	Solids	0.001066
3	Chloramines	0.000000
4	Sulfate	0.000000
5	Conductivity	0.007077
6	Organic_carbon	0.004014
7	Trihalomethanes	0.000000
8	Turbidity	0.003075

Si nota dai valori di “Score” che non ho degli esempi molto buoni, per questo motivo sono andato a selezionare come feature importanti tutte quelle che hanno uno score diverso da 0.



```
New Dataset
```

	Specs	Score
0	ph	0.000256
1	Hardness	0.026560
2	Solids	0.001066
5	Conductivity	0.007077
6	Organic_carbon	0.004014
8	Turbidity	0.003075

SPLITTING THE DATASET

Dopo di che, il dataset viene splittato in 20% test set e 80% training set. Il training set ha questa dimensione (2620, 6) e il test set (656, 6),.

In seguito, dopo numerosi tentativi, dal training set viene estratto un 20% di esempi grazie all'utilizzo della K-Fold Cross-Validation: il numero di split è uguale a 5 in modo tale da generare per ogni iterazione il 20% del training set, che andrà a creare il validation set.

FEATURE SCALING

Dopo la feature selection rimangono mediamente 6 feature, tali feature differiscono in ordine di grandezza fra di loro ed è stato scelto di scalare mediante z-score: esso permette di scalare le feature in modo che abbiano media nulla e varianza unitaria. Lo z-score è definito come:

$$x' = (x - \bar{x})/\sigma \text{ con } \bar{x} = \text{avg}(x).$$

Per questo tipo di scaling viene utilizzato lo standard scaler di sklearn,.

Per questo motivo è opportuno effettuare un processo di feature scaling in modo da agevolare gli algoritmi iterativi che utilizzano la discesa del gradiente, come calcolo differenziale per la minimizzazione di una funzione.

SELEZIONE MODELLO MIGLIORE (ML MODEL COMPARISON)

In questo punto, si andranno a confrontare i modelli di machine learning di classificazione per scegliere il migliore; che sarà poi sfruttato per fare fine-tuning degli hyper parametri.

In particolare, confronto i seguenti modelli: logistic regression, random forest, adaboost, gradient boosting, decision tree e XGBoosting.

Facendo svariati tentativi, andando a calcolare precision, recall e f1 score, noto fin da subito che i dati non sono di qualità perché ho, per quanto riguarda la logistic regression, tutti i valori pari a 0 e per gli altri modelli ho dei risultati estremamente bassi.

Per questo motivo, per migliorare questi risultati, utilizzo average che è un parametro dei metodi di precision, recall e f1 score e lo imposto uguale a micro (average = "micro").

Ho preso in considerazione l'utilizzo di quel tipo di parametro perché è ideale se si sospetta che ci possa essere uno squilibrio di classe (cioè si possono avere molti più esempi di una classe che di altre classi). I risultati che ho ottenuto sono i seguenti:

```
Precision logistic regression: 0.6049618320610687
Recall logistic regression: 0.6049618320610687
F1 score logistic regression: 0.6049618320610687

Precision random forest: 0.6545801526717557
Recall random forest: 0.6545801526717557
F1 score random forest: 0.6545801526717557

Precision adaboost: 0.6049618320610687
Recall adaboost: 0.6049618320610687
F1 score adaboost: 0.6049618320610687

Precision gradient boosting: 0.6316793893129771
Recall gradient boosting: 0.6316793893129771
F1 score gradient boosting: 0.6316793893129771

Precision decision tree: 0.5763358778625955
Recall decision tree: 0.5763358778625955
F1 score decision tree: 0.5763358778625955

Precision xgboosting: 0.615648854961832
Recall xgboosting: 0.615648854961832
F1 score xgboosting: 0.615648854961832
```

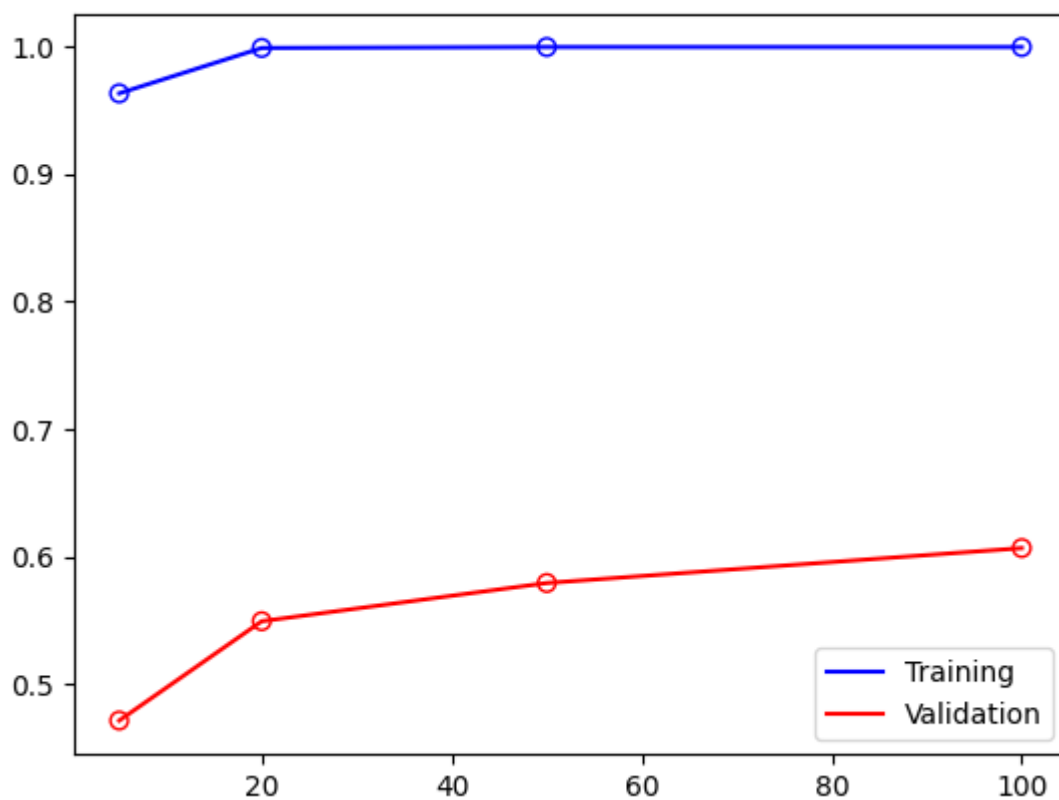
Si nota che, generalmente, il modello migliore è il random forest.

FINE TUNING CON IL MODELLO MIGLIORE

In questo punto, il modello su cui si vanno a fare operazione di fine-tuning è il Random Forest.

Avendo un dataset di modeste dimensioni si può andare a combinare insieme Grid Search e Random Search.

Vado prima ad utilizzare la Random Search con il numero di alberi nella random forest pari a `n_estimators`, visualizzando il grafico di training set e validation set per stabilire eventuale overfitting o underfitting.



Dopo diversi tentativi con grafici leggermente diversi tra loro, ho deciso di utilizzare i seguenti `n_estimators = [5, 20, 50, 100]`.

Dal seguente grafico si evince che la situazione migliore la si ha quando gli `n_estimators` sono più o meno uguali a 100, in quanto il valore di precisione del validation set è il più alto.

```
Computing 5 with validation precision: 57.86259541984733  
Computing 20 with validation precision: 63.39694656488549  
Computing 50 with validation precision: 63.282442748091604  
Computing 100 with validation precision: 65.0381679389313
```

Dopo di che, viene svolta una grid search in modo tale da riuscire a trovare il numero di estimators migliore e anche per trovare ottimi parametri di regolarizzazione.

Questi valori cambiano spesso, per cui grazie a questo algoritmo:

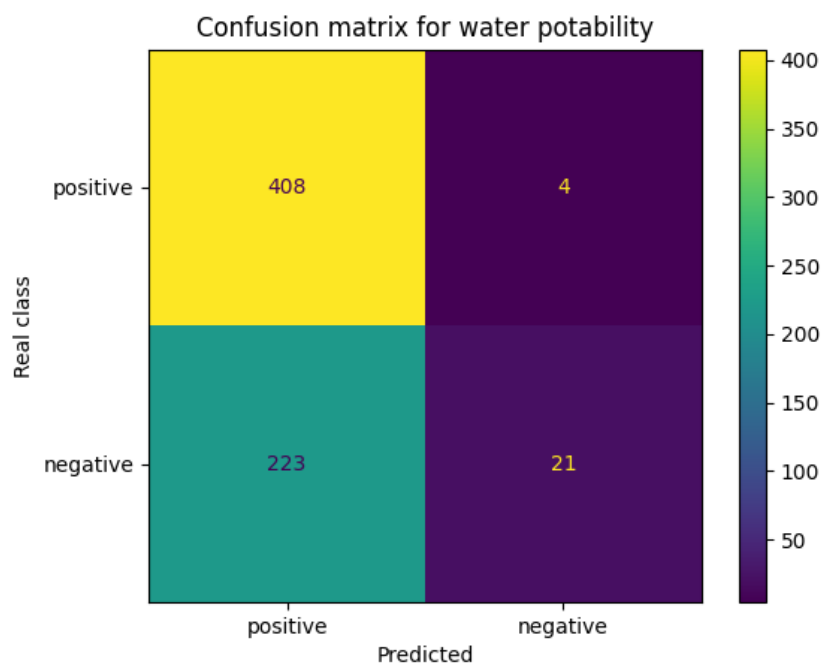
```
m = 0
md = 0
ne = 0
for i in range(len(best_score.values())):
    print(list(best_score.values())[i][2])
    if list(best_score.values())[i][2] >= m:
        m = list(best_score.values())[i][2]
        md = list(best_score.values())[i][1]
        ne = list(best_score.values())[i][0]
```

Vado sempre a prendere il migliore precision score e suoi dati relativi alla max_depth e agli n_estimators.

Per questo motivo ad ogni istanza dell'algoritmo, verranno automaticamente utilizzati i parametri migliori.

CONFUSION MATRIX

Ora vado ad utilizzare i migliori parametri, precedentemente ottenuti, per presentare la confusion matrix.

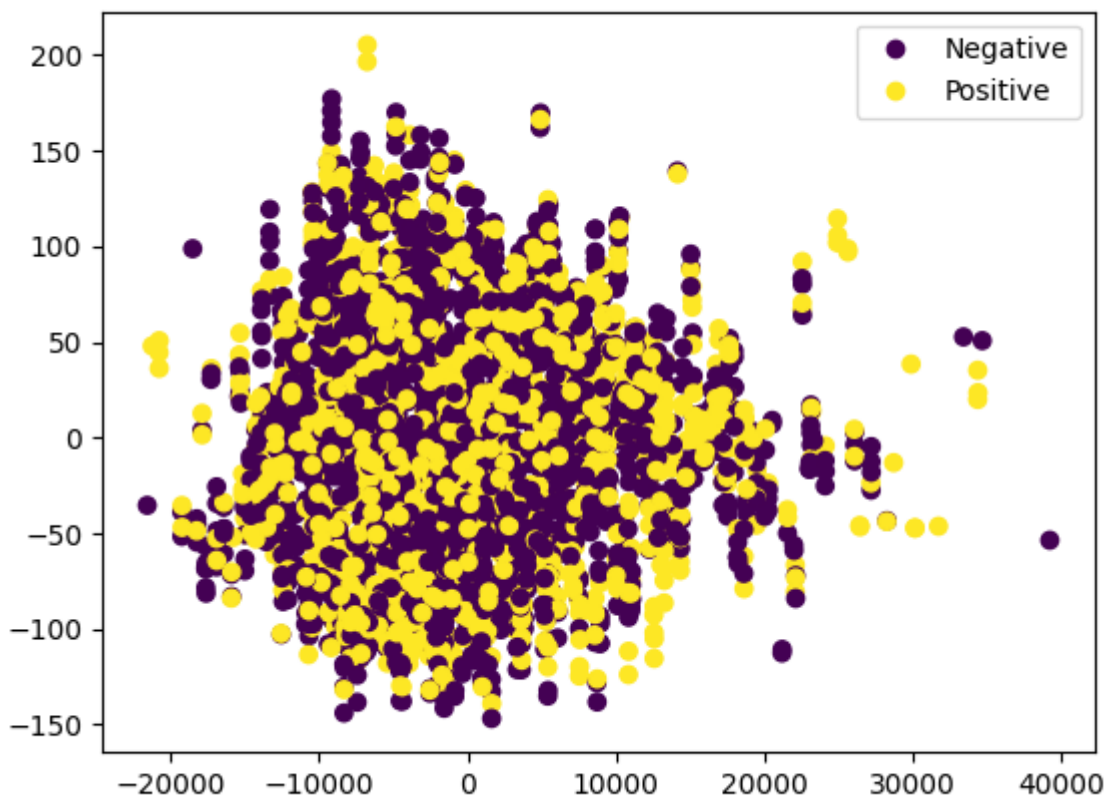


T-SNE

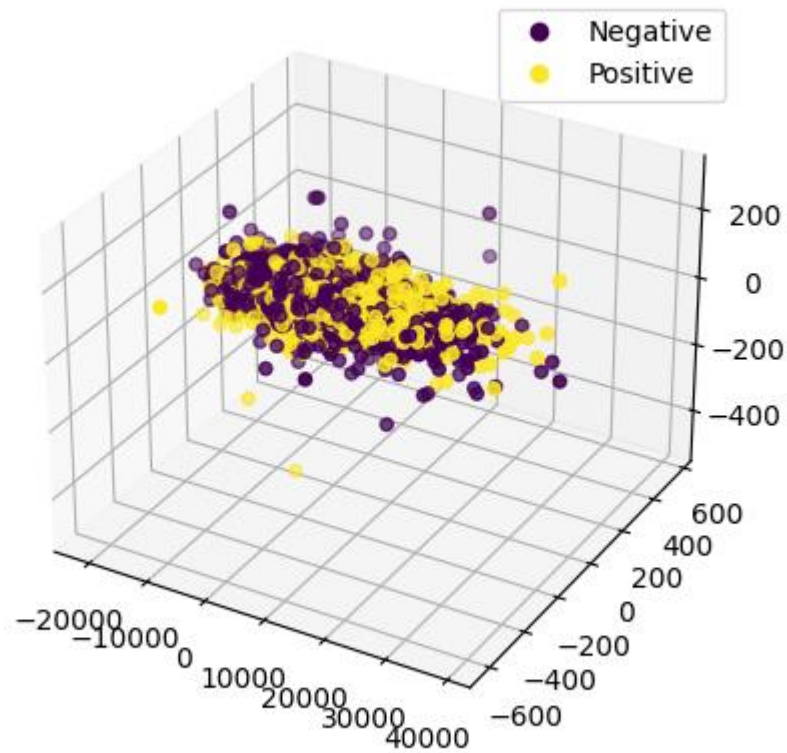
Ora utilizzo una delle tecniche di riduzione delle dimensioni dello spazio delle feature, per andare a guardare quanti esempi dicono che l'acqua risulti essere potabile e quanti no.

Per fare ciò, utilizzo nella sua interezza il dataset in quanto ha dimensioni modeste e non crea problemi computazionali.

In 2D:



In 3D:



Dai due grafici soprastanti emerge in modo piuttosto chiaro la scarsa qualità del dataset: le variabili non sono nettamente distinte e ciò aumenta la probabilità che i modelli cadano in errore.