UNIVERSITÀ DEGLI STUDI DI SALERNO

Corso di Laurea Triennale in informatica

Progetto di Fondamenti di Intelligenza Artificiale

Breast Cancer Analysis

Dario Mazza, Nicolò Delogu, Tommaso Sorrentino

Repository Github:

https://github.com/xDaryamo/BreastCancerPrediction


Breast Cancer
Analysis

Indice

1 Introduzione

Il cancro al seno è una delle patologie oncologiche più diffuse tra le donne di tutto il mondo. Esso rappresenta infatti la maggior parte dei nuovi casi di tumore che si manifesta nelle donne e ne costituisce la causa principale di decesso, il che lo rende un significativo problema di salute pubblica e sociale. Una diagnosi precoce di questa malattia può migliorare significativamente la prognosi e le possibilità di sopravvivenza, poiché permette di agire tempestivamente con un trattamento clinico per i pazienti.

Pertanto, la corretta diagnosi del tumore al seno e la corretta classificazione dello stesso è oggetto di molte ricerche.

2 Definizione del problema

2.1 Obiettivi

Lo scopo del progetto è quello di realizzare un agente intelligente capace di classificare la natura tumorale (benigna o maligna) di una cellula anomala del tessuto mammario di una paziente.

2.2 Specifica P.E.A.S

- **Performance**: La misura di performance dell'agente è l'accuratezza con cui diagnostica la natura della cellula anomala al seno di una paziente.
- **Environment**: L'ambiente in cui l'agente opera è composto dalle misurazioni effettuate sui campioni di cellule prelevate da noduli del tessuto mammario.
- **Actuators**: L'attuatore dell'agente corrisponde alla predizione effettuata.
- **Sensors**: I sensori dell'agente sono l'insieme dei dati forniti in input finalizzati all'apprendimento dello stesso.

2.2.1 Caratteristiche dell'ambiente

- **Completamente Osservabile**: l'agente ha sempre accesso a tutte le informazioni relative alle misurazioni.
- **Stocastico**: lo stato successivo dell'ambiente non dipende solo dallo stato corrente e dall'azione dell'agente ma anche da fattori indipendenti dall'agente.
- **Sequenziale**: la diagnosi calcolata è fortemente influenzata dalla sequenza di diagnosi effettuate precedentemente.
- **Dinamico**: Durante l'esecuzione dell'agente il dataset potrebbe essere aggiornato.
- **Discreto**: L'ambiente fornisce un numero limitato e ben definito di percezioni e azioni.
- **Singolo**: L'ambiente consente la presenza di un unico agente.

2.3 Analisi del problema

Una soluzione inerente al problema potrebbe essere sviluppata utilizzando le informazioni relative alle misurazioni fornite in input all'agente intelligente. A tal proposito, sono state utilizzate tecniche di **machine learning** per migliorare la qualità dei dati e di conseguenza la precisione della previsione. Abbiamo deciso di testare diversi algoritmi di machine learning andando infine a trattare il problema come un problema di **classificazione**.

3 Scelta del dataset

La creazione del dataset necessario al modello di intelligenza artificiale per effettuare previsioni di tumori maligni al seno è avvenuta modellando ed elaborando dei dati già esistenti che sono stati estrapolati da `https://www.kaggle.com/`.

3.1 Descrizione del dataset

Il dataset utilizzato, è disponibile pubblicamente al seguente link ed è stato creato dal Dottore *William H. Wolberg*, medico presso l'Ospedale dell'Università del Wisconsin. Per creare il dataset, il dottor Wolberg ha utilizzato campioni di liquidi, prelevati da pazienti attraverso l'agoaspirato al seno, e un sistema software chiamato *Xcyt*, in grado di eseguire l'analisi delle caratteristiche citologiche sulla base di una scansione digitale.

Il programma calcola dieci caratteristiche di ciascuna delle cellule del campione, ne calcola il valore medio, il valore pessimo (media delle tre misurazioni peggiori, o più grandi) e l'errore standard di ciascuna caratteristica citologica, restituendo un vettore di 30 valori reali.

3.2 Descrizione delle variabili

1. ID number (identificativo del paziente)
2. Diagnosis (M = maligno, B = benigno)
3. Unnamed: 32 (Colonna erronea dovuta a un difetto del dataset)

10 features legate alla cellula:

1. radius (media delle distanze dal centro ai punti del perimetro)
2. texture (deviazione standard della scala dei grigi)
3. perimeter (perimetro)
4. area
5. smoothness (levigatezza, cambiamenti locali al raggio)
6. compactness (compattezza, $\frac{perimetro^2}{area-1.0}$)
7. concavity (gravità dei punti concavi del contorno della cellula)
8. concave points (numero delle porzioni concave del contorno della cellula)
9. symmetry (simmetria)
10. fractal dimension (dimensione frattale, "coastline approximation" - 1)

La media (*mean*), l'errore standard (*se*) e la peggiore misurazione (*worst*), ossia la media tra le tre misurazioni peggiori, di queste features sono state calcolate per ogni immagine, ottenendo un totale di 33 colonne.

4 Ingegnerizzazione dei dati

4.1 Data cleaning

Nella prima fase di ingegnerizzazione dei dati ci siamo concentrati sul **data cleaning**, ovvero è stata verificata la presenza di dati nulli o non validi. Da questa analisi è stato ottenuto il seguente risultato:

```
#Controllo per valori null
df.isnull().any()
```

```
id                         False
diagnosis                  False
radius_mean                False
texture_mean               False
perimeter_mean             False
area_mean                  False
smoothness_mean            False
compactness_mean           False
concavity_mean             False
concave points_mean        False
symmetry_mean              False
fractal_dimension_mean     False
radius_se                  False
texture_se                 False
perimeter_se               False
area_se                    False
smoothness_se              False
compactness_se             False
concavity_se               False
concave points_se          False
symmetry_se                False
fractal_dimension_se       False
radius_worst               False
texture_worst              False
perimeter_worst            False
area_worst                 False
smoothness_worst           False
compactness_worst          False
concavity_worst            False
concave points_worst       False
symmetry_worst             False
fractal_dimension_worst    False
Unnamed: 32                 True
dtype: bool
```

```
#Controllo per valori na
df.isna().any()
```

```
id                        False
diagnosis                 False
radius_mean               False
texture_mean              False
perimeter_mean            False
area_mean                 False
smoothness_mean           False
compactness_mean          False
concavity_mean            False
concave points_mean       False
symmetry_mean             False
fractal_dimension_mean    False
radius_se                 False
texture_se                False
perimeter_se              False
area_se                   False
smoothness_se             False
compactness_se            False
concavity_se              False
concave points_se         False
symmetry_se               False
fractal_dimension_se      False
radius_worst              False
texture_worst             False
perimeter_worst           False
area_worst                False
smoothness_worst          False
compactness_worst         False
concavity_worst           False
concave points_worst      False
symmetry_worst            False
fractal_dimension_worst   False
Unnamed: 32                True
dtype: bool
```

4.1.1 Data Imputation

La colonna "**Unnamed: 32**" è completamente vuota, dato che si tratta di una colonna erronea del dataset. Dunque abbiamo deciso di rimuovere la colonna "**Unnamed: 32**".

4.2 Feature selection

Durante la fase di **feature selection** è stato ritenuto opportuno rimuovere il campo ID, in quanto superfluo ai fini dell'analisi.

Successivamente sono state trattate le problematiche relative alla **correlazione** tra le variabili **indipendenti** e la variabile **dipendente** (target della previsione) mentre la fase finale della selezione delle features si è focalizzata sull'analisi della **multicollinearità**, ovvero lo scenario in cui un gran numero di variabili indipendenti sono altamente correlate tra di loro.

4.2.1 Eliminazione univariata di feature

Figura 1: *Correlazione features con la variabile dipendente.*

Come indicato dall'istogramma (rappresentazione grafica della correlazione tra le coppie [**variabile indipendente, variabile dipendente**]), sono state rilevate alcune features che sono poco correlate con la variabile target.

Abbiamo proceduto dunque a fare una prima selezione delle nostre variabili indipendenti, rimuovendo tutte le features con un grado di correlazione inferiore al 10%.

```
corr_matrix = df.corr()
threshold = 0.1
filtre = np.abs(corr_matrix["diagnosis"]) <=
    threshold
to_delete = corr_matrix.columns[filtre].tolist()
df = df.drop(to_delete, axis = 1)
```

Il risultato della selezione è risultato nel taglio delle seguenti features:

- fractal dimension mean
- texture se
- smoothness se
- symmetry se
- fractal dimension se

Di seguito l'istogramma dopo la rimozione delle features elencate:

Figura 2: *Correlazione features con la variabile dipendente dopo la rimozione.*

4.2.2 Analisi della collinearità

La seguente matrice di correlazione mostra il grado di collinearità tra le variabili indipendenti:

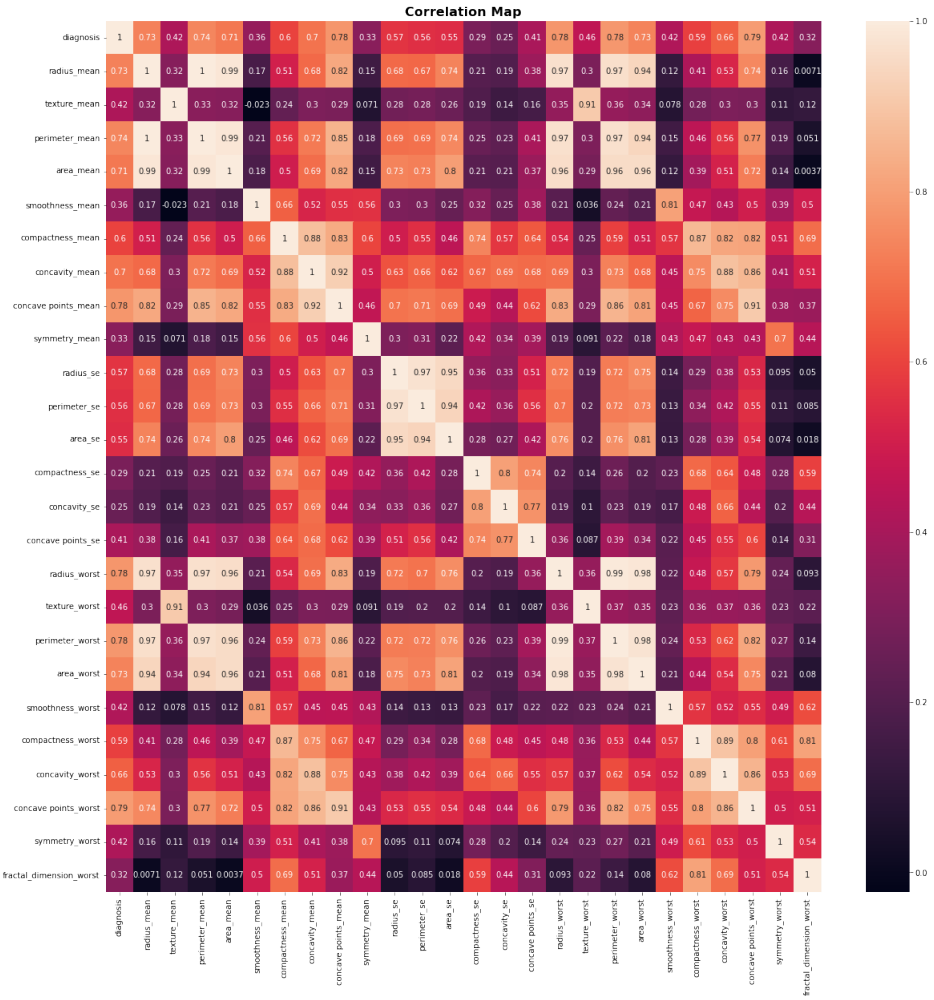

Figura 3: *Matrice di correlazione.*

Come possiamo notare dalla matrice, vi sono numerose variabili indipendenti altamente correlate tra di loro (multicollinearità). Al fine di migliorare la performance dell'agente è stato deciso di isolare tutte le variabili indipendenti con una correlazione maggiore o uguale ad una certa soglia stabilita a priori (Threshold).

Dall'operazione di isolamento otteniamo una matrice di questo tipo:

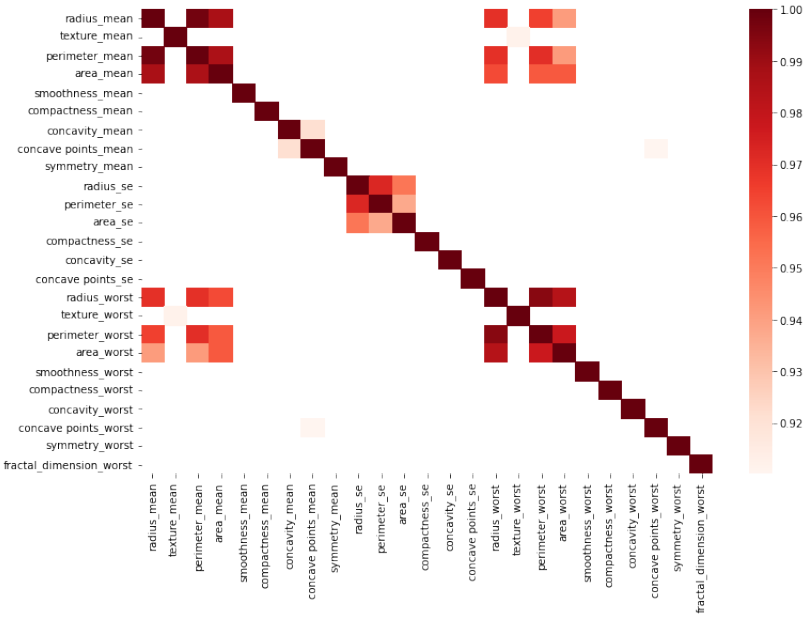

Figura 4: *Isolamento features con correlazione superiore al 90%.*

Di seguito è stata analizzata ogni coppia di features e, per ognuna di queste con correlazione maggiore o uguale alla threshold, è stata mantenuta la feature con la correlazione più alta con la variabile dipendente.

```
def remove_collinear_features(x, threshold):
    # Matrice di correlazione
    corr_matrix = x.corr()
    iters = range(len(corr_matrix.columns) - 1)
    drop_cols = []

    # Iterazione di tutte le correlazioni della matrice di
       correlazione.
    for i in iters:
        for j in range(i+1):
            item = corr_matrix.iloc[j:(j+1), (i+1):(i+2)]
            col = item.columns
            row = item.index
            val = abs(item.values)
            # Se la correlazione eccede la soglia di treshold
            if val >= threshold:
                # Eliminazione della variabile indipendente meno
                   correlata con la variabile dipendente.
                if(x[[col.values[0]]].corrwith(x.diagnosis).iloc[0]
                   >=
                   x[[row.values[0]]].corrwith(x.diagnosis).iloc[0]):
                  if row.values[0] not in drop_cols:
                    drop_cols.append(row.values[0])
                else:
                  if col.values[0] not in drop_cols:
                    drop_cols.append(col.values[0])
    return drop_cols

drop_cols = remove_collinear_features(df, 0.9)
df = df.drop(to_delete, axis = 1)
```

In seguito ad valutazioni empiriche abbiamo constatato che il valore di threshold che massimizza le prestazioni è 0.9 (90%).

Grazie a questo ulteriore raffinamento, sono state scartate le seguenti features:

- radius mean
- area mean
- concavity mean
- perimeter se
- area se
- perimeter mean
- texture mean
- radius worst
- worst
- concave points mean

Dopo aver selezionato le features migliori, visualizziamo di nuovo la matrice di correlazione per capire se la collinearità tra variabili indipendenti è stata ridotta:

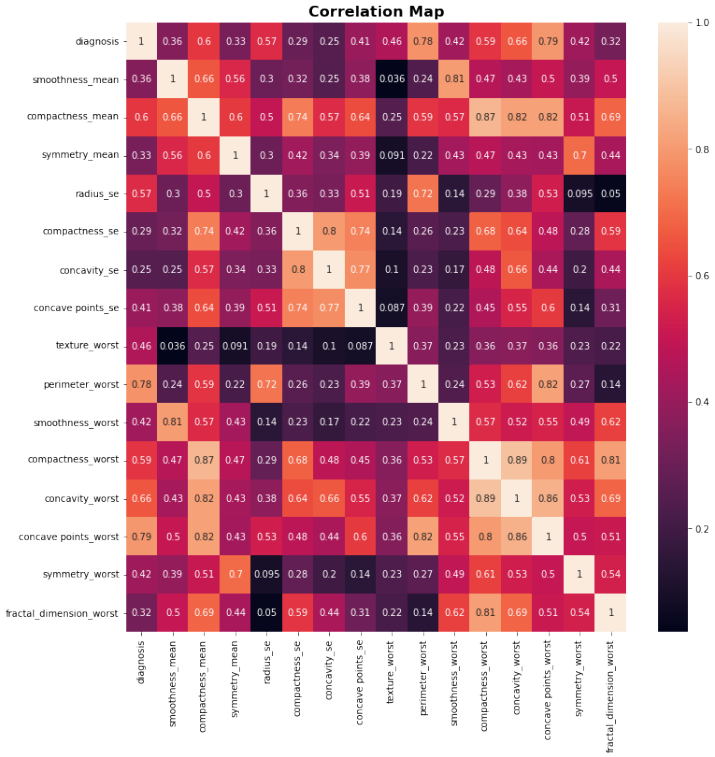

Figura 5: *Matrice di correlazione delle migliori features.*

4.3 Feature Scaling

Nella fase di **feature scaling** il focus è ricaduto sulla scelta di tecniche che consentono di normalizzare o scalare l'insieme dei valori.
In particolare sono stati prese in considerazione le seguenti strategie:

- Standard scaling: $\frac{x-\overline{x}}{\sigma}$, che normalizza i dati in modo di ottenere la somma delle medie pari a 0 e la deviazione standard uguale a 1.
- MinMax scaling: normalizza i valori dei dati in valori compresi in un intervallo $[a, b]$. In questo particolare caso è stato deciso di scegliere due range diversi per l'intervallo al fine di valutare empiricamente le prestazioni dell'agente in entrambi gli scenari.
    1. range $[a = 0, b = 1]$
    2. range $[a = -1, b = 1]$

Dopo aver confrontato le due strategie, è stato dedotto che
Standard scaling è quella che si presta meglio ai fini della risoluzione del problema.

4.4 Data Balancing

Per quanto concerne la fase di **Data Balancing**, abbiamo ritenuto opportuno non apportare modifiche al dataset.

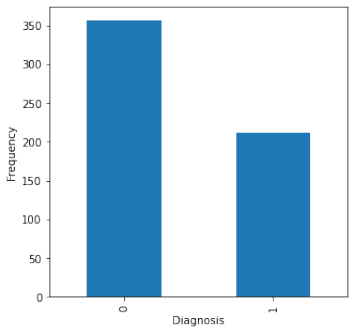

Figura 6: *Distribuzione della variabile indipendente - classe di minoranza e maggioranza.*

Come mostrato dall'istogramma, la classe di minoranza è chiaramente rappresentata dai casi con diagnosi uguale a 1 (tumore maligno).

Tuttavia, dopo un'operazione di **oversampling** per "bilanciare" questa differenza, abbiamo rilevato empiricamente che le prestazioni di tutti i modelli tendono a peggiorare.

Dato che questa tendenza a decadere è condivisa tra tutti i modelli abbiamo deciso di utilizzare il dataset privo di cambiamenti dovuti al bilanciamento.

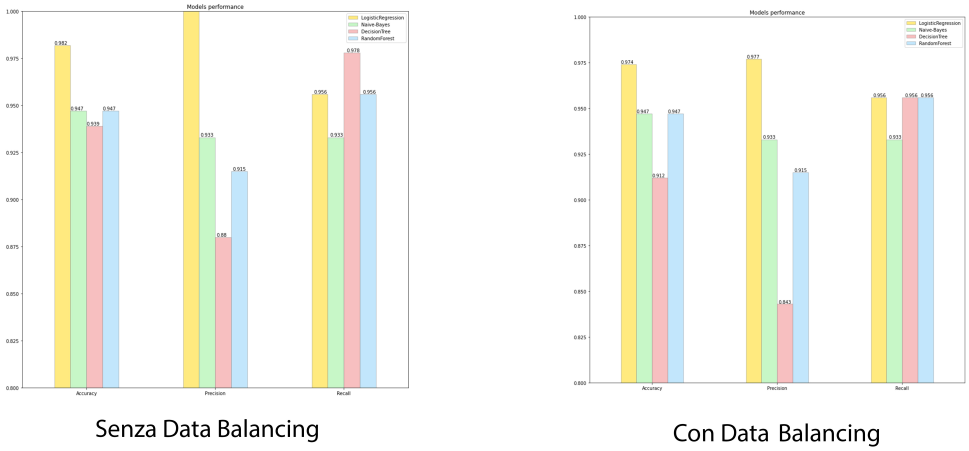

Figura 7: *Differenze di prestazioni senza e con data balancing.*

5 Soluzione al problema

5.1 Validazione del modello

In seguito alla definizione delle tecniche di ingegnerizzazione dei dati dell'agente intelligente, è necessario stabilire le metriche e le tecniche di validazione delle prestazioni dello stesso. Occorre suddividere l'insieme dei dati fin'ora analizzato in due insiemi: il training set, composto dalle istanze di dati che saranno utilizzate per l'addestramento, e il test set, composto dalle istanze di dati per cui l'agente dovrà predire il valore della variabile dipendente.

Abbiamo scelto di dividere il dataset in questo modo:

- Test set: 20% del dataset iniziale.
- Training set: 80% del dataset iniziale.

5.1.1 Logistic Regression

La Logistic Regression è utilizzata comunemente per stimare la probabilità di appartenenza a particolari classi, ciò la rende un classificatore binario perfetto per il nostro problema.
Riportiamo di seguito i risultati ottenuti:

Test	Classificatore	Scaler	Precision	Recall	Accuracy	Threshold
1	Logistic Regression	Z-Score	1.0	0.955555	0.982456	0.9
2	Logistic Regression	MinMax(0,1)	0.954545	0.933333	0.956140	0.9
3	Logistic Regression	MinMax(-1,1)	0.954545	0.933333	0.956140	0.9
4	Logistic Regression	NA	0.931818	0.911111	0.938596	0.9
5	Logistic Regression	Z-Score	0.954545	0.933333	0.956140	0.8
6	Logistic Regression	MinMax(0,1)	0.976190	0.911111	0.956140	0.8
7	Logistic Regression	MinMax(-1,1)	0.954545	0.933333	0.956140	0.8
8	Logistic Regression	NA	0.918918	0.755555	0.877192	0.8
9	Logistic Regression	Z-Score	0.857142	0.66666	0.824561	0.7
10	Logistic Regression	MinMax(0,1)	0.904761	0.422222	0.754385	0.7
11	Logistic Regression	MinMax(-1,1)	0.916666	0.488888	0.780701	0.7
12	Logistic Regression	NA	0.806451	0.555555	0.771929	0.7

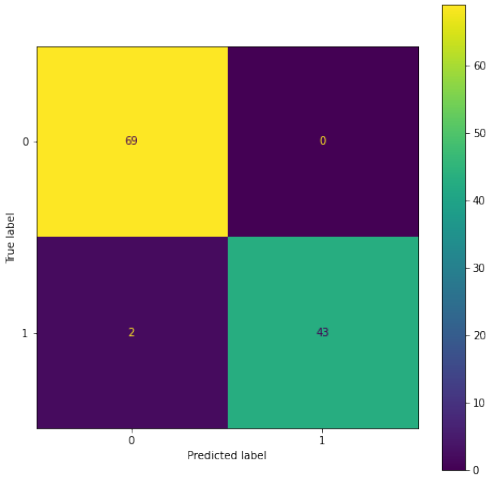

Figura 8: *Test 1 con parametri ottimali.*

5.1.2 Naive Bayes

Data la risoluzione della multicollinearità nella fase di feaure selection, abbiamo pensato di poter utilizzare un classificatore Baeysiano, in particolare il GaussianNB, questo tipo di algoritmo assume che le caratteristiche non siano correlate l'una all'altra.
Riportiamo di seguito i risultati ottenuti:

Test	Classificatore	Scaler	Precision	Recall	Accuracy	Threshold
1	GaussianNB	Z-Score	0.933333	0.933333	0.947368	0.9
2	GaussianNB	MinMax(0,1)	0.656716	0.977777	0.789473	0.9
3	GaussianNB	MinMax(-1,1)	0.656716	0.977777	0.789473	0.9
4	GaussianNB	NA	0.933333	0.933333	0.947368	0.9
5	GaussianNB	Z-Score	0.872340	0.911111	0.912180	0.8
6	GaussianNB	MinMax(0,1)	0.626865	0.933333	0.754385	0.8
7	GaussianNB	MinMax(-1,1)	0.626865	0.933333	0.754385	0.8
8	GaussianNB	NA	0.872340	0.911111	0.912280	0.8
9	GaussianNB	Z-Score	0.848484	0.622222	0.807017	0.7
10	GaussianNB	MinMax(0,1)	0.866666	0.577777	0.798245	0.7
11	GaussianNB	MinMax(-1,1)	0.866666	0.577777	0.798245	0.7
12	GaussianNB	NA	0.848484	0.622222	0.807017	0.7

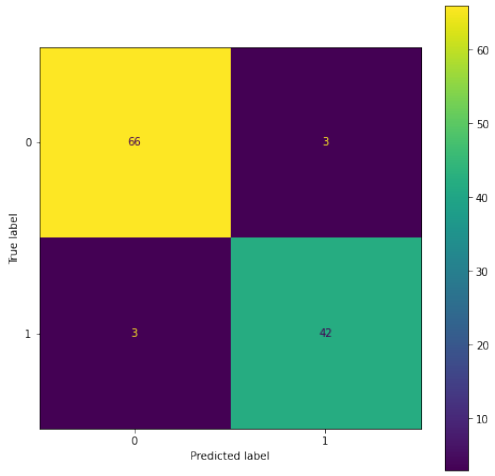

Figura 9: *Test 1 con parametri ottimali.*

5.1.3 Decision Tree

Il Decision Tree mira a creare un albero i cui nodi rappresentano un sotto-insieme di caratteristiche del problema e i cui archi rappresentano delle decisioni.
Riportiamo di seguito i risultati ottenuti:

Test	Classificatore	Scaler	Precision	Recall	Accuracy	Threshold
1	Decision Tree	Z-Score	0.88	0.979777	0.938596	0.9
2	Decision Tree	MinMax(0,1)	0.758620	0.977777	0.868421	0.9
3	Decision Tree	MinMax(-1,1)	0.781818	0.955555	0.877192	0.9
4	Decision Tree	NA	0.862745	0.977777	0.929824	0.9
5	Decision Tree	Z-Score	0.875	0.933333	0.921052	0.8
6	Decision Tree	MinMax(0,1)	0.741379	0.955555	0.850877	0.8
7	Decision Tree	MinMax(-1,1)	0.732142	0.911111	0.838383	0.8
8	Decision Tree	NA	0.869565	0.88888	0.903508	0.8
9	Decision Tree	Z-Score	0.756756	0.62222	0.771929	0.7
10	Decision Tree	MinMax(0,1)	0.826086	0.422222	0.736842	0.7
11	Decision Tree	MinMax(-1,1)	0.833333	0.444444	0.745614	0.7
12	Decision Tree	NA	0.8	0.622222	0.789473	0.7

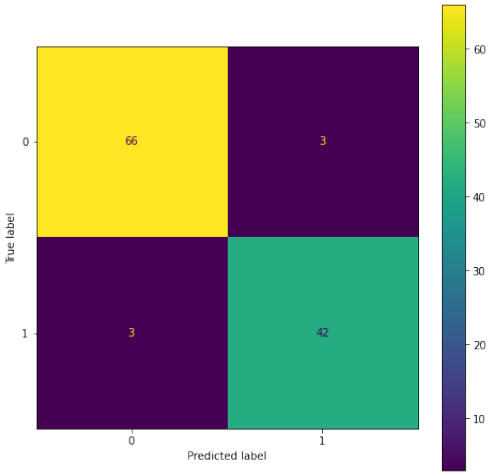

Figura 10: *Test 1 con parametri ottimali.*

5.1.4 Random Forest

Il RandomForest è un algoritmo di apprendimento supervisionato formato da un insieme di diversi alberi decisionali. Una foresta casuale stabilisce il risultato in base alle previsioni degli alberi decisionali prendendo l'output più frequente. In questo modo è possibile eliminare i limiti di un algoritmo dell'albero decisionale poiché riduce l'overfitting dei set di dati e aumenta la precisione ed è ottimo per gestire dati mancanti producendo buoni risultati. Riportiamo di seguito i risultati ottenuti:

Test	Classificatore	Scaler	Precision	Recall	Accuracy	Threshold
1	RandomForest	Z-Score	0.976190	0.911111	0.956140	0.9
2	RandomForest	MinMax(0,1)	0.843137	0.955555	0.912280	0.9
3	RandomForest	MinMax(-1,1)	0.826923	0.955555	0.903508	0.9
4	RandomForest	NA	0.954545	0.933333	0.956140	0.9
5	RandomForest	Z-Score	0.933333	0.933333	0.947368	0.8
6	RandomForest	MinMax(0,1)	0.911111	0.911111	0.929824	0.8
7	RandomForest	MinMax(-1,1)	0.869665	0.888888	0.903508	0.8
8	RandomForest	NA	0.933333	0.933333	0.947368	0.8
9	RandomForest	Z-Score	0.906295	0.60444	0.833333	0.7
10	RandomForest	MinMax(0,1)	0.9	0.4	0.745614	0.7
11	RandomForest	MinMax(-1,1)	0.888888	0.355555	0.728070	0.7
12	RandomForest	NA	0.911746	0.688888	0.850877	0.7

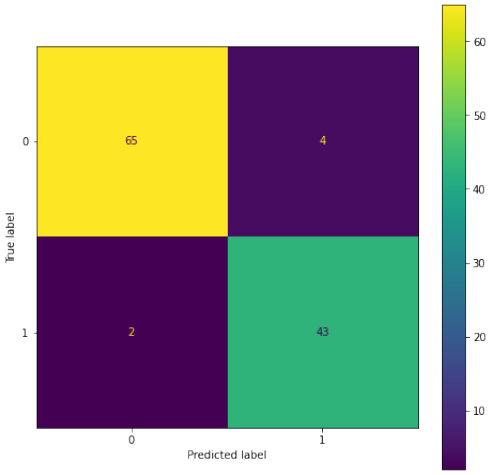

Figura 11: *Test 1 con parametri ottimali.*

6 Valutazioni finali e possibili miglioramenti

Come si può evincere dalla *Figura 12*, il modello che si è prestato meglio al nostro problema si è rivelato essere **LogisticRegression**, con le seguenti metriche:

- **Recall**: 0.95
- **Precision**: 1.00
- **Accuracy**: 0.98

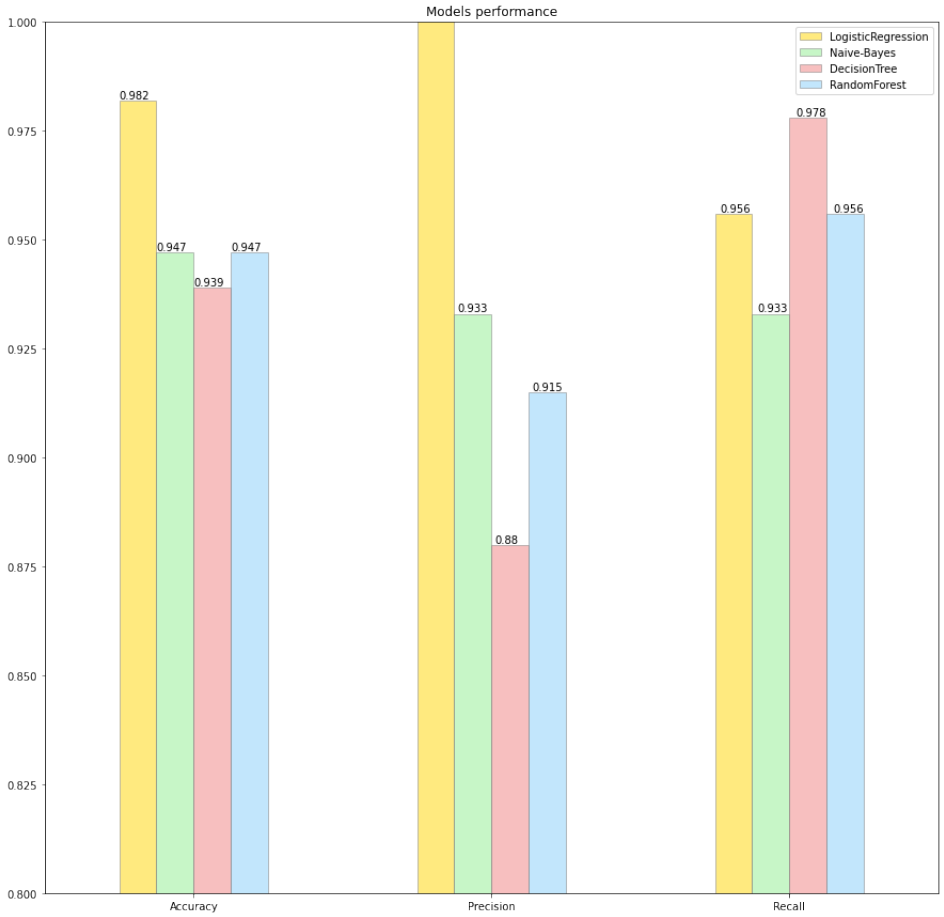

Figura 12: *Metriche dei vari modelli nella soluzione finale.*

Tra i possibili eventuali miglioramenti, elenchiamo quelli che abbiamo reputato più adatti al nostro contesto:

- **Tuning dei parametri:** sarebbe stato utile usare algoritmi di ricerca per ottimizzare i parametri dei modelli di machine learning utilizzati.
- **Data balancing:** Abbiamo provato a bilanciare il nostro dataset utilizzando l'oversampling, un'alternativa da provare sarebbe potuta essere l'undersampling.
- **Cross-validation:** utilizzare la convalida incrociata per cercare di ridurre al minimo l'overfitting.
- **Utilizzo di più modelli:** nel nostro caso abbiamo usato solo 4 modelli, si potrebbero provare ad utilizzarne altri per valutare le prestazioni.
- **Reinforcement Learning:** Si poteva pensare di applicare tecniche di apprendimento automatico, per rendere l' agente più autonomo.
- **Integrazione con altri software:** Come detto in precedenza, il dataset utilizzato è stato prodotto a partire dall'elaborazione di immagini digitalizzate con l'ausilio del software *Xcyt*. Un'idea interessante sarebbe potuta essere quella di integrare il nostro modulo di intelligenza artificiale all'interno di *Xcyt*.

7 Glossario

Termini, abbreviazioni e sigle.

- **Specifica P.E.A.S.**: Performance Environment Actuators Sensors, sintesi delle specifiche da considerare nello studio dell'intelligenza Artificiale.
- **Machine Learning**: branca dell'intelligenza artificiale che si occupa del design di sistemi che apprendono e migliorano le proprie performance in base alla distribuzione dei dati che utilizzano.
- **Dataset**: è un insieme di dati.
- **Feature**: è una proprietà individuale e misurabile, una caratteristica.
- **Variabile dipendente (y)**: variabile target che deve essere calcolata tramite il modello di machine learning attraverso i valori delle variabili indipendenti.
- **Variabile indipendente (x)**: variabile che il modello sfrutta per il calcolo del valore della variabile dipendente.
- **Data cleaning**: insieme di strategie per la gestione di dati mancanti o rumorosi.
- **Feature scaling**: insieme di tecniche per normalizzare o scalare range di valori delle caratteristiche molto diversi tra di loro.
- **MinMax scaling**: strategia di normalizzazione dei dati, normalizza i valori dei dati, in valori compresi in un intervallo $[a, b]$.
- **Standard scaling**: strategia di normalizzazione dei dati, normalizza i dati secondo l'algoritmo Z-Score Normalization.
- **Z-Score Normalization**: algoritmo di normalizzazione, normalizza i valori dei dati in modo da avere la somma delle medie pari a 0 e la deviazione standard pari a 1.
- **Deviazione standard**: indice di dispersione statistica, è uno dei modi per esprimere la dispersione dei dati, inclusa le loro medie o stime, ha la stessa unità di misura dei dati osservati.

- **Cross-fold validation**: La convalida incrociata è una tecnica statistica che consiste nella suddivisione dell'insieme di dati totale in k parti uguali e, a ogni passo, la $k-esima$ parte dell'insieme di dati viene usata come test set, mentre la restante parte costituisce sempre il training set.

- **Data balancing**: processo per la conversione di un dataset sbilanciato in uno bilanciato.

- **Model evaluation**: processo finalizzato a trovare il miglior modello che rappresenta i nostri dati e come il modello scelto funzionerà in futuro.

- **Multicollinearità**: fenomeno in cui una o più variabili indipendenti possono essere previste linearmente dalle altre con un sostanziale range di precisione.

- **Classificazione**: processo in cui l'obiettivo è predire il valore di una variabile dipendente di tipo categorico mediante l'uso di un training set.

- **Confusion Matrix**: : matrice restituita da un algoritmo di validazione di un modello, indica in quanti casi il modello ha predetto correttamente o meno il valore di una variabile dipendente del test set. Le sue componenti sono: TP, TN, FP, FN.

- **Data Imputation**: Insieme di tecniche che possono stimare il valore di dati mancanti sulla base dei dati disponibili oppure mitigare il problema dei dati mancanti.

- **Feature selection**: Processo tramite il quale vengono selezionate le caratteristiche più correlate al problema in esame, a partire da un insieme di caratteristiche esistenti.

- **Correlazione**: misura statistica che esprime la relazione lineare tra due variabili, molto usata per descrivere semplici relazioni tra dati senza dover parlare di causa ed effetto.

- **Oversampling**: aggiunta di record ad un dataset per bilanciare le frequenze delle classi.

- **Undersampling**: processo di rimozione di campioni ad un dataset per bilanciare le frequenze delle classi.

- **Overfitting**: : fenomeno che si verifica quando un modello si adatta eccessivamente ai dati di training e di conseguenza non è in grado di stimare accuratamente dati reali.

- **Logistic Regression(logit)**: è un modello di regressione non lineare utilizzato quando la variabile dipendente è di tipo dicotomico (binario, ammette solo due modalità). L'obiettivo del modello è di stabilire la probabilità con cui un'osservazione può generare uno o l'altro valore della variabile dipendente; può inoltre essere utilizzato per classificare le osservazioni, in base alla caratteristiche delle stesse, in due categorie.

- **Naive Bayes**: algoritmo di classificazione basato sul teorema di Bayes che stabilisce la probabilità del verificarsi di un evento dato il verificarsi di un altro. Questo si basa su due assunzioni: l'indipendenza degli attributi e la medesima importanza delle features.

- **Decision Tree**: algoritmo il cui obbiettivo è creare un albero i cui nodi rappresentano un sottoinsieme di features e i cui archi rappresentano delle condizioni decisionali.

- **Random Forest**: metodo di apprendimento collettivo per la classificazione, la regressione e altri compiti che funziona costruendo una moltitudine di alberi decisionali nella fase di training del modello.

- **Scaler**: indica il tipo di strategia di scaling utilizzata.

- **Precision**: metrica di valutazione definita dalla seguente espressione: $\frac{TP}{(TP+FP)}$. Indica la quantità di predizioni corrette per la classe True rispetto a ogni predizione effettuate dal modello.

- **Classificatore** indica la tipologia di algoritmo utilizzata dal modello per la risoluzione del problema.

- **Recall**: metrica di valutazione definita dalla seguente espressione: $\frac{TP}{(TP+FN)}$. Indica il numero di predizioni corrette per la classe True rispetto al totale di istanze positive di quella classe.

- **Accuracy**: metrica di valutazione definita dalla seguente espressione: $\frac{TP+TN}{(TP+TN+FP+FN)}$. Indica il numero totale di predizioni corrette.

- **Threshold**: soglia aribtraria che indica un certo grado di correlazione.

- **TP (True positive)**: Istanze positive predette come positive.

- **TN (True negative)**: Istanze negative predette come negative.

- **FP (False positive)**: Istanze negative predette come positive.
- **FN (False negative)**: Istanze positive predette come negative.
- **pandas(ps)**: è uno strumento di analisi e manipolazione dei dati open source veloce, potente, flessibile e facile da usare, costruito sopra il linguaggio di programmazione Python.
- **sklearn**: è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy.