Report

Cella Arianna (363746) ed Azzi Elena (358514)

Sommario

Analisi del Paper "Improving Land Cover Classification Using Genetic Programming for Feature Construction"

Introduzione

Il paper "Improving Land Cover Classification Using Genetic Programming for Feature Construction" esplora l'uso dell'algoritmo M3GP per migliorare la **classificazione delle coperture del suolo**, generando nuove hyperfeatures a partire dalle bande spettrali delle immagini satellitari di diversi Paesi.

L'obiettivo dello studio è stato confrontare i risultati ottenuti con i dataset standard e quelli arricchiti con hyperfeatures, valutando se l'aggiunta di queste nuove caratteristiche migliorasse le prestazioni di alcuni algoritmi di ML come Decision Tree, Random Forest e XGBoost.

Dataset

Nel paper vengono utilizzati nove dataset, ottenuti da immagini satellitari Landsat-7, Landsat-8 e Sentinel-2A, per addestrare i modelli nel classificare le aree bruciate e le coperture del suolo a livello di pixel.

Table 2. Summary of the datasets used.

Dataset	Classes (No. Pixels)			No. Classes	No. Bands No. Features	Total Pixels
Ao2	Burnt (1573)	Non-Burnt (2309)		2	7	3882
Br2	Burnt (2033)	Non-Burnt (2839)		2	7	4872
Cd2	Burnt (877)	Non-Burnt (1972)		2	7	2849
Gw2	Burnt (1101)	Non-Burnt (3430)		2	7	4531
IM-3	Savanna Woodland (114)	Dense Forest (68)	Open Forest (140)	3	6	322
IM-10	Agriculture/Bare Soil (950) Grassland (75) Savanna Woodland (1626) Water (908)	Burnt (77) Mangrove (1240) Sand (166)	Dense Forest (524) Open Forest (723) Mud (509)	10	6	6798
Ao8	Agriculture/Bare Soil (73) Forest (662) Savanna Woodland (598)	Burnt (301) Grassland (12) Water (152)	Clouds (332) Urban (53)	8	10	2183
Gw10	Agriculture/Bare Soil (449) Grassland (16) Savanna Woodland (1308) Wetland (389)	Burnt (157) Mangrove (1383) Sand (50)	Dense Forest (62) Open Forest (646) Water (620)	10	7	5080
Mz6	Agriculture/Bare Soil (33611) Urban (4194)	Forest (63190) Wetland (35673)	Grassland (28406) Other (25128)	6	10	190202

I dataset di classificazione binaria hanno l'obiettivo di addestrare i modelli per identificare le aree bruciate, classificando ogni pixel come "bruciato" o "non bruciato".

I dataset di classificazione multiclasse hanno l'obiettivo di addestrare i modelli a classificare correttamente ogni pixel come uno dei diversi tipi di copertura del suolo.

Nella nostra analisi abbiamo usato 2 dataset per verificare i risultati ottenuti, uno binario (br2) e uno multiclasse (IM3).

Dataset Br2

Classificazione binaria per identificare le aree bruciate del Brasile, utilizzando il satellite Landsat-8; le due classi con cui classificare i dati sono:

- Burnt (2033 campioni)
- Non-Burnt (2839 campioni)

Dataset IM3

Dati di tre tipi di copertura forestale provenienti dalla Guinea Bissau, recuperati attraverso il satellite Landsat-7. Questo dataset è stato estratto dal dataset IM-10 (10 classi), estraendo solo i pixel classificati dei tre tipi di copertura forestale che i modelli di ML non sono riusciti a discriminare correttamente.

Le tre classi con cui classificare i dati sono:

- Savanna Woodland (114 campioni)
- Dense Forest (68 campioni)
- Open Forest (140 campioni)

Metodologie in breve

Lo studio mira a migliorare le prestazioni dei modelli di Machine Learning espandendo i dataset di riferimento con hyperfeatures. Ciascun dataset viene suddiviso in training set (70%) e test set (30%), e un algoritmo di feature construction (M3GP) genera le corrispondenti hyperfeatures.

Vengono testati vari algoritmi di ML, tra cui Decision Tree (DT), Random Forest (RF) e XGBoost (XGB). Ogni esperimento viene ripetuto 30 volte con differenti suddivisioni casuali del dataset.

Tools e Parametri

Tutti gli esperimenti vengono eseguiti utilizzando l'implementazione dell'algoritmo M3GP (codice Python disponibile su https://github.com/jespb/Python-M3GP), che include l'implementazione del classificatore MD.

In particolare, la Mahalanobis Distance (MD) è un algoritmo non parametrico basato su cluster che classifica un dato associandolo al centroide del cluster più vicino, usando la distanza di Mahalanobis.

Le implementazioni DT e RF appartengono alla libreria Python sklearn, mentre XGB appartiene alla libreria Python xgboost.

I parametri principali utilizzati in questo lavoro e le variazioni sono specificate nella Tabella 3. Tra le variazioni presenti si ha l'implementazione dell'M3GP utilizzando il WAF del classificatore MD come fitness, piuttosto che l'accuratezza. Abbiamo verificato che questa modifica introdotta dagli autori ha portato a dei risultati migliori rispetto all'utilizzo dell'accuracy come fitness.

In particolare, WAF sta per "Weighted Average of F-measures", ovvero la media pesata delle F-measure. Questa metrica risulta particolarmente utile in problemi di classificazione multiclasse o in presenza di classi sbilanciate, perché tiene conto sia delle performance sui singoli gruppi sia della loro dimensione relativa.

Table 3. Main parameters and variations used in the experiments.

General:	
Runs	30
Training Set	70% of the samples of each class
Statistical Significance	p-value < 0.01 (Kruskal–Wallis H-test)
M3GP:	
Stopping Criteria	50 generations or 100% training accuracy
Fitness	WAF (Weighted Average of F-measures)
Pruning	Final individual
XGBoost:	
Maximum Depth	20 in the Mz6 dataset and 6 (default) in the other datasets

Risultati

Iniziamo questa sezione presentando i risultati e le hyperfeatures ottenute eseguendo M3GP su tutti i dataset. Dopo questo, discutiamo l'interpretabilità delle hyperfeatures e la popolarità delle diverse bande satellitari nelle soluzioni proposte da M3GP. Successivamente, confrontiamo l'accuratezza complessiva e l'accuratezza delle classi ottenute eseguendo gli algoritmi DT, RF e XGB sui set di dati originali e sui set di dati espansi con hyperfeatures.

*Risultati sul **dataset binario Br2***

In questa fase viene usata M3GP per generare le hyperfeatures e sono stati registrati nella seguente tabella i valori di accuratezza che ha ottenuto sul dataset Brazil2.

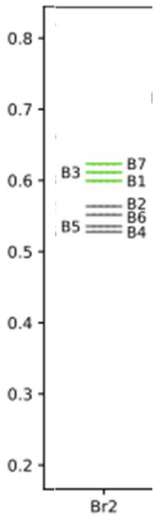
Inoltre, nella tabella vengono riportati il numero di hyperfeatures generate e la dimensione media delle hyperfeatures, con i valori minimi e massimi tra parentesi.

	Risultati del paper	*Nostri risultati*
Accuracy training	0.992	0.992
Accuracy test	0.990	0.990
Numero di Hyperfeatures	8 (1-15)	8 (1-12)
Dimensione media Hyperfeatures	14 (6-39)	27 (7-53)

Successivamente è stata fatta un'analisi sulla popolarità delle diverse bande satellitari nelle hyperfeatures evolute. Si è verificato se una banda è presente in una hyperfeature, senza misurarne l'importanza all'interno della stessa.

Nella figura a destra sono riportati i risultati della percentuale di presenza di ciascuna banda del paper, mentre i nostri risultati sono i seguenti:

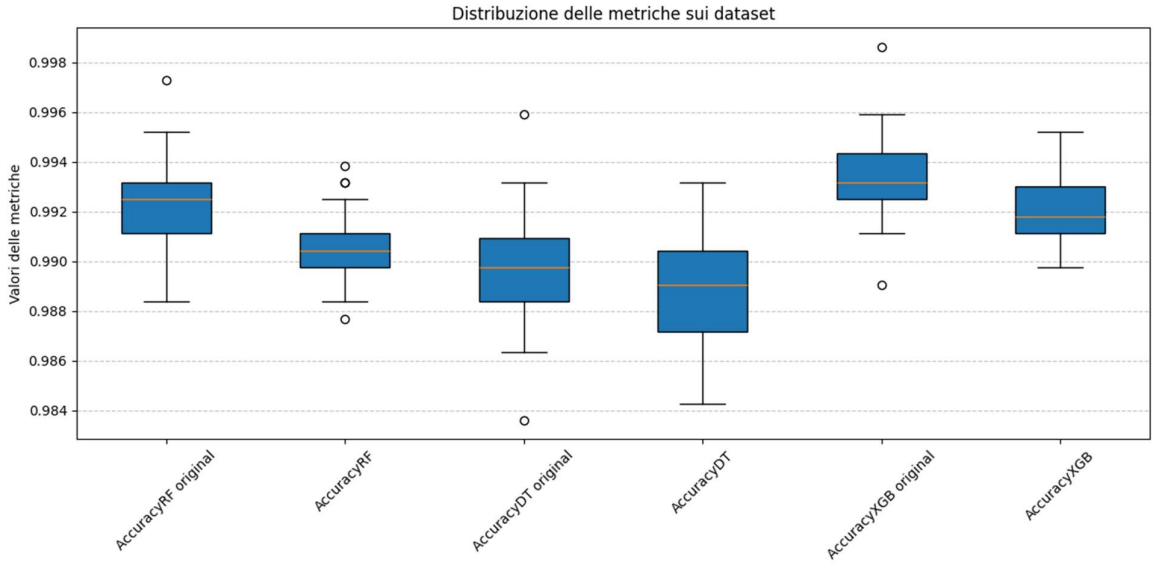
- 'B3': 0.67
- 'B7': 0.67
- 'B1': 0.66
- 'B2': 0.64
- 'B4': 0.62
- 'B5': 0.57
- 'B6': 0.55



Risultati paper

In un secondo momento, sono state valutate le prestazioni di tre modelli di classificazione (Random Forest, Decision Tree e XGBoost) sia sul dataset originale che sui 30 dataset ottenuti aumentando quello originale sulla base delle hyperfeatures del miglior modello di ciascuna run. Come è possibile verificare dal confronto riportato dalla tabella e dalla visualizzazione tramite boxplot, l'utilizzo di M3GP non ha portato alcun tipo di miglioramento.

TEST ACCURACY	*Decision Tree*	*Random Forest*	*XGBoost*
Risultati Paper sul Dataset Originale	0.989	0.992	0.993
Nostri risultati sul Dataset Originale	0.989	0.992	0.993
Risultati Paper M3GP	0.990	0.992	0.993
Nostri risultati M3GP	0.989	0.991	0.992



Successivamente viene riportato il confronto tra l'accuracy media per classe usando XGBoost come classificatore; abbiamo calcolato anche l'accuracy per classe sui nuovi risultati ottenuti aggiungendo le hyperfeatures, notando anche in questo caso come non ci sia stato un effettivo miglioramento delle prestazioni.

	Risultati paper di XGB original	*Nostri risultati su XGB original*	*Nostri risultati su XGB con Hyperfeatures*
Burnt	99.51%	99.59%	99.60%
Other	99.16%	99.12%	98.89%

Risultati sul ***dataset multiclasse IM3***

In questa fase viene usata M3GP per generare le hyperfeatures e sono stati registrati nella seguente tabella i valori di accuratezza che ha ottenuto sul dataset IM3.

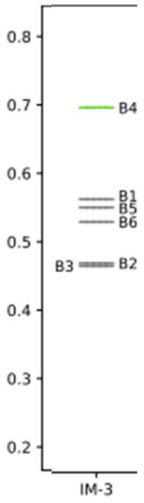
Inoltre, nella tabella vengono riportati il numero di hyperfeatures generate e la dimensione media delle hyperfeatures, con i valori minimi e massimi tra parentesi.

	Risultati del paper	*Nostri risultati*
Accuracy training	0.996	0.998
Accuracy test	0.948	0.947
Numero di Hyperfeatures	8.5 (5-13)	8.6 (4-12)
Dimensione media Hyperfeatures	11 (5-22)	27 (6-69)

Successivamente è stata fatta un'analisi sulla popolarità delle diverse bande satellitari nelle hyperfeatures evolute. Si è verificato se una banda è presente in una hyperfeature, senza misurarne l'importanza all'interno della stessa.

Nella figura a destra sono riportati i risultati della percentuale di presenza di ciascuna banda del paper, mentre i nostri risultati sono i seguenti:

- 'B4': 0.77
- 'B5': 0.67
- 'B1': 0.64
- 'B3': 0.63
- 'B6': 0.61
- 'B2': 0.57



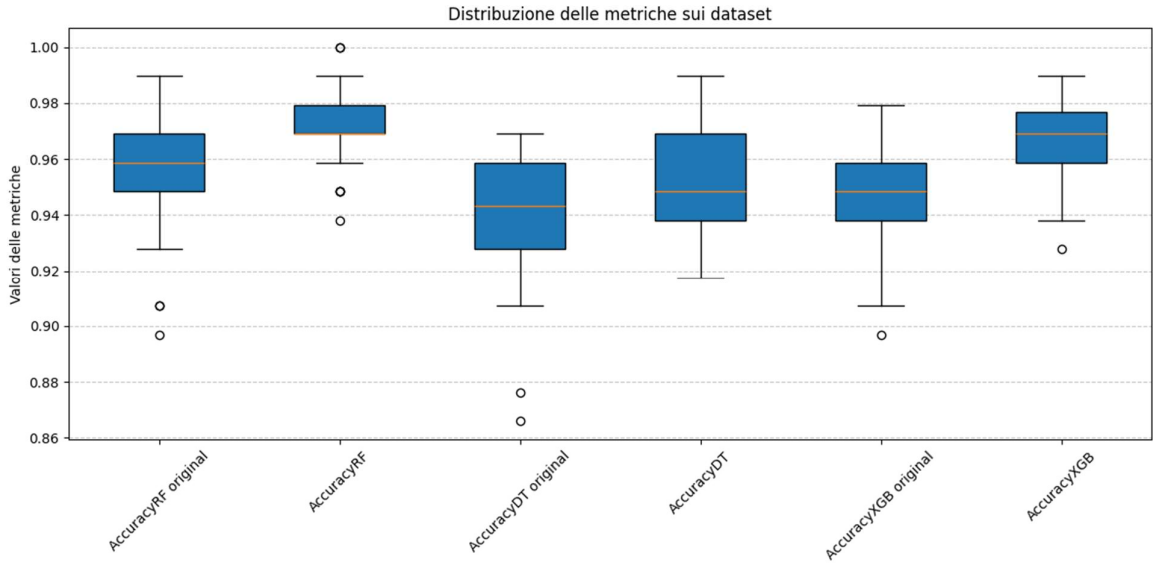
Risultati paper

La banda più popolare nei dataset IM-3, come nei risultati del paper, è stata la Near InfraRed (NIR, B4), che è apparsa nel 77% delle hyperfeatures; la popolarità di quest'ultima nella creazione di hyperfeatures può essere spiegata dalla sua importanza nella visualizzazione della vegetazione in buona salute, che è utile per distinguere tra i pixel appartenenti a *Dense Forest* e *Open Forest*.

In un secondo momento sono state valutate le prestazioni di tre modelli di classificazione (Random Forest, Decision Tree e XGBoost) sia sul dataset originale che sui 30 dataset ottenuti aumentando quello originale sulla base delle hyperfeatures del miglior modello di ciascuna run.

Come è possibile verificare dal confronto riportato dalla tabella e dalla visualizzazione tramite boxplot, l'utilizzo di M3GP ha portato ad un miglioramento delle prestazioni in tutti e tre i modelli di classificazione; in particolare, nei nostri risultati si ottiene un miglioramento notevole con Random Forest e XGB.

TEST ACCURACY	*Decision Tree*	*Random Forest*	*XGBoost*
Risultati Paper sul Dataset Originale	0.948	0.969	0.958
Nostri risultati sul Dataset Originale	0.937	0.954	0.945
Risultati Paper M3GP	0.958	0.969	0.969
Nostri risultati M3GP	0.951	0.972	0.964



Successivamente viene riportato il confronto tra l'accuracy media per classe usando XGBoost come classificatore; abbiamo calcolato anche l'accuracy per classe sui nuovi risultati ottenuti aggiungendo le hyperfeatures, notando anche in questo caso come ci sia stato un effettivo miglioramento delle prestazioni.

Accuracy media per classe	*Risultati paper di XGB original*	*Nostri risultati su XGB original*	*Nostri risultati su XGB con Hyperfeatures*
Dense Forest	92.67%	88.54%	96.17%
Open Forest	93.02%	93.51%	94.08%
Savanna Woodland	99.71%	99.84%	98.92%

Le hyperfeatures hanno migliorato la distinzione tra Dense Forest e Open Forest, ma hanno ridotto l'accuratezza per Savanna Woodland.

Si può infatti notare che Savanna Woodland aveva già un'accuratezza quasi perfetta; quindi, qualsiasi modifica su questa classe avrebbe probabilmente peggiorato i risultati. Alla fine, l'accuratezza tra le tre classi è diventata più bilanciata, anche se a scapito della classe Savanna Woodland.

Conclusioni paper “Improving Land Cover Classification Using Genetic Programming for Feature Construction”

Il paper evidenzia come combinare la programmazione genetica con algoritmi classici di Machine Learning possa rappresentare un approccio molto efficace per migliorare i risultati finali.

Grazie alla capacità della programmazione genetica di esplorare in maniera evolutiva lo spazio delle possibili trasformazioni dei dati, diventa possibile generare automaticamente nuove variabili, note come hyperfeatures, in grado di catturare relazioni non lineari e complesse tra le caratteristiche originali.

Queste nuove rappresentazioni, integrate negli algoritmi tradizionali, arricchiscono lo spazio delle feature e permettono al modello di apprendere in modo più profondo e accurato, contribuendo così a migliorare sia l’accuratezza che la capacità di generalizzazione.

È interessante notare che, nel confronto effettuato tra dataset binari e multiclasse, i miglioramenti sono stati evidenti solo per i problemi multiclasse, mentre nei dataset binari non si sono registrati incrementi significativi.

Infatti, è importante sottolineare che l’utilizzo di M3GP non sempre garantisce un aumento delle performance: in alcuni contesti l'introduzione di hyperfeatures può aumentare la complessità del modello senza tradursi in benefici reali, specialmente se le feature originali sono già ben strutturate e informative.

Applicazione dell'algoritmo M3GP su un *case study*: Naso elettronico per la classificazione binaria della presenza dei funghi nelle farine

Abbiamo applicato l'algoritmo M3GP, studiato in precedenza, ad un caso di classificazione binaria finalizzato a valutare la presenza di sostanze fungine in campioni di farina, rilevati tramite un sistema noto come "naso elettronico".

Il dataset, raccolto dallo stesso dispositivo, è strutturato in modo tale che per il nostro compito di classificazione siano state utilizzate solamente le colonne fino a quella denominata "class", omettendo le colonne successive non rilevanti per l'analisi.

Un aspetto strutturale importante è che i campioni sono organizzati in blocchi di tre righe; tale disposizione richiede particolare attenzione durante le operazioni di validazione incrociata (k-fold), in quanto è fondamentale mantenere intatti questi gruppi per garantire una corretta valutazione del modello. Inoltre, data la limitata quantità di dati disponibili, l'impiego della cross validation è stato indispensabile per rendere la valutazione del modello più robusta e affidabile.

Abbiamo utilizzato come metrica di fitness la Weighted Average of F-measures (WAF), che consente di valutare in maniera bilanciata le performance di classificazione considerando l'importanza di ciascuna classe.

Implementazione

In particolare, l'algoritmo M3GP è stato eseguito per 30 run indipendenti; in ogni run, il dataset è stato suddiviso in 3 fold e il modello è stato addestrato e valutato su ciascuno di essi. Successivamente, i risultati ottenuti in ciascun fold sono stati mediati, garantendo così una stima complessiva robusta e affidabile delle performance.

Una volta completate le 30 run, viene generato un file CSV che raccoglie tutte le metriche di valutazione ottenute, insieme ai modelli finali che contengono le hyperfeatures individuate in ciascuna run. Questi final model, che rappresentano le trasformazioni ottimizzate dei dati, vengono poi utilizzati per arricchire il dataset originale, dando origine a 30 nuovi dataset, ognuno basato sulle hyperfeatures ottenute dalla rispettiva run.

Questi successivi dataset verranno utilizzati per confrontare le prestazioni dei modelli di Machine Learning (Random Forest, Decision Tree e XGBoost).

In particolare, sul dataset originale sono state eseguite 30 run, ciascuna utilizzando una differente validazione incrociata a 3 fold per ciascuna run, per ottenere una valutazione robusta delle performance.

Parallelamente, sui dataset arricchiti con le hyperfeatures ottenute dalle 30 run, verrà effettuata una singola validazione incrociata a 3 fold. La media dei risultati ottenuti da entrambe le analisi verranno poi confrontati per valutare l'impatto delle hyperfeatures sul miglioramento delle performance.

Risultati

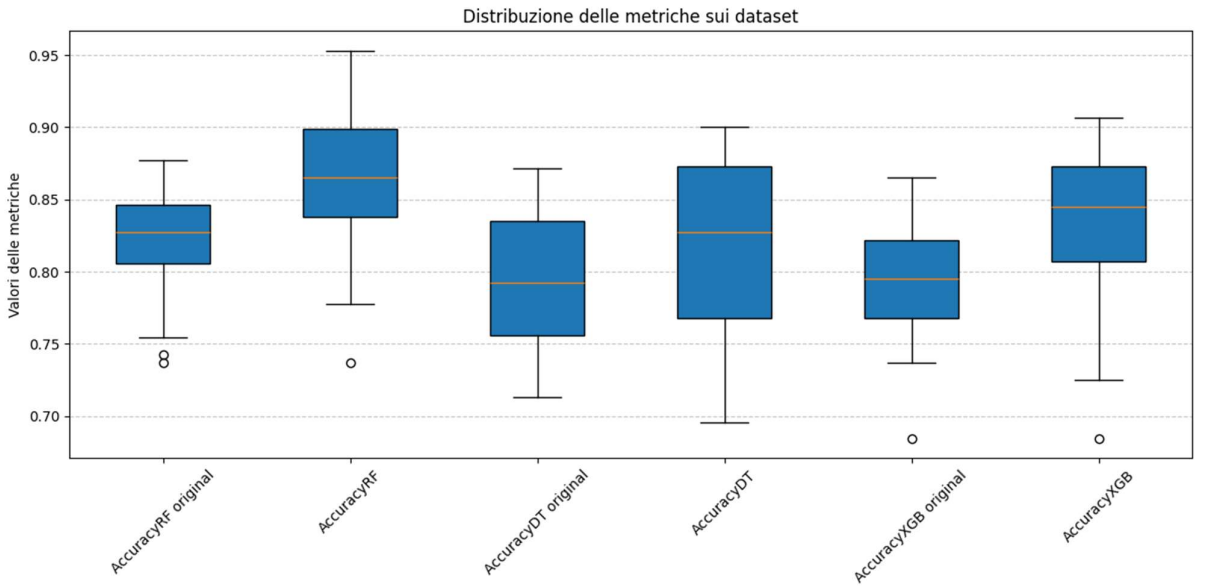
In questa fase viene usata M3GP per generare le hyperfeatures e sono stati registrati nella seguente tabella i valori di accuratezza che ha ottenuto sul dataset.

	Nostri risultati
Accuracy training	0.994
Accuracy test	0.885

In un secondo momento, sono state valutate le prestazioni di tre modelli di classificazione (Random Forest, Decision Tree e XGBoost) sia sul dataset originale che sui 30 dataset ottenuti aumentando quello originale sulla base delle hyperfeatures del miglior modello di ciascuna run.

Come è possibile verificare dal confronto riportato dalla tabella e dalla visualizzazione tramite boxplot, l'utilizzo di M3GP ha portato un buon miglioramento delle performance su tutti e tre i modelli di classificazione.

TEST ACCURACY	*Decision Tree*	*Random Forest*	*XGBoost*
Dataset Originale	0.791	0.822	0.795
Dataset + M3GP	0.817	0.865	0.828



Guida pratica di utilizzo del codice

Questa guida descrive passo-passo come impostare ed eseguire il codice M3GP per effettuare esperimenti sui dataset desiderati.

1. **Installazione progetto:** Scaricare e installare il nostro progetto dal repository GitHub:

   ```
   git clone https://github.com/azzielena/M3GP-Project.git
   cd Python-M3GP
   pip install -r requirements.txt
   ```

2. **Organizzazione Cartelle:** Il progetto è composto in due cartelle principali:

   - **`Python-M3GP`:** esegue M3GP sui dataset di interesse, generando un file CSV con le metriche calcolate e le migliori hyperfeatures.

   - **`Classification_with_hyperfeatures`:** per ciascun dataset, crea nuovi dataset arricchiti con le hyperfeatures e confronta i risultati della classificazione tra il dataset originale e quello aumentato, utilizzando tre modelli di Machine Learning (Random Forest, Decision Tree e XGBoost).

```
Python-M3GP/
│
├── datasets/
│   ├── nose.csv
│   ├── mcd3.csv
│   └── brazil2.csv
│
├── m3GP/
│   └── [vari file per eseguire m3GP]
│
├── results/
│   ├── m3gpWAF_brazil2.csv
│   ├── m3gpWAF_mcd3.csv
│   └── m3gpWAF_nose.csv
│
│
├── M3GP_naso_core.py
├── M3GP_RS_core.py
├── visual_results.py
└── Arguments.py
```

```
Classification_with_hyperfeatures/
│
├── tabelle_hyperfeatures/
│   ├── brazil2_WAF_hyper/
│   │   └── 30 dataset brazil2 aumentati con HF
│   ├── mcd3_WAF_hyper/
│   │   └── 30 dataset im3 aumentati con HF
│   └── nose_WAF_hyper/
│       └── 30 dataset nose aumentati con HF
│
│
├── creazioneDataset.py
├── nose_comparison_classificator.py
├── RS_comparison_classificator.py
└── risultati_metriche_nose.csv
```

Esecuzione Remote Sensing

3. **Entriamo nella cartella di interesse**:

   `cd Python-M3GP`

4. **Configurazione dei parametri**:
   Modificare il file `Arguments.py` per impostare i parametri desiderati. In particolare:
   - `DATASETS`: specificare il nome del dataset (`mcd3` o `brazil2`).
   - `OUTPUT_DIR`: indicare la cartella dove salvare i risultati.
   - `MODEL_NAME`: impostare il modello (default: `Mahalanobis Distance Classifier`).
   - `RUNS`: numero di esecuzioni (default: 30)
   - `MAX_GENERATION`: numero di generazioni (default: 50)
   - `FITNESS_TYPE`: scegliere il tipo di fitness (default: `Accuracy`, consigliato: `WAF` per migliori performance, come suggerito anche dal paper).
   - `RANDOM_STATE`: 0 per ottenere i nostri risultati (default: 42)

   Ulteriori parametri per la suddivisione del dataset o per l'esecuzione con determinate specifiche per la programmazione genetica possono essere personalizzati nello stesso file.

5. **Esecuzione del codice**
   Avviare il codice con il seguente comando : `python M3GP_RS_core.py`

6. **Risultati**
   Al termine delle 30 esecuzioni, verrà prodotto un file CSV contenente tutte le metriche calcolate che sarà salvato nella cartella `\results`.

Nel nostro esempio, i risultati ottenuti saranno:

- `m3gpWAF_brazil2.csv` per il dataset `brazil2`
- `m3gpWAF_mcd3.csv` per il dataset `mcd3`

I file CSV verranno salvati nella cartella indicata nel parametro `OUTPUT_DIR`.

7. **Visualizzazione dei risultati**
   Usando il file `visual_results.py` possiamo effettuare le analisi sulle metriche riportate nel file CSV di output, impostando il dataset di nostro interesse [righe 6,7,8].

8. Ora possiamo passare al confronto dei risultati tra dataset originale e aumentato.
   Creiamo i dataset aumentati nel seguente modo**:**

   ```
   cd classification_with_hyperfeatures
   ```

   Avviare il codice con il seguente comando, assicurandosi di specificare nel codice il dataset a cui si vuole far riferimento nei campi `dataset_file`, `hyperfeatures_file` e `output_folder` : `python creazioneDataset.py`
   I nuovi dataset saranno salvati nella cartella tabelle_hyperfeatures.

9. **Eseguiamo la classificazione con DT, RF e XGB e confrontiamo i risultati su dataset originale e aumentato:**

   ```
   python RS_comparison_classificator.py
   ```

   Come output verranno mostrati i risultati ottenuti da ciascun modello ML sia sul dataset originale sia sui dataset aumentati, riassumendo i risultati con la visualizzazione di un boxplot.

Esecuzione Nose

3. **Entriamo nella cartella di interesse**:

   ```
   cd Python-M3GP
   ```

4. **Configurazione dei parametri**:
   Modificare il file `Arguments.py` per impostare i parametri desiderati. In particolare:
   - `DATASETS`: specificare il nome del dataset `nose.csv`.
   - `OUTPUT_DIR`: indicare la cartella dove salvare i risultati.
   - `MODEL_NAME`: impostare il modello (default: `Mahalanobis Distance Classifier`).
   - `RUNS`: numero di esecuzioni (default: 30)
   - `MAX_GENERATION`: numero di generazioni (default: 50)
   - `FITNESS_TYPE`: scegliere il tipo di fitness (default: `Accuracy`, consigliato: `WAF` per migliori performance, come suggerito anche dal paper).
   - `RANDOM_STATE`: 0 per ottenere i nostri risultati (default: 42)

Ulteriori parametri per la suddivisione del dataset o per l'esecuzione con determinate specifiche per la programmazione genetica possono essere personalizzati nello stesso file.

5. **Esecuzione del codice**
   Avviare il codice con il seguente comando : `python M3GP_naso_core.py`

6. **Risultati**
   Al termine delle 30 esecuzioni, verrà prodotto un file CSV contenente tutte le metriche calcolate che sarà salvato nella cartella `\results`.
   Nel nostro esempio, il risultato ottenuto sarà:
   - `m3gpWAF_nose.csv`

   Il file CSV sarà salvato nella cartella indicata nel parametro `OUTPUT_DIR`.

7. **Visualizzazione dei risultati**
   Usando il file `visual_results.py` possiamo effettuare le analisi sulle metriche riportate nel file CSV di output, impostando il dataset di nostro interesse [righe 6,7,8] e visualizzando la media di training accuracy e test accuracy.

8. Ora possiamo passare al confronto dei risultati tra dataset originale e aumentato.
   Creiamo i dataset aumentati nel seguente modo**:**

   ```
   cd classification_with_hyperfeatures
   ```

   Avviare il codice con il seguente comando, assicurandosi di specificare nel codice il dataset a cui si vuole far riferimento nei campi `dataset_file`, `hyperfeatures_file` e `output_folder` : `python creazioneDataset.py`

   I nuovi dataset saranno salvati nella cartella tabelle_hyperfeatures.

9. **Eseguiamo la classificazione con DT, RF e XGB e confrontiamo i risultati su dataset originale e aumentato:**

   ```
   python nose_comparison_classificator.py
   ```

   Come output verranno mostrati i risultati ottenuti da ciascun modello ML sia sul dataset originale sia sui dataset aumentati, riassumendo i risultati con la visualizzazione di un boxplot.

   Verrà creato anche un file `risultati_metriche_nose.csv` in cui vengono riportati i risultati dell'accuracy per ciascun classificatore su dataset originale e dataset aumentato.