



UNIVERSITÀ  
DEGLI STUDI DI BARI  
ALDO MORO

PROGETTO DI INGEGNERIA DELLA  
CONOSCENZA

# Indice

- **Introduzione**
  - Gruppo
  - Spiegazione progetto e link (scelte progettuali)
  - Ambiente di sviluppo e tecnologie usate
- **I Modulo: Sistema esperto**
  - Introduzione
  - Base di conoscenza
    - Fatti
    - Regole
    - Induzione delle regole ILP (usata ricerca greedy hill climbing in algoritmo ILP)
  - Interfaccia Python
    - Dettagli tecnici
- **II Modulo: Machine Learning**
  - Dataset
  - Analisi esplorativa & preparazione dei dati
    - Data Cleaning
    - Prima Fase: riempimento dataset
    - Seconda Fase: trasformazione feature
  - Apprendimento supervisionato
    - Cross Validation
    - Modelli di Classificazione
      1. Decision Tree
      2. Naive Bayes
      3. Random Forest
      4. Gradient Boost
      5. Support Vector machine
  - Valutazione complessiva dei modelli

# Gruppo

Il progetto è stato realizzato da :

- Tommaso Perniola - MAT. 717766
- Daniele Cecca - MAT. 718588

# Introduzione Progetto e Link

Il progetto realizzato consiste in un sistema rivolto agli appassionati di **astronomia**, che permette di ottenere svariate informazioni riguardo gli esopianeti scoperti nelle varie missioni spaziali.

In particolare, il sistema si compone di due moduli:

1. Un sistema esperto, in Prolog
2. Un sistema di classificatori

## Ambiente di Sviluppo

Il progetto è stato sviluppato interamente attraverso i linguaggi:

- **Prolog**
- **Python**

Gli IDE utilizzati sono **Colab**, che sfrutta **Jupyter Notebook**, e **Visual Studio Code**, oltre al software **SWI-Prolog** per creare e testare la Knowledge Base.

Le librerie utilizzate sono le seguenti:

- per il sistema esperto in prolog:
  - *PySwip*, per interfacciarsi alla KB tramite Python
- per la batteria dei classificatori:
  - *sklearn*: per i modelli di apprendimento supervisionato
  - *pandas*: per I/O sul dataset
  - *matplotlib*, *seaborn*: per la realizzazione e visualizzazione di grafici
  - *imblearn*: per l'automatizzazione di processi tramite pipeline e per l'oversampling

Il progetto è inoltre supportato parzialmente via Web dalla libreria open-source **Streamlit**, che consente di avere una visione d'insieme dell'intero progetto.

# I Modulo : Sistema esperto

## Introduzione

Nelle seguenti sezioni verrà descritto il sistema esperto, con cui l'utente dovrà interagire ponendogli una serie di domande.

Inoltre il nostro sistema oltre a fornire informazioni sarà anche in grado di indurre le regole che determinano l'abitabilità di un pianeta a partire dai fatti e dalle regole presenti nella nostra KB.

## Base di Conoscenza

La **base di conoscenza (KB)** è stata interamente realizzata dal gruppo, facendo riferimento ai dati presenti nel dataset usato per la classificazione.

Sia le regole, sia i fatti sono stati realizzati seguendo la rappresentazione tripla, così chiamata perché tutte le rappresentazioni sono rappresentate come triple:

**prop(Individuo, Proprietà, Valore)**

- L' **INDIVIDUO** corrisponde al **SOGGETTO**
- La **PROPRIETÀ** corrisponde al **VERBO**
- Il **VALORE** corrisponde all' **OGGETTO**

Si è deciso di utilizzare questo tipo di rappresentazione, piuttosto che la classica rappresentazione prolog (es. relazione(a,b)) sia perché questo tipo di rappresentazione ci permette di interrogare la KB non solo sugli individui e sui valori, ma anche sulle proprietà;

sia perché ci permette di rappresentare un tipo/classe e una sottoclasse in maniera più semplice : es. prop (a , type, nomeClasse) oppure prop (a, is\_class\_nomeClasse, true)

## Fatti

Come anticipato precedentemente nella nostra KB avremo una serie di fatti e regole, quindi avremo rispettivamente della primitive knowledge e della derived knowledge, per un numero di campioni iniziale pari a 18.

Per quanto riguarda i **fatti** avremo 16 differenti proprietà, che riprendono, tra le numerose features del dataset, le features che forniscono le informazioni più importanti e più interessanti riguardo l'**abitabilità** e in più in generale riguardo gli esopianeti stessi;  
Inoltre questa scelta è stata dettata da una approfondita ricerca e da alcune direttive fornite dalla stessa NASA.

Di seguito proponiamo l'elenco delle proprietà realizzate con un relativo esempio:

1. <b>hostname</b>	prop(k2_139_b, hostname, k2_139)
2. <b>was_discovered_in</b>	prop(k2_139_b, was_discovered_in, 2017)
3. <b>has_radius</b>	prop(k2_139_b, has_radius, 9.09)
4. <b>has_mass</b>	prop(k2_139_b, has_mass, 121.14)
5. <b>has_density</b>	prop(k2_139_b, has_density, 0.16)
6. <b>has_gravity</b>	prop(k2_10_b, has_gravity, 2.23)
7. <b>has_temp</b>	prop(k2_139_b, has_temp, 508.0)
8. <b>has_composition</b>	prop(k2_139_b, has_composition, gas)
9. <b>has_atmosphere</b>	prop(k2_139_b, has_atmosphere, hydrogene_rich)
10. <b>has_eccentricity</b>	prop(k2_139_b, has_eccentricity, 0.12)
11. <b>has_orbit_period</b>	prop(k2_139_b, has_orbit_period, 28.38)
12. <b>distance_from_star</b>	prop(k2_139_b, distance_from_star, -2.37)
13. <b>has_stars_in_sys</b>	prop(k2_139_b, has_stars_in_sys, 1)
14. <b>his_star_has_met</b>	prop(k2_139_b, his_star_has_met, 0.23)
15. <b>his_star_has_temp</b>	prop(k2_18_b, his_star_has_temp, 3503)
16. <b>is_hab_class</b>	prop(k2_139_b, is_hab_class, non_habitable)

```

exo2.pl
File Edit Browse Compile Prolog Pce Help
exo2.pl
prop(hd_38283_b, has_density, 0.21).
prop(kepler_68_b, has_density, 0.49).
prop(wasp_80_b, has_density, 0.15).

%gravity in Earth Unit (9,807 m/sÂ²)
prop(k2_139_b, has_gravity, 1.46).
prop(k2_10_b, has_gravity, 2.23).
prop(k2_110_b, has_gravity , 2.49).
prop(k2_18_b, has_gravity, 3.28).
prop(k2_72_e, has_gravity, 1.39).
prop(wasp_118_b, has_gravity , 1.03).
prop(wasp_75_b, has_gravity, 2.08).
prop(wolf_503_b, has_gravity , 3.05).

```

(esempio fatti realizzati su prolog)

## Regole

Per quanto riguarda le **regole**, sono state realizzate 13 regole base di cui la maggior parte sono state create per definire delle classi e sottoclassi, seguendo le seguenti strutture:

**prop( P , is\_class, C)**

**prop(P, type, C) :-**  
**prop(S, subClassOf, C),**  
**prop(P, type, S).**

Di seguito proponiamo l'elenco delle proprietà realizzate con un relativo esempio:

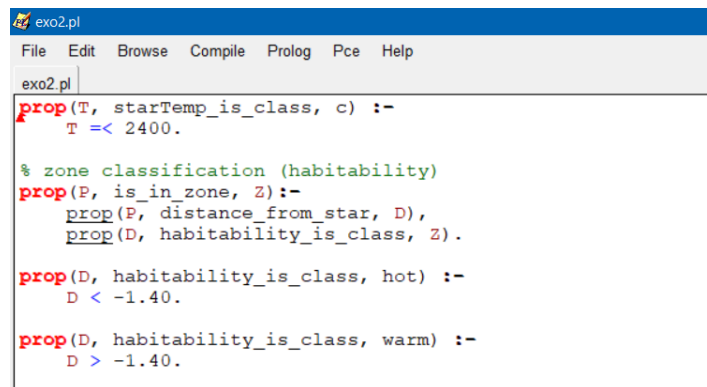
- |                                 |   |
|---------------------------------|---|
| <b>1. has_volume</b>            | prop(P, has_volume, V) :-<br>prop(P, has_density, D),<br>prop(P, has_mass, M),<br>V is M/D,<br>write('(mass is in Earth Unit)') |
| <b>2. his_star_is_class,</b>    | prop(P, his_star_is_class, C) :-<br>prop(P, his_star_has_temp, T),<br>prop(T, starTemp_is_class, C).                            |
| <b>3. starTemp_is_class</b>     | prop(T, starTemp_is_class, c) :-<br>T <= 2400   |
| <b>4. is_in_zone</b>            | prop(P, is_in_zone, Z):-<br>prop(P, distance_from_star, D),<br>prop(D, habitability_is_class, Z)                                |
| <b>5. habitability_is_class</b> | prop(D, habitability_is_class, warm) :-<br>D > -1.40  |
| <b>6. has_composition</b>       | prop(P, has_composition, rocky) :-<br>prop(P, has_composition, rocky_iron);<br>prop(P, has_composition, rocky_water)            |
| <b>7. massRadius_is_class</b>   | prop([M R], massRadius_is_class, jovian) :-<br>M > 50, M < 5000;<br>R > 3.5, R < 27   |
| <b>8. density_is_class, low</b> | prop(D, density_is_class, high) :-<br>D >= 2.   |
| <b>9. gravity_is_class</b>      | prop(G, gravity_is_class, weak) :-<br>G >= 1, G <= 1.5  |
| <b>10. temp_is_class</b>        | prop(ET, temp_is_class, too_high) :-  |

ET > 300, ET < 1000

**11. eccentricity\_is\_class**      `prop(ECC, eccentricity_is_class, hyperbolic) :-  
ECC > 1.`

**12. operiod\_\_is\_class**      `prop(OP, operiod__is_class, few) :-  
OP =< 10`

**13. metallicity\_is\_class**      `prop(MS, metallicity_is_class, low) :-  
MS =< 0.`



```
exo2.pl
File Edit Browse Compile Prolog Pce Help
exo2.pl
prop(T, starTemp_is_class, c) :-
    T =< 2400.

% zone classification (habitability)
prop(P, is_in_zone, Z):-
    prop(P, distance_from_star, D),
    prop(D, habitability_is_class, Z).

prop(D, habitability_is_class, hot) :-
    D < -1.40.

prop(D, habitability_is_class, warm) :-
    D > -1.40.
```

(esempio regole realizzati su prolog)

## Inductive Logic Programming

Il nostro progetto include un algoritmo ILP in grado di indurre, apprendendo dagli esempi in input, le regole di classificazione di abitabilità e non abitabilità.

Principalmente, l'algoritmo è composto da due funzioni:

- *apprendi(Classe)*
- *classifica(Oggetto, Classe)*

### ***apprendi(Classe):***

1. raccoglie tutti gli esempi in una lista
2. costruisce e manda in output una descrizione per la classe: è una lista di congiunzioni di coppie (attributo, valore)
3. asserisce (apprende) la corrispondente regola

In particolare, al suo interno chiama:

- *apprendi(Esempi, Classe, Descrizione)*: che crea la descrizione corrispondente alla classe in input in relazione agli esempi di training
- *assert(Classe←Descrizione)*: che associa alla classe la descrizione creata

**classifica(Oggetto, Classe):**

1. prende in input una lista di coppie (attr, val); un oggetto da classificare
2. ottiene la descrizione della classe (una lista di liste)
3. chiama *soddisfa(Oggetto, CongiunzioneAttributi)*: condizioni soddisfatte?
  - a. *CongiunzioneAttributi* è un singolo elemento di Descrizione, quindi una lista di coppie (attributo, valore)

Ora l'algoritmo procede a costruire la descrizione della classe, congiungendo varie coppie (attr, val) al fine di trovare la descrizione migliore (più **concisa**).

Dopo di ch , viene creata una lista di congiunzioni di coppie (attr, val) che copre almeno un esempio positivo per la classe di interesse e nessuno negativo, *Conge*. Questa lista viene usata per apprendere una possibile congiunzione di coppie (attr, val) per la classe di interesse, con la funzione **apprendi\_cong(Esempi, Classi, Conge)**:

1. sceglie una singola congiunzione nella lista: una *condizione*
2. **filtra** fra gli esempi di training che soddisfano quella condizione
3. restituisce la lista degli esempi filtrati

Per scegliere la condizione di *filtering* viene utilizzata un' **euristica**, per ovviare all'elevata combinatoriet  dell'algoritmo (complessit  esponenziale).

Di ci  si occupa **scegliCond(Esempi, Classe, AttVal)**:

1. cerca le coppie (attr, val) *candidate* e le ordina in base ad un *punteggio* che li viene assegnato
2. trova la coppia con il *miglior* punteggio: quale discrimina meglio?

Cosa fa l'euristica in questo caso?

- Determina ogni volta la miglior coppia (attributo, valore) da aggiungere alla congiunzione, rigettando le altre possibilit 

La funzione che la contiene   **punti(Esempi, Classe, AttVal, Punti)**, che sfrutter  una **RICERCA GREEDY (hill-climbing)**, ma che potrebbe portare all'incompletezza:

- si potrebbe perdere la descrizione pi  concisa

In particolare, *punti()* prende la coppia con maggior probabilit  di discriminare effettivamente gli esempi positivi da quelli negativi:

- $$\text{Punti} = \frac{\text{numero esempi positivi} + 1}{\text{Numero esempi totali} + 2}$$
- **NB:**   stata applicata una *correzione di Laplace*, dovuta a casi dove si hanno esempi di piccola cardinalit 



# Interfaccia Python

Il programma inizialmente presenta all'utente una lista di comandi che, sfruttando le regole prolog con cui ci si può interfacciare grazie alla libreria PySwip, permettono di:

1. Visualizzare tutti gli esopianeti nella KB
2. Visualizzare tutti gli esopianeti abitabili nella KB
3. Cercare un esopianeta o una sua caratteristica in particolare
4. Aggiungere un nuovo esopianeta
5. Classificare un nuovo esopianeta sfruttando l'algoritmo di ILP

```
Commands available:
1) Visualize exoplanets list
2) Visualize habitable exoplanets list
3) Search for an exoplanet
4) Add a new exoplanet
5) Classify an exoplanet: is it habitable or not?
```

in particolare:

- 1) richiama una regola prolog che, specificando la proprietà "hostname" (il nome della stella), è in grado di restituire tutti gli esopianeti, poichè ogni esopianeta sicuramente possiede un "hostname";
- 2) restituisce una lista di esopianeti specificando la proprietà "is\_hab\_class" che indica se un esopianeta è potenzialmente abitabile o no;

- 3) consente di:

```
> 2
List of all the habitable exoplanets:
> k2_18_b
> k2_72_e
> trappist_1_f
> trappist_1_g
> ross_128_b
> gj_180_c
> gj_422_b
> gj_667_Cc
> gj_180_c
> gj_422_b
> gj_667_Cc
> hd_283869_b
> kepler_61_b
> kepler_443_b
> wolf_1061_c
> gj_163_c
> kapteyn_b
> trappist_1_e
> kepler_1652_b
> kepler_442_b
> kepler_1229_b
> proxima_cen_b
```

```

Enter a number...
> 3

Press 'e' to escape
'd' to get a brief description of a chosen planet
'f' to get information about a feature of a chosen planet

> d

Enter exoplanet name: proxima_cen_b

proxima_cen_b is a habitable exoplanet discovered in 2016 and it is in a planetary system whose star is proxima_cen.
It is a subterranean-like planet.

```

- d) ottenere una breve descrizione di un esopianeta scelto;
  - f) ricavare il valore di una specifica feature di un esopianeta scelto;
- 4) aggiunge un nuovo esopianeta forzando l'inserimento di tutte le feature necessarie a far funzionare correttamente la KB:
- a) se si è conoscenza della potenziale abitabilità dell' esopianeta, viene chiesto di immetterla in input;
  - b) altrimenti, è possibile sfruttare la regola di abitabilità appresa dai fatti e classificare il pianeta in quel momento;
- 5) classifica un esopianeta in "habitable" o "non habitable" in base ad una regola indotta dagli esempi (i fatti) presenti nella KB [vedere sezione precedente ILP]

```

Enter a number...
> 5

Enter an example to classify:
> enter radius: 2
> enter mass: 3
> enter density: 2
> enter gravity: 3
> enter equilibrium temperature: 340
> enter composition: gas
> enter atmosphere: metals_rich
> enter eccentricity: 1
> enter orbit period: 8.4
> enter zone distance: -0.2
> enter number of stars: 1
> enter star metallicity: 0.02
> enter star temperature: 7800

The exoplanet entered belongs to the class: non_habitable

```

## Dettagli tecnici

Mentre i dettagli tecnici dell'algoritmo di Induzione Logica delle regole sono stati discussi precedentemente, ora vorremmo soffermarci, per completezza, su alcuni espedienti e scelte progettuali attuate nel nostro programma **Python**:

Viene utilizzata una funzione di **inizializzazione** in cui vengono chiamate le seguenti funzioni:

- *listProp()*: conversione dei fatti
- *learn()*: apprendimento delle regole

```
def initialization(planetList):  
    pl.listProp(planetList) #converte i fatti  
    pl.learn()
```

In particolare, *listProp()*:

→ è una funzione necessaria per far funzionare l'algoritmo di Induzione, perchè esso esige esempi nel formato:

◆ *esempio(class, [feature<sub>1</sub> = value, feature<sub>2</sub> = value, ...])*

→ ma tutti i fatti nella KB sono, come detto precedentemente, nella notazione tripla:

◆ *prop(individuo, property, value)*

A tal proposito, si è scelto di **convertire** a run-time tutti i fatti presenti nella KB nella notazione di cui necessita l'algoritmo di ILP, e memorizzandoli temporaneamente nella KB avvalendoci della funzione *assertz()* di Prolog.

Tutto ciò al fine di avere una base di conoscenza più **coerente** dal punto di vista della notazione utilizzata.

Inoltre, prendendo in input una lista, *planetList*, output della funzione *getExoplanets()*, che restituisce una lista di tutti gli esopianeti aggiornati fino a quel momento, *listProp* è in grado di convertire anche i **nuovi** eventuali esopianeti inseriti dall'utente e renderli parte attiva dell'algoritmo di Induzione.

Infine, il processo di conversione si compone di essenzialmente due parti:

1. classificazione delle feature
  - a. passo fondamentale perché l'algoritmo di Induzione richiede valori **categorici o discreti** e non continui

- i. tramite delle regole prolog siamo stati in grado di classificare tutti i valori di ciascuna feature di ogni esopianeta, rifacendoci, quando possibile, a classificazioni esistenti
2. “composizione” dell’ esempio da asserire

## Il Modulo: Sistema di classificazione

### Introduzione

Nelle seguenti sezioni, verrà descritto l’intero processo di apprendimento della classe di abitabilità degli esopianeti; a partire dall’ Analisi Esplorativa & Preparazione dei Dati

fino ad arrivare all’ utilizzo di diversi modelli di classificazione e al loro successivo confronto.

In particolare i modelli valutati saranno i seguenti:

- **Decision**
- **Naive Bayes**
- **Random Forest**
- **Gradiend Boost**
- **Support vector machine**

### Dataset

Il dataset utilizzato è il [PHL-EC](#), che contiene dati provenienti dal [NASA Exoplanet Archive](#), ma organizzati in maniera diversa, indicando diverse caratteristiche che potrebbero rendere tali esopianeti potenzialmente abitabili.

Il nostro dataset contiene inizialmente 3876 righe (esopianeti) e 68 features, tra cui 13 categoriche e 55 continue.

In particolare la feature target di nostro interesse è **P. Habitable Class**, che individua le 5 classe di abitabilità:

1. **non-habitable**
2. **mesoplanet**
3. **psychroplanet**

#### 4. hypopsychroplanet

#### 5. thermoplanet

Di seguito riportiamo le feature più rilevanti:

Nome	Significato	Tipo	Nome	Significato	Tipo
P. Name	nome dell'esopianeta	categorica	P. SFlux Min P. SFlux Mean P. SFlux Max (EU)	<a href="#">flusso</a>	continua
P. Composition Class	classe di composizione e dell'esopianeta	categorica	P. Teq Min P. Teq Mean P. Teq Max (K)	temperatura di equilibrio	continua
P. Atmosphere Class	classe dell'atmosfera presente sull'esopianeta	categorica	P. Mag	<a href="#">magnitudine</a>	continua
P. Zone Class	classe di distanza dalla stella	categorica	P. Period (days)	periodo orbitale misurato in giorni terrestri	continua
P. Mass Class	classe derivante da raggio/massa	categorica	P. Eccentricity	<a href="#">Eccentricità</a> dell'orbita dell'esopianeta	continua
P. Habitable Class	indica la classe di abitabilità	categorica	P. Inclination (deg)	inclinazione dell'orbita dell'esopianeta misurata in gradi	continua
P. Min Mass P. Mass P. Max Mass (EU)	massa dell'esopianeta in Earth Unit ( $5.97 \times 10^{24}$ kg)	continua	S. Name	stella attorno a cui orbita l'esopianeta	categorica
P. Radius (EU)	raggio in Earth Unit ( $6.37 \times 10^6$ m)	continua	S. Constellation	nome della costellazione in cui si trova la stella	categorica
P. Density (EU)	densità in E.U. (5515)	continua	S. Type	Tipo della stella	categorica

	kg/m <sup>3</sup> )			(include classe di temperatura e magnitudine)	
P. Gravity (EU)	gravità in E.U. (9.81 m/s <sup>2</sup> )	continua	S. Mass (SU)	massa stellare	continua
P. Esc Vel (EU)	<a href="#">velocità di fuga</a> dell'esopianeta	continua	S. Radius (SU)	raggio stellare	continua
S. Teff (K)	temperatura effettiva	continua	S. DEC (deg)	declinazione della stella	continua
S. Luminosity (SU)	<a href="#">luminosità</a> espressa in Solar Units (3.828×10 <sup>26</sup> W)	continua	S. No. Planets	numero di pianeti nel sistema	discreta
P. Appar Size (deg)	dimensione apparente da una certa <a href="#">angolazione</a>		P. HZD	habitability zone distance	continua
P. Disc. Year	anno di scoperta dell'esopianeta	discreta	P. HZC	<a href="#">habitability zone composition</a>	continua
P. ESI	Indice di similarità con la Terra	continua	P. HZA	habitable zone atmosphere	continua
P. Surf Press (EU)	Pressione superficiale in Earth Unit (1014 bar)	continua	P. HZI	habitable zone index	continua

# Analisi Esplorativa & Preparazione dei Dati

## Data Cleaning

Inizialmente ho applicato alcune operazioni di **dataset cleaning**.

Come prima operazione eliminiamo gli esempi che hanno come classi di abitabilità **hypopsychroplanet**, **thermoplanet**, essendo gli rispettivamente 3 per ogni classe. Dopo l'eliminazione delle classi con pochi esempi avremo pertanto le seguenti 3 classi di abitabilità:

1. **non-habitable**
2. **mesoplanet**
3. **psychroplanet**

```
non-habitable    3820
mesoplanet       31
psychroplanet    18
hypopsychroplanet 3
thermoplanet     3
Name: P. Habitable Class, dtype: int64
```

```
non-habitable    3820
mesoplanet       31
psychroplanet    18
Name: P. Habitable Class, dtype: int64
```

Continuiamo rimuovendo una serie di features per i quali si hanno pochi esempi, in modo tale da ridurre la dimensionalità del features space.

Oltre queste features rimuovo anche una serie di features che non hanno valenza da un punto di vista fisico per il task di classificazione che stiamo realizzando; infatti molte delle features che stiamo rimuovendo sono incertezze su altre features date.

Le features rimosse sono le seguenti:

```
cols_to_drop = ['P. Name', 'S. Constellation', 'S. Type', 'P. Int ESI',
                'P. Surf ESI', 'P. Disc. Method', 'P. Disc. Year', 'P.
Hab Moon', 'P. SFlux Min (EU)', 'P. SFlux Max (EU)',
                'P. Teq Min (K)', 'P. Teq Max (K)', 'P. SFlux Mean
(EU)', 'S. Name', 'S. Hab Zone Min (AU)', 'S. Hab Zone Max (AU)']
```

Mentre le features rimaste saranno le seguenti:

```
['P. Zone Class', 'P. Mass Class', 'P. Composition Class', 'P. Atmosphere
Class', 'P. Habitable Class', 'P. Mass (EU)', 'P. Radius (EU)', 'P. Density
```

```
(EU)', 'P. Gravity (EU)', 'P. Esc Vel (EU)', 'P. Teq Mean (K)', 'P. Surf Press (EU)', 'P. Mag', 'P. Appar Size (deg)', 'P. Period (days)', 'P. Sem Major Axis (AU)', 'P. Eccentricity', 'P. Mean Distance (AU)', 'P. Omega (deg)', 'S. Mass (SU)', 'S. Radius (SU)', 'S. Teff (K)', 'S. Luminosity (SU)', 'S. RA (hrs)', 'S. DEC (deg)', 'S. Mag from Planet', 'S. Size from Planet (deg)', 'S. No. Planets', 'S. No. Planets HZ', 'P. HZD', 'P. HZC', 'P. HZA', 'P. HZI', 'P. ESI', 'S. HabCat', 'P. Habitable', 'P. Confirmed']
```

Arrivato a questo punto per determinare se eliminare altre features osserviamo la correlazione tra le features a due a due , e per farlo sfruttiamo [la correlazione di Pearson](#) .

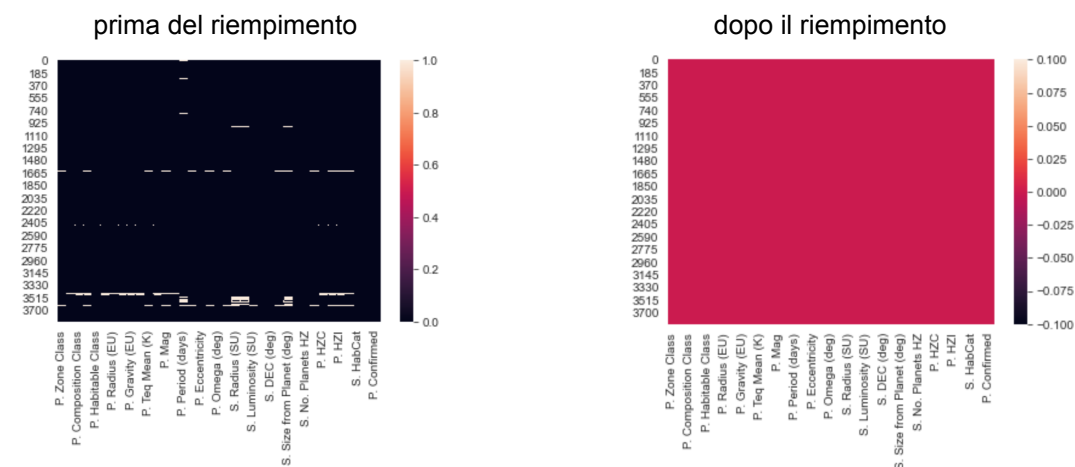
Prima di poter calcolare la correlazione devo eseguire i seguenti passaggi:

1. riempire il dataset con dati mancanti
2. trasformare le feature categoriche in features numeriche

## Prima Fase: riempire il dataset con dati mancanti

Per le features numeriche adottiamo il **simple Imputer di sklearn** che permette di riempire il valore mancante con il valore medio all'interno della colonna, specificando come strategia **mean**.

Menter per le features categoriche adottiamo il **simple Imputer di sklearn** che permette di riempire il valore mancante con quello più ricorrente all'interno della colonna, specificando come strategia **most frequent**.



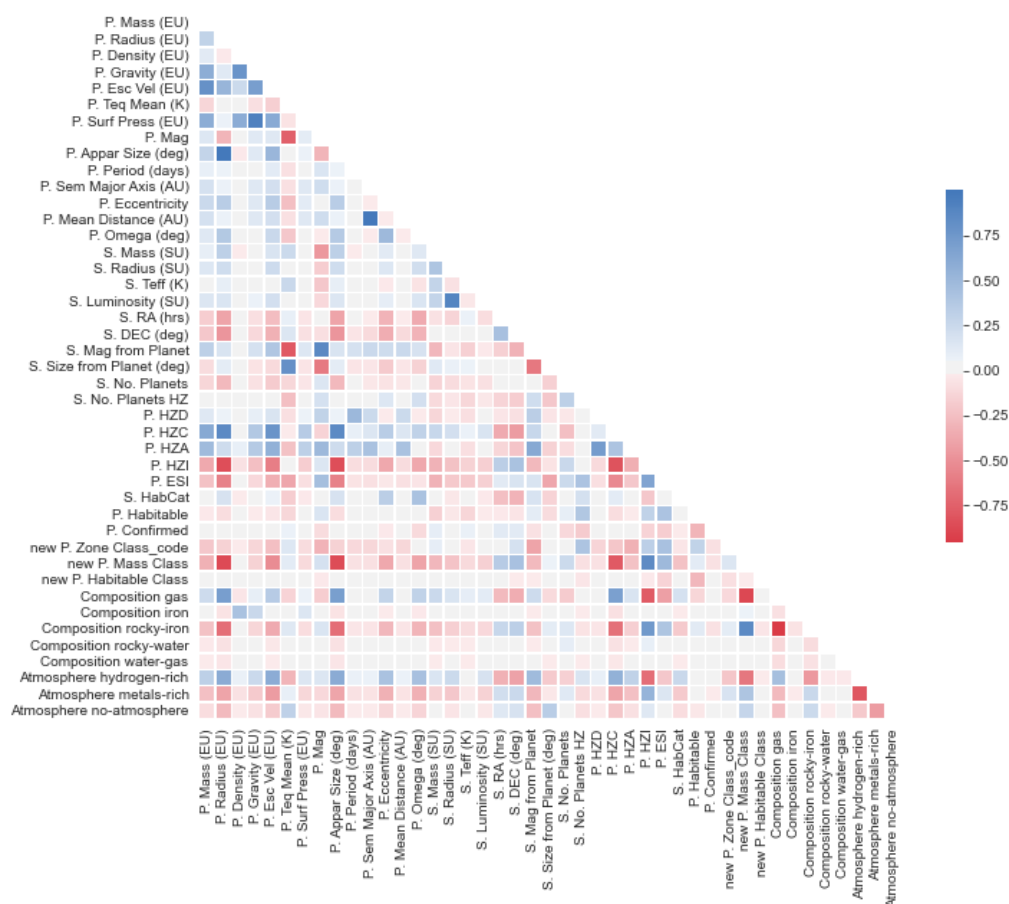


## Seconda Fase: trasformare le feature categoriche in features numeriche

Le seguenti tre features **P. Zone Class**, **P. Mass Class**, **new P. Habitable Class** verranno trasformate utilizzando il **label encoding**, essendoci effettivamente un ordine tra i values le features.

Mentre le altre due features **P. Composition Class**, **P. Atmosphere Class** verranno trasformate utilizzando **\*\*l'hot encoder\*\*** che crea nuove features booleane a partire dai values delle features, ossia crea le **\*variabili indicatrici\***

Dopo queste due fasi possiamo finalmente osservare il risultato del calcolo della correlazione



Si può notare come ci siano delle features altamente correlate che possono essere rimosse:

- P. Mass Class ---> P. Radius (EU) and P. Mass (EU)
- P. Appar Size ---> P. Radius (EU)
- P. Surf Press ---> Gravity
- S. Mag from Planet ---> P. Mag
- P. Sem Major Axis(AU) ---> P. Mean Distance(AU)
- S. Size from Planet(deg) -->S. Mag from Planet
- P. HZC --> P.Composition Class
- P. HZA --> P Atmosphere Class
- P. HZI --> HZD and HZC and HZA

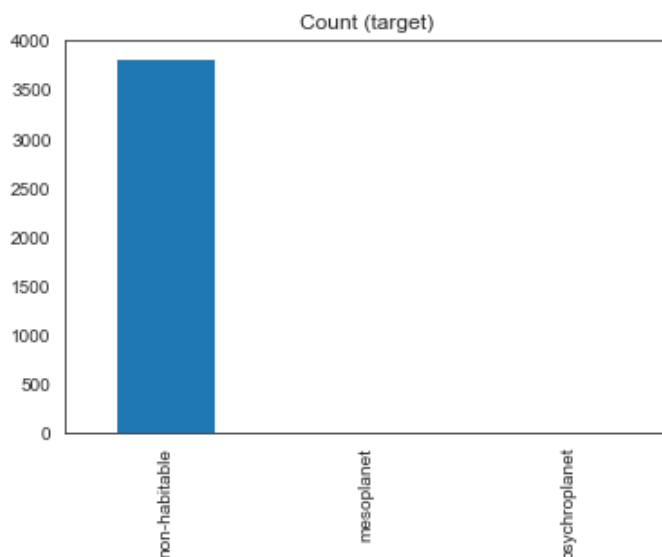
Inoltre sebbene non sia altamente evidenziata la seguente correlazione, la rimuoviamo perchè sono correlate da un punto di vista fisico:

- P. Zone Class ---> HZD

La veridicità di tutte le precedenti correlazioni è state inoltre confermata da ricerche esplorative sul web.

## Divisione dei dati

Dopo questa prima fase di pulizia del dataset analizziamo i dati rimasti. Possiamo osservare come il nostro dataset sia molto sbilanciato.



**non-habitable:** 3820  
**mesoplanet:** 31  
**psychroplanet :** 18  
**Percentage of non-habitable:** 98.72  
**Percentage of mesoplanet:** 0.8  
**Percentage of psychroplanet:** 0.47

Arrivati a questo punto effettuiamo la divisione del nostro set di dati in set di training e set di test, andando a comporre rispettivamente il nostro set di training con l'80% dei dati, mentre il nostro set di test con il 20% dei dati. Per evitare che nel nostro set di test ci siano solo pianeti non abitabili non splittiamo in maniera casuale, ma usiamo il parametro **stratify** nella funzione di splitting, che ci permette di mantenere le proporzioni del vettore assegnato, in questo caso il vettore è la colonna delle feature target.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=11)
```

# Apprendimento Supervisionato

## Cross Validation

Come detto precedentemente il nostro dataset è molto sbilanciato, pertanto nella fase precedente alla fase di test vera e propria (nella fase di validation) adotteremo la tecnica di oversampling **SMOTE** sul set di training, la quale permette di creare una serie di esempi delle classi minoritarie in modo tale da bilanciare la classe maggioritaria.

L'oversampling viene effettuato solamente sul set di training e non sul set di test, poiché creare degli esempi sintetici andrebbe ad intaccare la veridicità dei dati e delle valutazioni successive, sebbene nel nostro caso il set di test sia composto per lo più da pianeti non abitabili.

Come accennato precedentemente prima della fase di test, per ogni classificatore adotteremo lo **stratify k-fold cross** validation per poter ottenere il miglior modello. Infatti durante la fase di cross validation i dati del set di training sono splittati in k fold, dove ogni fold viene usato come set di validation mentre gli altri k-1 come set di training, e il tutto viene ripetuto k volte.

In questo modo è possibile ottimizzare i parametri di ogni modello, e come vedremo grazie all'utilizzo di particolari funzioni siamo stati in grado di ottimizzare i modelli di classificazione anche sulla base di alcuni **iper-parametri**, che come vedremo saranno differenti da modello a modello.

Nel nostro caso abbiamo scelto  $k = 3$  semplicemente per i limiti di potenza computazionale imposti dai nostri calcolatori.

Per individuare il modello migliore abbiamo utilizzato la funzione **GridSearchCV**, che come anticipato, ci permette di provare diversi iperparametri durante la cross validation, inoltre poichè vogliamo adottare l'oversampling durante la cross validation, passiamo come **estimator**, ossia come modello, non un modello ma una **Pipeline**, che ci permette di definire una catena di lavoro, ossia una serie di azioni che verranno svolte in maniera consecuziale.

Nel nostro caso la pipeline si presenta nel seguente modo:

```
pipeline = imbpipeline(steps = [['smote', SMOTE(random_state=11)],  
                                ['scaler', MinMaxScaler()],  
                                ['classifier', ModelloDiClassif]])
```

L'utilizzo dello **stratify k-fold cross** ha portato però anche delle limitazioni; difatti questo tipo di cross non funziona con le features target multi-label ossia con le variabili indicatrici ma accetta solamente un vettore unidimensionale.

Pertanto, come detto precedentemente, si è deciso di optare per il label encoding. Così facendo però ci siamo preclusi la possibilità di plottare curve Precision-Recall e ROC, funzionando questo tipo di funzioni solamente con classi multilabel per quanto riguarda la classificazione a più label.

Per trovare il modello migliore dobbiamo specificare una metriche di valutazione per i vari modelli, nel nostro caso abbiamo utilizzato l'**accuracy**, poichè a differenza del nostro set iniziale il set utilizzato non è più sbilanciato e quindi ha senso utilizzare l'accuracy, la quale calcola la porzione dei dati predetti in maniera corretta su tutti i dati predetti:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Mentre per la valutazione del modello sul set di test abbiamo deciso di adottare come metriche di valutazione le seguenti metriche:

- **precision**
  - La precisione è l'accuratezza con cui il sistema di machine Learning prevede le classi positive
  - E' definito come il rapporto fra i *true positive* e la somma dei *true positive* e *false positive*
- **recall**

- Il *recall*, o *true positive rate*, indica il rapporto di istanze positive correttamente individuate dal sistema di machine learning.
- **f1-score**
  - l' F1 score è la media armonica della precision e della recall, utile per avere una visione complessiva delle due metriche
- **matthews correlation coefficient**
  - Il coefficiente di correlazione Matthews è una misura bilanciata di accuratezza, che può essere usata anche se una classe ha molti più campioni di un altro, come nel nostro caso
- **geometric mean score**
  - Il G-mean è una metrica che misura il bilanciamento fra le performance di classificazione sia sulle classi maggioritarie (non habitable, nel nostro caso) che minoritarie

Essendo queste le metriche più adatte per la classificazione di dataset sbilanciati, inoltre nel calcolare ognuna di queste metriche è stato utilizzato l'attributo **macro**, il quale permette di calcolare le metriche per ciascuna label/classe e trova la loro media non ponderata. Cioè non tiene conto dello squilibrio della classe.

## Modelli di Classificazione

Si è deciso di utilizzare diversi modelli di classificazione, dalla complessità crescente, in modo tale da poter osservare il funzionamento della maggior parte delle tipologie di classificatori affrontati a lezioni e per poter osservare come la bontà di un modello non sia strettamente legata alla sua complessità. Inoltre questi classificatori sono tutti suggeriti per una ottimale **classificazione multiclasse**.

## Decision Tree

**L'albero di decisione** è un albero che divide l'insieme degli esempi ricorsivamente finchè non si ottengono solo foglie (tutti gli esempi classificati).

Il nodo radice rappresenta l'intero dataset; I nodi di decisione rappresentano le feature del dataset; le ramificazioni indicano le regole di decisione.

E' stato scelto questo modello essendo un modello versatile ,consentendo di lavorare facilmente sia con valori numerici che categorici, oltre ad essere estremamente comprensibile e rendere immediata l'interpretazione dei dati.Inoltre un albero decisionale lavora naturalmente con un multi-class target.

Durante la fase di validation sono stati provati, come detto precedentemente, una serie di **iper-parametri**;

Gli iper-parametri che sono stati scelti per ottimizzare il modello , con i rispettivi values da provare,sono i seguenti:

- **classifier\_\_criterion:** [gini, entropy]
- **classifier\_\_max\_depth:** range(1,10)
- **classifier\_\_min\_samples\_leaf:** range(2,10)
- **classifier\_\_min\_samples\_split:** range(1,5)

In seguito alla cross validation siamo giunti alla conclusione che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

- classifier\_\_criterion: **gini**
- classifier\_\_max\_depth: **4**
- classifier\_\_min\_samples\_leaf: **1**
- classifier\_\_min\_samples\_split: **4**

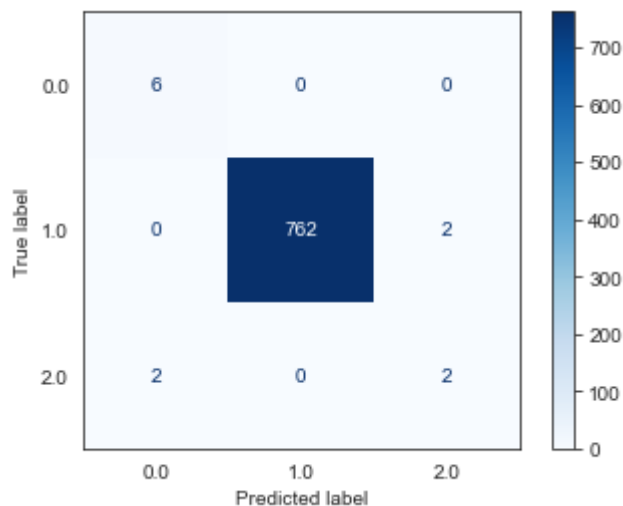
Pertanto come si evince dalla teoria è stato scelto come modello ottimale un modello piccolo, in termini di profondità e in termini di nodi.

I risultati della successiva fase di test sono descritti dal risultato delle seguenti metriche:

- **accuracy:** 0.9948320413436692
- **precision:** 0.75
- **recall:** 0.8324607329842931
- **f1:** 0.7852774137177807
- **matthews\_corrcoef:** 0.8197002684542365
- **geometric\_mean\_score:** 0.9116015388298121

Si noti come, come anticipato prima l'accuracy non sia una buona metriche nel nostro caso.

Di seguito rappresentiamo la **matrice di confusione** che ci permette di avere una panoramica completa dei risultati ottenuti:



## Naive Bayes

**Naive Bayes** è un algoritmo per risolvere problemi di classificazione e apprendimento automatico (machine learning) che utilizza il teorema di Bayes, il quale permette di calcolare per ogni istanza la probabilità di appartenenza a una classe.

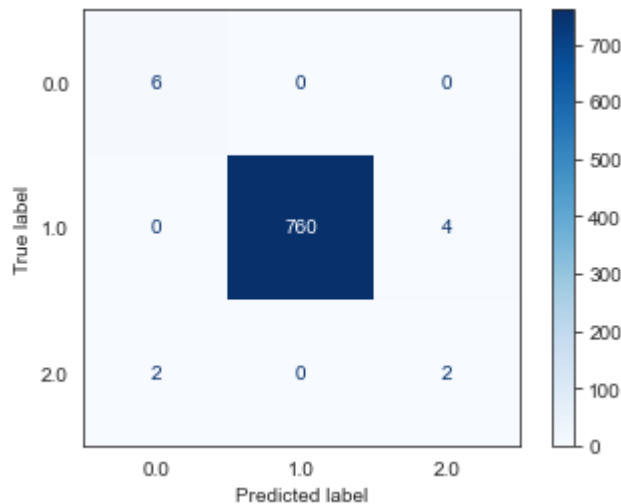
Abbiamo deciso di utilizzarlo per la sua capacità di trattare bene i dati mancanti, oltre al fatto che richiede un numero ridotto di esempi di training per stimare i parametri necessari alla classificazione.

A differenza degli altri modelli usati nel modello naive-Bayes non ci sono iper-parametri, pertanto la griglia dei parametri sarà vuota.

I risultati della successiva fase di test sono descritti dal risultato delle seguenti metriche:

- **accuracy:** 0.9922480620155039
- **precision:** 0.6944444444444445
- **recall:** 0.831588132635253
- **f1:** 0.7515060617422823
- **matthews\_corrcoef:** 0.7579632427016252
- **geometric\_mean\_score:** 0.9107284383867313

Di seguito rappresentiamo la **matrice di confusione** che ci permette di avere una panoramica completa dei risultati ottenuti:



## Random Forest

Il modello random forest è un modello di classificazione all'interno del quale sono usati diversi alberi di classificazione, per poi fare una media tra i risultati ottenuti dai vari alberi, oppure è usato il meccanismo del voto

E' stato scelto questo modello, essendo un modello migliore di un semplice albero di decisione, poiché un albero di decisione non è un ottimo predictor a causa della sua imprecisione nel classificare nuovi esempi (non di training, difatti si potrebbe verificare più facilmente l'overfitting). Ciò che rende effettivamente migliore questo modello è la varietà della selezione delle features di input(set di input), le quali vengono selezionate in maniera randomica per ogni albero, portando a predizioni diverse fra loro.

Durante la fase di validation sono stati provati, come detto precedentemente, una serie di iper-parametri;

**Gli iper-parametri** che sono stati scelti per ottimizzare il modello , con i rispettivi values da provare,sono i seguenti:

- **n\_estimators:** [50, 100, 200]
- **max\_depth:** [4, 6, 10, 12]

in seguito alla cross.validation siamo giunti che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

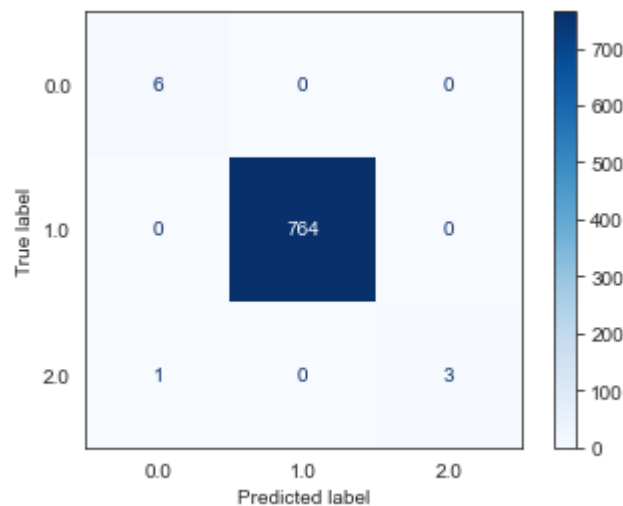
- **n\_estimators:** **50**
- **max\_depth:** **12**



I risultati della successiva fase di test sono descritti dal risultato delle seguenti metriche:

- **accuracy:** 0.9987080103359173
- **precision:** 0.9523809523809524
- **recall:** 0.9166666666666666
- **f1:** 0.9267399267399267
- **matthews\_corrcoef:** 0.9495595614242498
- **geometric\_mean\_score:** 0.9572193102264341

Di seguito rappresentiamo la **matrice di confusione** che ci permette di avere una panoramica completa dei risultati ottenuti:



## Gradient Boost

Sono un gruppo di algoritmi di apprendimento automatico che combinano insieme molti **modelli di apprendimento debole** (base-level algorithms) per creare un modello predittivo forte. Solitamente sono utilizzati gli alberi decisionali.

I modelli di boosting sono diventati popolari grazie alla loro efficacia nella classificazione di set di dati complessi, e per il medesimo motivo è stato scelto questo modello, come rappresentativo della categoria dei modelli ensemble

Durante la fase di validation sono stati provati, come detto precedentemente, una serie di **iper-parametri**;

Gli iper-parametri che sono stati scelti per ottimizzare il modello , con i rispettivi values da provare,sono i seguenti:

- **max\_depth:** range (2, 10, 1)
- **n\_estimators:** range(60, 220, 40)
- **learning\_rate:** [0.1, 0.01, 0.05]

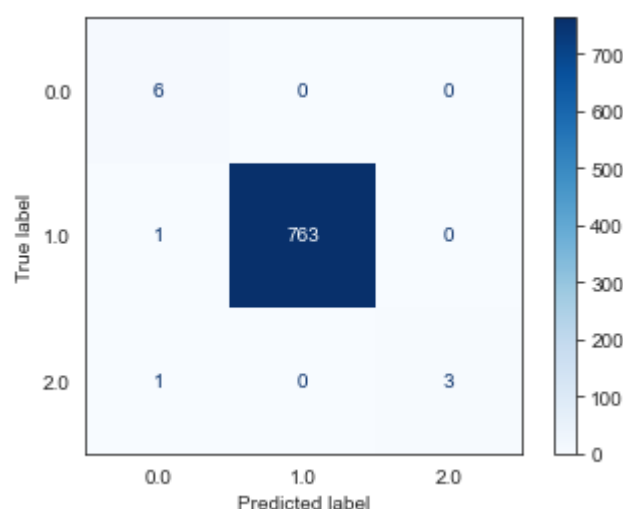
In seguito alla cross.validation siamo giunti che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

- **max\_depth:** range 4
- **n\_estimators:** range 100
- **learning\_rate:** 0.05

I risultati della successiva fase di test sono descritti dal risultato delle seguenti metriche:

- **accuracy:** 0.9974160206718347
- **precision:** 0.9166666666666666
- **recall:** 0.9162303664921465
- **f1:** 0.904543611812767
- **matthews\_corrcoef:** 0.9049164998345826
- **geometric\_mean\_score:** 0.9567836890501136

Di seguito rappresentiamo la **matrice di confusione** che ci permette di avere una panoramica completa dei risultati ottenuti:



# Support Vector Machine

Gli **SVM** sono modelli di classificazione il cui obiettivo è quello di trovare la retta di separazione delle classi che massimizza il *margin* tra le classi stesse, dove con *margin* si intende la distanza minima dalla retta ai punti delle due classi.

La SVM ammette la *misclassification*, ovvero l'errata classificazione di particolari esempi di training, gli *outliers*. Infatti la SVM cerca gli esempi più difficili, quelli che tendono ad essere più vicini all'altra classe (i vettori di supporto, appunto) e considera solo quelli per eseguire la classificazione, e ciò ci torna molto utile considerato il nostro dominio.

Nonostante la SVM supporti nativamente la classificazione binaria, è possibile applicarla ad una classificazione multi-layer come la nostra, ottenendo comunque buoni risultati.

Durante la fase di validation sono stati provati, come detto precedentemente, una serie di iper-parametri;

Gli iper-parametri che sono stati scelti per ottimizzare il modello, con i rispettivi values da provare, sono i seguenti:

- C: [0.1, 1, 10, 100, 1000]
- gamma: [1, 0.1, 0.01, 0.001, 0.0001]
- kernel: [rbf]

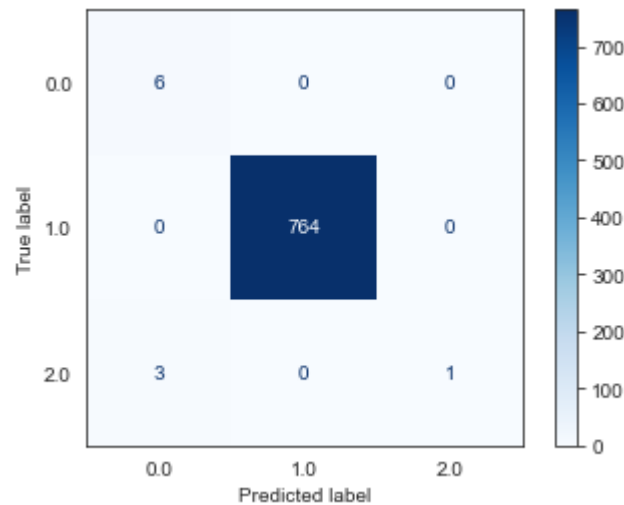
In seguito alla cross.validation siamo giunti che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

- C: 0.1
- gamma: 0.1
- kernel: rbf

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- accuracy: 0.9961240310077519
- precision: 0.8888888888888888
- recall: 0.75
- f1: 0.7333333333333334
- matthews\_corrcoef: 0.8489522782601875
- geometric\_mean\_score: 0.8654614015078893

Di seguito rappresentiamo la **matrice di confusione** che ci permette di avere una panoramica completa dei risultati ottenuti:

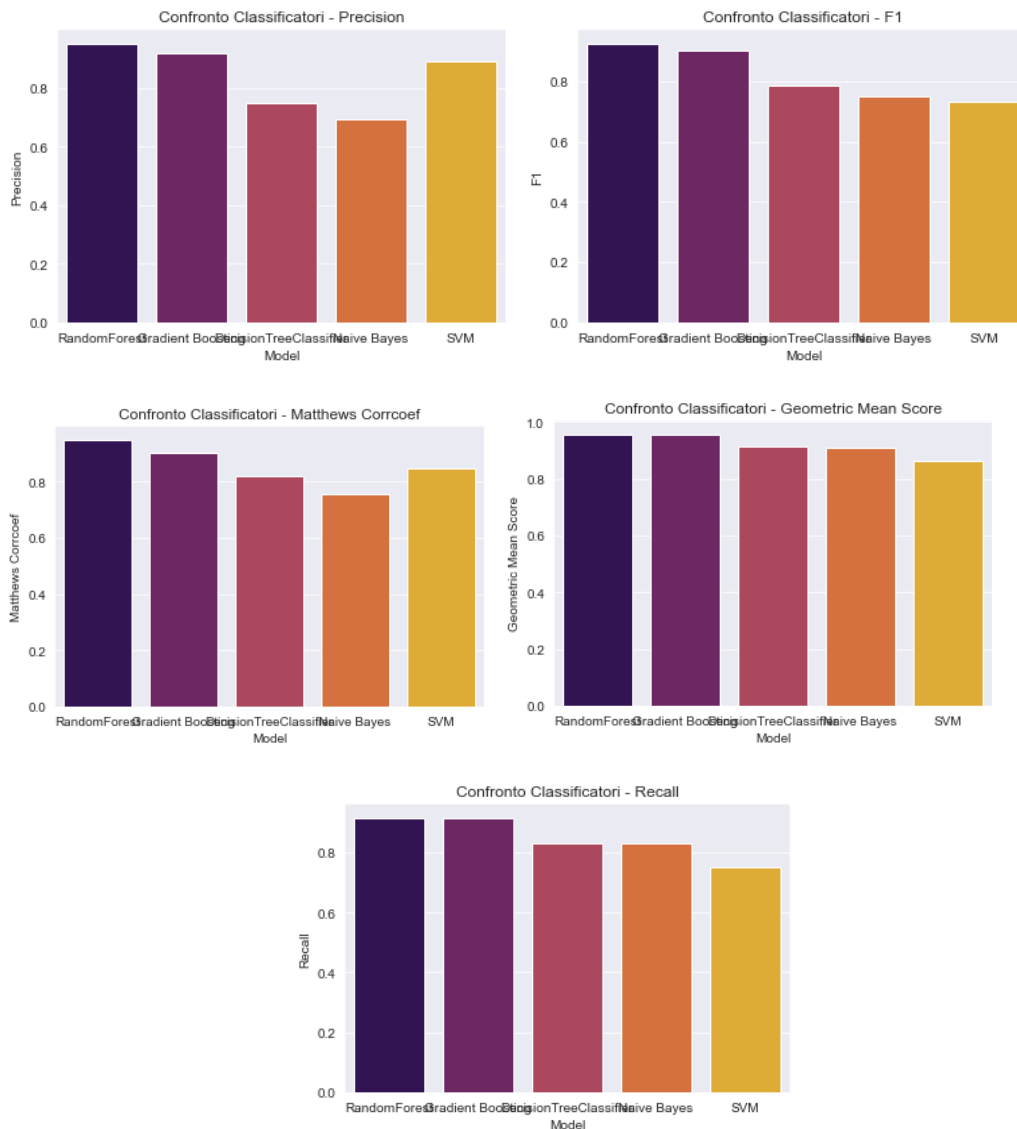


## Confronto Riassuntivo

Dopo aver addestrato i vari modelli li abbiamo messi a confronto con una tabella riassuntiva e con una serie di grafici che confronteranno le singole metriche per ogni modello.

Come si evince dai grafici precedenti il modello migliore è risultato il **Random Forest**, sebbene nel complesso si siano comportati tutti ottimamente. Unica eccezione è **Gradient Boost**, il quale sebbene abbia avuto anche esso degli ottimi risultati, la fase di apprendimento ha impiegato una quantità di risorse (tempo) eccessiva.

	Model	Train Accuracy	Test Accuracy	Precision	Recall	F1	Matthews Corrcoeff	Geometric Mean Score
2	RandomForest	0.998	0.999	0.952	0.917	0.927	0.950	0.957
3	Gradient Boosting	0.998	0.997	0.917	0.916	0.905	0.905	0.957
0	DecisionTreeClassifier	0.997	0.995	0.750	0.832	0.785	0.820	0.912
1	Naive Bayes	0.991	0.992	0.694	0.832	0.752	0.758	0.911
4	SVM	0.996	0.996	0.889	0.750	0.733	0.849	0.865



## Considerazioni finali e implementazioni future

Un relativamente grande problema che riconosciamo al nostro progetto è il non essere riusciti a importare il dataset, in formato .csv, e formattare i dati in modo tale da renderli fatti della Knowledge Base.

Abbiamo dovuto quindi riempire a mano la KB di fatti, con il risultato di aver ottenuto una base con un numero di fatti **esiguo**, e ciò ha portato l'algoritmo di induzione

logica ad apprendere regole di classificazione sì esatte, ma non contraddistinte dalla completezza che ci saremmo aspettati.

Una possibile implementazione futura è quella appunto di automatizzare la conversione dei dati **.csv** in fatti prolog e scoprire, oltre a indurre, le nuove regole risultanti.

Un'altra eventuale implementazione è fornire un supporto **Web** totale al programma con strumenti come Streamlit, appunto, o PyWebIO.

Sicuramente il dataset utilizzato con le future missioni spaziali verrà aggiornato con nuovi pianeti, rendendolo meno sbilanciato, e pertanto le valutazioni fatte sui vari modelli saranno più significative.

Trovare un metodo alternativo per poter fare la cross validation, in modo tale da realizzare un hot encoding sulle features target che ci permetta quindi di sfruttare l'insieme di quelle funzioni che permettono di plottare le metriche valutate.  
Es. ROC SPACE o PRECISION RECALL SPACE.

## Dizionario spaziale

Brevi cenni e definizioni per consentire a ciascun lettore di comprendere il dominio trattato:

### Esopianeta

Un **planeta extrasolare** o **esopianeta** è un [planeta](#) non appartenente al [sistema solare](#), [orbitante](#) cioè attorno a una [stella](#) diversa dal [Sole](#).

Al 20 agosto 2022 risultano conosciuti 5151 pianeti extrasolari in 3800 sistemi planetari diversi.

### Zona abitabile

La [zona abitabile](#), è una regione dello spazio le cui condizioni favoriscono la presenza della vita. Anzitutto, affinché una zona del cosmo sia abitabile, deve rispettare delle importanti condizioni spaziali: deve infatti avere una certa posizione nella galassia (zona galattica abitabile), e un'altra determinata posizione all'interno di un sistema solare (zona circumstellare abitabile). I pianeti e le lune che rispettano queste prime condizioni sono i **migliori candidati al sostentamento della vita**.

Da un altro punto di vista, è il termine scientifico per indicare la regione intorno ad una stella dove è teoricamente possibile per un pianeta mantenere **acqua liquida** sulla sua superficie. Tuttavia, la teoria della zona abitabile risulta per alcuni scienziati troppo semplicistica in quanto viene presa in considerazione solo la vita presente sulla Terra, mentre potrebbero

esistere zone abitabili in cui altri composti **diversi** dall'acqua, come l'[ammoniaca](#) e il [metano](#), possono esistere in forme liquide stabili.

## Temperatura di equilibrio

La **temperatura di equilibrio planetaria** è la [temperatura](#) teorica che corrisponde alla situazione in cui l'energia apportata dal Sole è in perfetto equilibrio con l'energia irradiata dalla superficie planetaria. In questo modello, non è considerata l'eventuale presenza di un'[atmosfera](#) (e quindi dell'[effetto serra](#) che potrebbe interessarla, che tratterrebbe l'energia irradiata dalla superficie).

Per gli esopianeti è pressoché impossibile stabilire la [temperatura di superficie](#), perciò ci si rifà a quella di equilibrio.

## Classe spettrale

La [classificazione](#) più usata per dividere le stelle si basa sulla **temperatura (superficiale)**, ossia sul loro [spettro](#), dato che sono parametri strettamente collegati, essendo lo spettro l'insieme delle diverse lunghezze d'onda che compongono la luce.

## Eccentricità

L'[eccentricità](#) può essere considerata come la misura di quanto l'**orbita** sia deviata da un cerchio (quella della Terra è pressoché circolare, avendo eccentricità inferiore a 0,02).

La maggior parte dei tanti [esopianeti](#) scoperti ha un'eccentricità orbitale maggiore rispetto ai pianeti nel nostro sistema planetario.

Maggiore è l'eccentricità orbitale, più grande è la fluttuazione della temperatura sulla superficie del pianeta. Nonostante si adattino, gli organismi viventi non sono in grado di sopportare eccessive variazioni.

## Magnitudine

La **magnitudine** è, dunque, un numero caratterizzante la brillantezza, apparente oppure assoluta, di un [corpo celeste](#), quindi non solo delle stelle.

## Habitable Zone Indexes

- L'**HZD** (in inglese: *Habitable Zone Distance*) è la distanza dal centro della zona abitabile sulla scala da -1 ad 1, dove -1 rappresenta il confine inferiore della zona, ed 1 rappresenta il confine esterno. Questo valore dipende dalla [luminosità](#) e dalla temperatura di una stella e dal raggio dell' [orbita planetaria](#).<sup>[18]</sup>
- L'**HZC** (in inglese: *Habitable Zone Composition*) è una misura di composizione di un pianeta, dove valori vicini a 0 rappresentano probabilmente la miscela di ferro, di roccia, e di acqua. Valori inferiori a -1 rappresentano corpi probabilmente composti prevalentemente di ferro, e valori superiori a +1 rappresentano corpi composti prevalentemente di gas. Il valore dipende dalla massa e dal raggio di un pianeta.<sup>[18]</sup>

- L'**HZA** (in inglese: *Habitable Zone Atmosphere*) è una misura di abilità di un pianeta di avere un'[atmosfera](#) abitabile, dove valori inferiori a -1 rappresentano corpi che probabilmente non hanno un'atmosfera, e valori superiori a +1 corrispondono a corpi con spesse atmosfere di [idrogeno](#). Pianeti con valori tra il -1 e l'1 più probabilmente hanno un'atmosfera idonea per la vita, benché lo zero non sia necessariamente l'ideale. Il valore dipende dalla massa, dal raggio e l'orbita planetaria e dalla luminosità della stella.<sup>[18]</sup>