



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DOCUMENTAZIONE DI PROGETTO – CORSO DI INGEGNERIA
DELLA CONOSCENZA A.A. 2021-2022

Obesity Prediction:

Analisi e predizione di condizioni di obesità in base alle abitudini alimentari e alle condizioni fisiche: base di conoscenza, ontologia, classificazione e previsione

ANTONIO SEQUENZA (728325) a.sequenza@studenti.uniba.it

ZAGARIA RUGGIERO (727707) r.zagaria15@studenti.uniba.it

SOMMARIO

INTRODUZIONE	3
OBIETTIVO	3
STRUMENTI	3
DATASET	4
DOMINIO	5
KNOWLEDGE BASE	6
GESTORE BASE DI CONOSCENZA	7
ONTOLOGIA	8
GESTORE ONTOLOGIA	12
MACHINE LEARNING	14
DATASET	14
PRE-PROCESSING	16
APPRENDIMENTO SUPERVISIONATO	17
CLASSIFICAZIONE	19
RANDOM FOREST (1° task)	19
RANDOM FOREST (2° task)	21
K-NEAREST NEIGHBORS (1° task)	23
K-NEAREST NEIGHBORS (2° task)	25
SUPPORT VECTOR MACHINE (1° task)	27
SUPPORT VECTOR MACHINE (2° task)	29
RISULTATI FINALI	31
INTERFACCIA	32
CONCLUSIONI	33

INTRODUZIONE

OBIETTIVO

Il progetto nasce con l'obiettivo di predire una condizione di obesità nelle persone in base al loro stile di vita, ed in particolare in base alle loro abitudini alimentari e alle loro condizioni fisiche. Per prima cosa, il dominio di interesse è stato rappresentato tramite knowledge base e ontologia, le quali sono state successivamente interrogate per estrapolare le informazioni. Inoltre, sono state utilizzate tecniche di apprendimento supervisionato, nello specifico la classificazione binaria, e la valutazione di quest'ultima. I classificatori utilizzati sono i seguenti: Random Forest, K-Nearest Neighbors, ed SVM. Infine, dopo aver salvato uno dei modelli, è stata creata un'interfaccia che prevede l'obesità dell'utente in base ai suoi dati, sfruttando l'algoritmo salvato.

STRUMENTI

Gli strumenti utilizzati per la realizzazione di questo progetto sono i seguenti:

- PyCharm: IDE nel quale è stata implementata la maggior parte del progetto, con linguaggio di programmazione Python (<https://www.jetbrains.com/pycharm/>)
- Protégé: software per la creazione e modellazione delle ontologie (<https://protege.stanford.edu/>)
- SWI-Prolog: strumento utilizzato per la creazione della base di conoscenza in Prolog (<https://www.swi-prolog.org/>)

Le librerie utilizzate in Python sono le seguenti:

- PySWIP: libreria che consente di integrare SWI-Prolog all'interno di Python (<https://pypi.org/project/pyswip/0.2.2/>).
- OwlReady2: libreria usata per gestire ontologie all'interno di Python (<https://pypi.org/project/Owlready2/>)
- Pandas: libreria utilizzata per la manipolazione e l'analisi dei Dataset (<https://pandas.pydata.org/>)
- Matplotlib: libreria utilizzata per la creazione e visualizzazione dei grafici (<https://matplotlib.org/>)
- Seaborn: libreria utilizzata per la creazione dei e la visualizzazione dei dati (<https://seaborn.pydata.org/>)

- Sklearn: libreria utilizzata per il pre-processing e apprendimento supervisionato (<https://scikit-learn.org/stable/>)
- Pickle5 : libreria usata per serializzare e deserializzare una struttura di oggetti (<https://pypi.org/project/pickle5/>)

DATASET

Inizialmente il Dataset utilizzato, reperibile attraverso il link <https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+>, ha una struttura composta da 2111 esempi e 17 features:

Data Set Characteristics:	Multivariate	Number of Instances:	2111	Area:	Life
Attribute Characteristics:	Integer	Number of Attributes:	17	Date Donated	2019-08-27
Associated Tasks:	Classification, Regression, Clustering	Missing Values?	N/A	Number of Web Hits:	90505

Tali attributi vengono etichettati con la variabile di classe 'NObesity' (Obesity Level) , che permette di classificare i dati utilizzando i seguenti valori:

- Insufficient_Weight;
- Normal_weight;
- Overweight_Level_I;
- Overweight_Level_II;
- Obesity_Type_I;
- Obesity_Type_II;
- Obesity_Type_III;

DOMINIO

Sono state create due rappresentazioni diverse del dominio di interesse. La prima attraverso la base di conoscenza sviluppata in SWI-Prolog e successivamente importata in PyCharm tramite la libreria PySWIP, mentre la seconda tramite l'ontologia sviluppata in Protégé e importata in PyCharm tramite la libreria OWLReady2.

Per poter visualizzare entrambe le rappresentazioni del dominio basta avviare lo script denominato domain.py, e, attraverso un menù è possibile aprire i menù relativi alla KB e all'Ontologia:

```
----- MENU' DOMINIO -----
| Qui puoi selezionare come visualizzare il dominio di interesse: |
| 1. Knowledge Base in Prolog |
| 2. Ontologia con owlready2 |
| 3. Esci |
-----
```

KNOWLEDGE BASE

Una base di conoscenza è un insieme di regole e fatti validi per il dominio di interesse. A tal proposito, è stata costruita una base di conoscenza in Prolog, composta da 17 regole, e poi importata in Python mediante la libreria PySWIP, permettendo la l'interrogazione e la verifica sia di proprietà di informazioni personali come l'altezza, il peso, il genere, e sia proprietà più complesse come le abitudini alimentari e le condizioni fisiche della persona:

```
% RULES

% La persona ha delle informazioni personali
prop(X, has, personal_information):-
    prop(X, has_age, A),
    A:=0,
    prop(X, has_gender, G),
    G:=0,
    prop(X, has_weight, W),
    W:=0,
    prop(X, has_height, H),
    H:=0,
    prop(X, has_fwo, F),
    F:=0.

% La persona ha un'età
prop(X, has_age, A):-
    prop(A, isOf, X),
    prop(A, subClassOf, personal_information).

% La persona ha un genere
prop(X, has_gender, G):-
    prop(G, isOf, X),
    prop(G, subClassOf, personal_information).

% La persona ha un peso
prop(X, has_weight, W):-
    prop(W, isOf, X),
    prop(W, subClassOf, personal_information).

%...▲

% FACTS

prop(mario_rossi, has_age, 26).
prop(gaia_bianchi, has_age, 21).
prop(angelica_borraccino, has_age, 25).
prop(christian_riefolo, has_age, 27).
prop(cristina_miani, has_age, 22).
prop(ilaria_dicandia, has_age, 21).
prop(luca_verdi, has_age, 2 ).
prop(mattia_biondi, has_age, 22).
prop(nicole_marrone, has_age, 26).

prop(mario_rossi, has_gender, male).
prop(gaia_bianchi, has_gender, female).
prop(angelica_borraccino, has_gender, female).
prop(christian_riefolo, has_gender, male).
prop(cristina_miani, has_gender, female).
prop(ilaria_dicandia, has_gender, female).
prop(luca_verdi, has_gender, male).
prop(mattia_biondi, has_gender, male).
prop(nicole_marrone, has_gender, female).

prop(mario_rossi, has_bmi, 31).
prop(gaia_bianchi, has_bmi, 24).
prop(angelica_borraccino, has_bmi, 36).
prop(christian_riefolo, has_bmi, 27).
prop(cristina_miani, has_bmi, 17).
prop(ilaria_dicandia, has_bmi, 27).
prop(luca_verdi, has_bmi, 26).
prop(mattia_biondi, has_bmi, 29).
prop(nicole_marrone, has_bmi, 41).

%...▲
```

GESTORE BASE DI CONOSCENZA

Selezionando dunque dal menù dominio l'opzione 1, si apre un ulteriore menù che permette l'interrogazione della base di conoscenza tramite apposite query in PySWIP sviluppate all'interno del file denominato KBManager.py:

```
----- MENU' PROLOG -----  
| Qui puoi selezionare le informazioni da visualizzare tramite query prolog: |  
| 1. Visualizza le persone obese |  
| 2. Visualizza le persone non obese |  
| 3. Cerca una persona in base al suo BMI |  
| 4. Cerca il bmi di una persona |  
| 5. Cerca una persona in base ad un'informazione |  
| 6. Cerca un'informazione relativa ad una persona |  
| 7. Torna al menu dominio |  
| 8. Esci |  
-----
```

Attraverso il precedente menù si possono essere effettuate ad esempio delle query che permettono la visualizzazione delle persone obese:

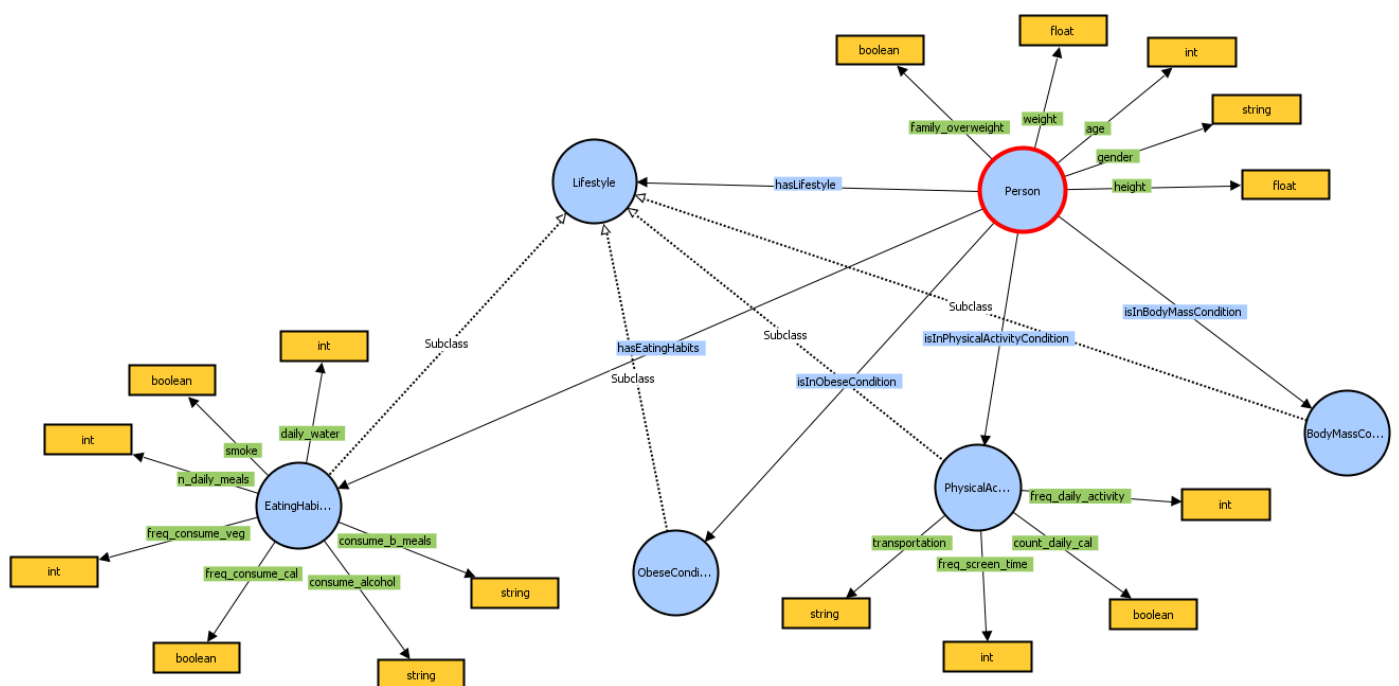
```
1  
- mario_rossi  
- angelica_borraccino  
- nicole_marrone
```

Oppure si possono trovare le persone appartenente una determinata categoria di peso:

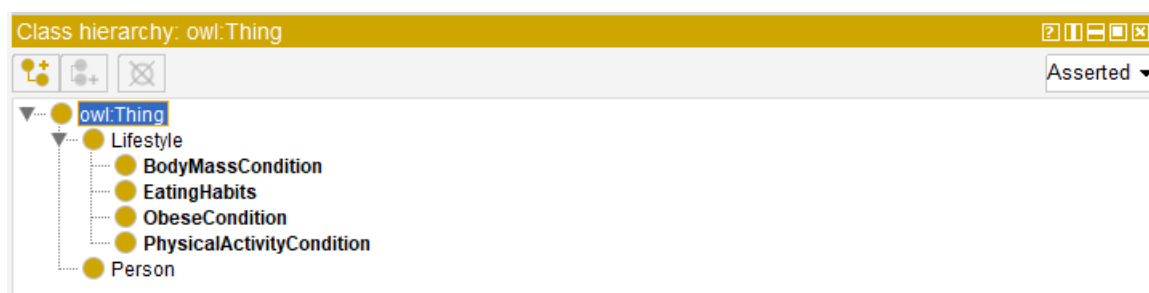
```
1  
Scegli un BMI tra: 1. Insufficient_Weight, 2. Normal_Weight, 3. Overweight, 4. Obesity_Type_1, 5. Obesity_Type_2, 6. Obesity_Type_3  
1  
- cristina_miani
```

ONTOLOGIA

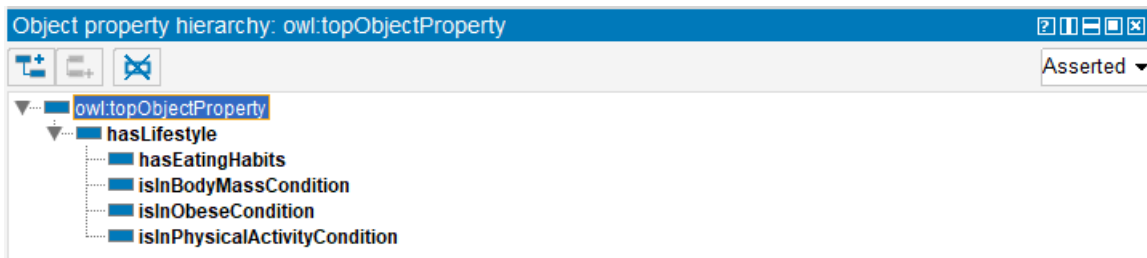
Un'ontologia è una specifica del significato dei simboli usato in un sistema d'informazione. Per poter crearla è stato utilizzato il programma Protégé. Sono state create quindi varie classi e sottoclassi, e sono state aggiunte proprietà di oggetto e proprietà di dati con un dominio e un relativo range. Inoltre è stato possibile interrogare l'ontologia effettuando delle query usando l'estensione *DL Query*. Di seguito viene mostrata l'ontologia per intero visualizzata tramite il plugin di Protégé chiamato VOWL:



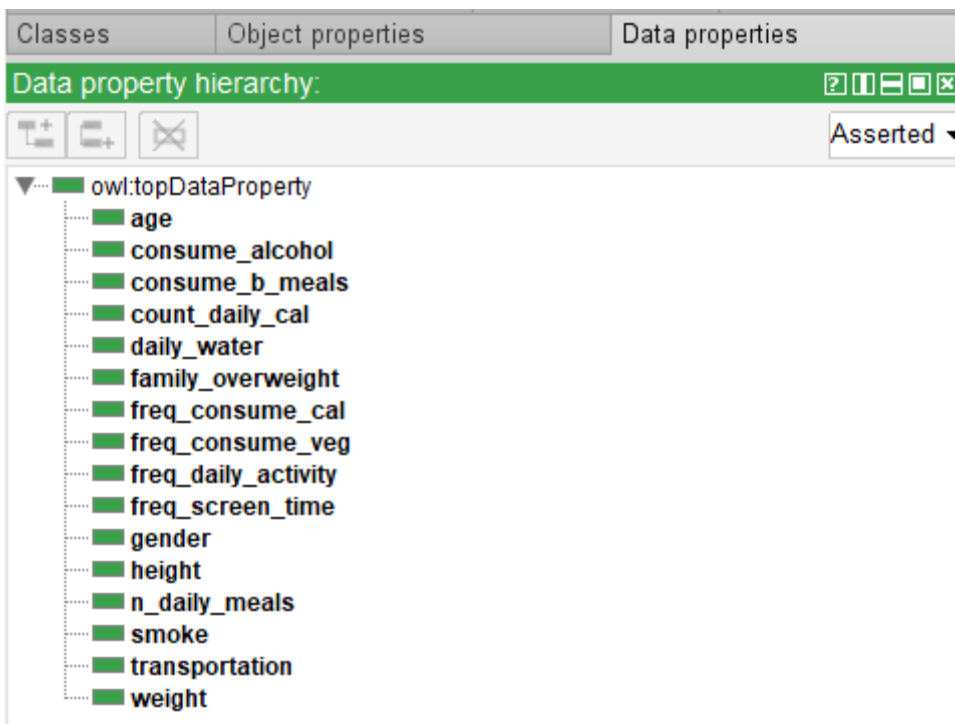
Dunque l'ontologia è composta essenzialmente dalle seguenti classi, che rappresentano i concetti essenziali:



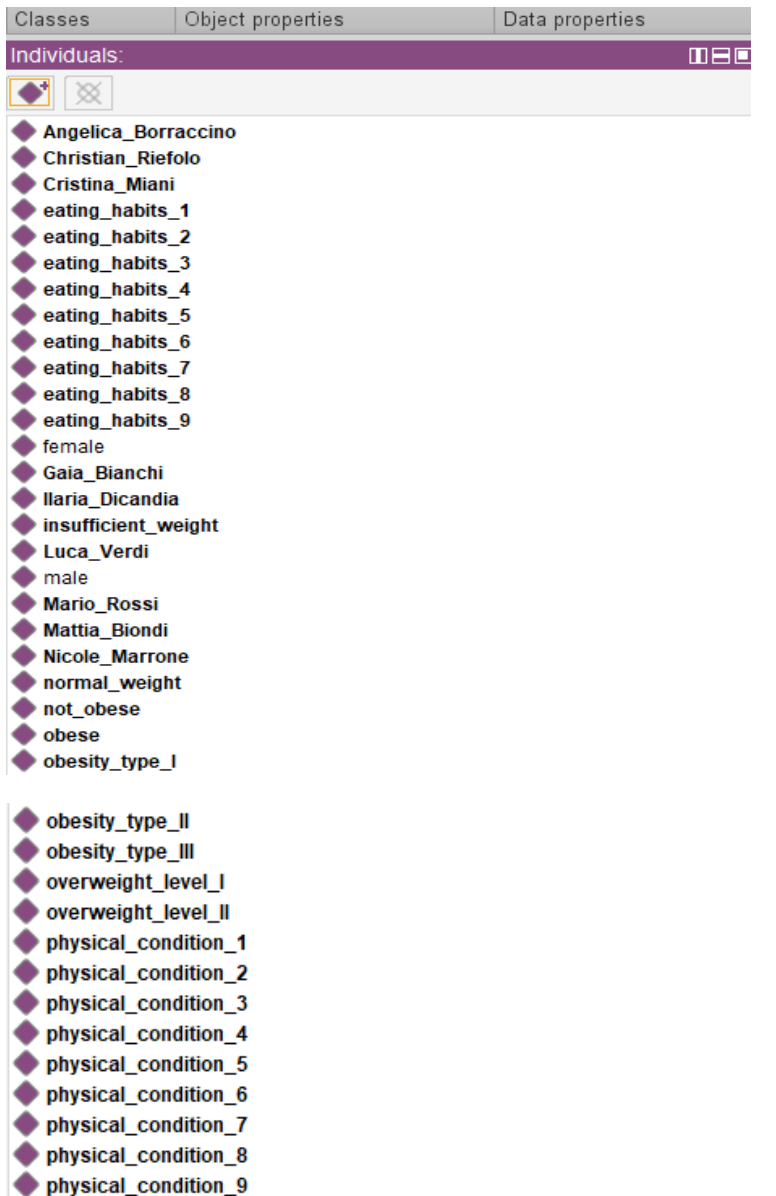
Si riportano inoltre le proprietà di oggetto, ovvero delle proprietà in grado di mettere in relazione due individui di stessa classe oppure diversa:



Oltre a quest'ultime, sono state anche definite le proprietà sui dati che mettono in relazione un individuo con il suo valore primitivo (int, string, etc), come mostrato nella seguente figura:



Successivamente è stata creata una serie di istanze per ogni entità. Ad esempio abbiamo le istanze relative alle persone, alle condizioni fisiche, alle abitudini alimentari ecc:



Infine, dopo aver creato gli individui, si è passati a formulare delle query tramite il plug-in *Query DL* per interrogare l'ontologia e visualizzare la correttezza degli output:

The image shows two side-by-side screenshots of a web interface for querying an ontology using DL (Description Logic) queries. Each screenshot has a yellow header bar labeled 'DL query:'. Below the header is a section titled 'Query (class expression)' with a text input field. Underneath the input field are two buttons: 'Execute' and 'Add to ontology'. Below this is a section titled 'Query results' which shows the number of instances found and a list of individual names, each preceded by a purple diamond icon.

Left Screenshot:

DL query: **Query (class expression)**
Person **that** isInObeseCondition **value** obese

Execute Add to ontology

Query results
Instances (3 of 3)

- ◆ Angelica_Borraccino
- ◆ Mario_Rossi
- ◆ Nicole_Marrone


Right Screenshot:

DL query: **Query (class expression)**
Person **that** hasEatingHabits **value** eating_habits_4 **or** Person **that** isInPhysicalActivityCondition **value** physical_condition_1

Execute Add to ontology

Query results
Instances (2 of 2)

- ◆ Gaia_Bianchi
- ◆ Ilaria_Dicandia

 Figura 1 La prima query restituisce le persone che rientrano nella categoria di tipo 'Obese', mentre la seconda restituisce le persone che hanno delle abitudini alimentari di tipo 4 oppure le persone che si trovano in una condizione fisica di tipo 1

GESTORE ONTOLOGIA

Nell'ambito del dominio studiato, l'ontologia è stata sviluppata in Protégé, e successivamente importata in Python tramite la libreria OWLReady2. Selezionando dal menù dominio l'opzione 2 si può accedere al menù relativo all'ontologia, che permette *un'ulteriore* interrogazione di quest'ultima tramite query sviluppate all'interno del file denominato ontologyManager.py:

```
2
----- MENU' ONTOLOGIA -----
| Qui puoi selezionare le informazioni da visualizzare tramite query owlready2: |
| 1. Mostra il contenuto principale dell'ontologia |
| 2. Mostra le persone presenti nell'ontologia |
| 3. Mostra le tipologie di BMI nell'ontologia |
| 4. Mostra le tipologie di condizioni fisiche |
| 5. Mostra le tipologie di abitudini alimentari |
| 6. Mostra le tipologie di condizioni di attività fisiche |
| 7. Esempio di query |
| 8. Torna al menu' dominio |
| 9. Esci |
-----
```

Attraverso il precedente menù possono essere effettuate ad esempio delle query che permettono la visualizzazione delle informazioni contenute all'interno dell'ontologia.

Il contenuto dell'ontologia è il seguente:

```
ONTOLOGIA

Lista classi nella ontologia:

[OntologiaProgetto.Person, OntologiaProgetto.EatingHabits, OntologiaProgetto.BodyMassCondition, OntologiaProgetto.HealthCondition, OntologiaProgetto.ObeseCondition, OntologiaProgetto.PhysicalActivityCondition]

Lista Persone nella ontologia:

[OntologiaProgetto.Person, OntologiaProgetto.Angelica_Borraccino, OntologiaProgetto.Christian_Riefolo, OntologiaProgetto.Cristina_Miani, OntologiaProgetto.Gaia_Bianchi, OntologiaProgetto.Maria_Cristina_Mariani]

Lista tipologie BMI:

[OntologiaProgetto.BodyMassCondition, OntologiaProgetto.obesity_type_II, OntologiaProgetto.overweight_level_II, OntologiaProgetto.insufficient_weight, OntologiaProgetto.normal_weight]

Lista tipologie condizioni fisiche:

[OntologiaProgetto.ObeseCondition, OntologiaProgetto.obese, OntologiaProgetto.not_obese]

Lista tipologie abitudini alimentari:

[OntologiaProgetto.EatingHabits, OntologiaProgetto.eating_habits_8, OntologiaProgetto.eating_habits_5, OntologiaProgetto.eating_habits_7, OntologiaProgetto.eating_habits_1, OntologiaProgetto.eating_habits_2, OntologiaProgetto.eating_habits_3, OntologiaProgetto.eating_habits_4, OntologiaProgetto.eating_habits_6]

Lista tipologie condizioni di attività fisiche:

[OntologiaProgetto.PhysicalActivityCondition, OntologiaProgetto.physical_condition_8, OntologiaProgetto.physical_condition_5, OntologiaProgetto.physical_condition_7, OntologiaProgetto.physical_condition_1, OntologiaProgetto.physical_condition_2, OntologiaProgetto.physical_condition_3, OntologiaProgetto.physical_condition_4, OntologiaProgetto.physical_condition_6]
```

Un altro esempio di query è il seguente, con la quale è possibile stampare la lista di persone che hanno abitudini alimentari di tipo 2;

```
7
ONTOLOGIA

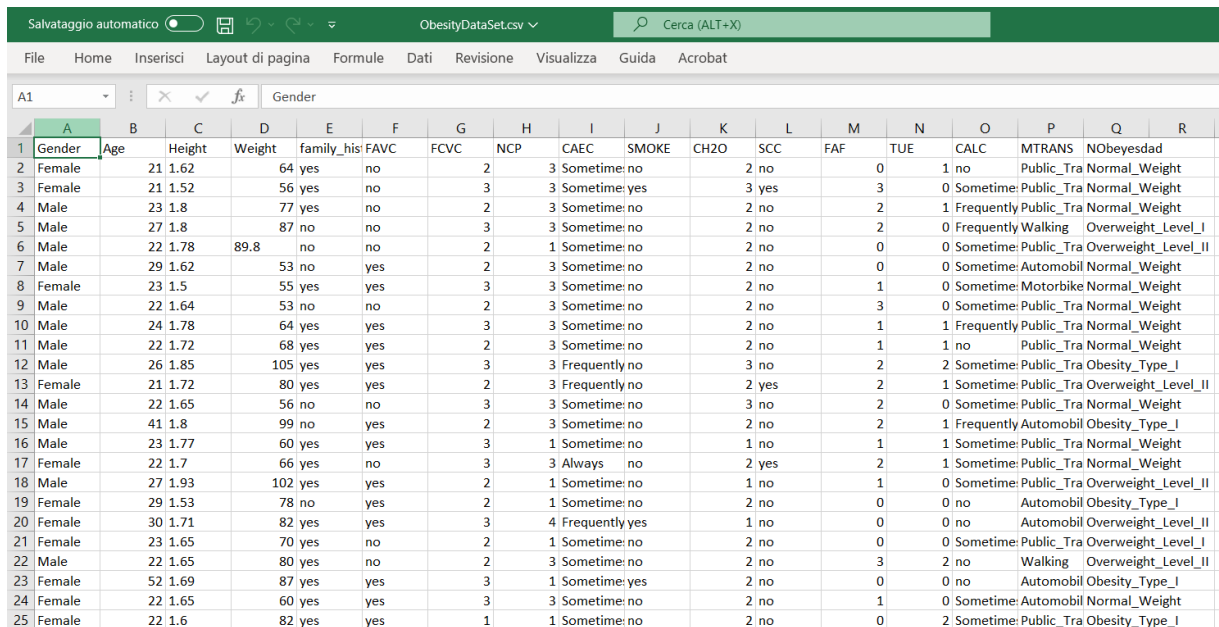
Lista di persone che hanno le abitudini alimentari di tipo 2:

[OntologiaProgetto.Luca_Verdi]
```

MACHINE LEARNING

DATASET

Il dataset inizialmente si presenta nel seguente modo in Excel:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Gender	Age	Height	Weight	family_his	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObesyedad	
2	Female	21	1.62	64	yes	no		2	3	Sometime: no		2	no	0	1	no	Public_Tra Normal_Weight	
3	Female	21	1.52	56	yes	no		3	3	Sometime: yes		3	yes	3	0	Sometime: Public_Tra Normal_Weight		
4	Male	23	1.8	77	yes	no		2	3	Sometime: no		2	no	2	1	Frequently Public_Tra Normal_Weight		
5	Male	27	1.8	87	no	no		3	3	Sometime: no		2	no	2	0	Frequently Walking Overweight_Level_I		
6	Male	22	1.78	89.8	no	no		2	1	Sometime: no		2	no	0	0	Sometime: Public_Tra Overweight_Level_II		
7	Male	29	1.62	53	no	yes		2	3	Sometime: no		2	no	0	0	Sometime: Automobil Normal_Weight		
8	Female	23	1.5	55	yes	yes		3	3	Sometime: no		2	no	1	0	Sometime: Motorbike Normal_Weight		
9	Male	22	1.64	53	no	no		2	3	Sometime: no		2	no	3	0	Sometime: Public_Tra Normal_Weight		
10	Male	24	1.78	64	yes	yes		3	3	Sometime: no		2	no	1	1	Frequently Public_Tra Normal_Weight		
11	Male	22	1.72	68	yes	yes		2	3	Sometime: no		2	no	1	1	no	Public_Tra Normal_Weight	
12	Male	26	1.85	105	yes	yes		3	3	Frequently no		3	no	2	2	Sometime: Public_Tra Obesity_Type_I		
13	Female	21	1.72	80	yes	yes		2	3	Frequently no		2	yes	2	1	Sometime: Public_Tra Overweight_Level_II		
14	Male	22	1.65	56	no	no		3	3	Sometime: no		3	no	2	0	Sometime: Public_Tra Normal_Weight		
15	Male	41	1.8	99	no	yes		2	3	Sometime: yes		2	no	2	1	Frequently Automobil Obesity_Type_I		
16	Male	23	1.77	60	yes	yes		3	1	Sometime: no		1	no	1	1	Sometime: Public_Tra Normal_Weight		
17	Female	22	1.7	66	yes	no		3	3	Always no		2	yes	2	1	Sometime: Public_Tra Normal_Weight		
18	Male	27	1.93	102	yes	yes		2	1	Sometime: no		1	no	1	0	Sometime: Public_Tra Overweight_Level_II		
19	Female	29	1.53	78	no	yes		2	1	Sometime: no		2	no	0	0	no	Automobil Obesity_Type_I	
20	Female	30	1.71	82	yes	yes		3	4	Frequently yes		1	no	0	0	no	Automobil Overweight_Level_II	
21	Female	23	1.65	70	yes	no		2	1	Sometime: no		2	no	0	0	Sometime: Public_Tra Overweight_Level_I		
22	Male	22	1.65	80	yes	no		2	3	Sometime: no		2	no	3	2	no	Walking Overweight_Level_II	
23	Female	52	1.69	87	yes	yes		3	1	Sometime: yes		2	no	0	0	no	Automobil Obesity_Type_I	
24	Female	22	1.65	60	yes	yes		3	3	Sometime: no		2	no	1	0	Sometime: Automobil Normal_Weight		
25	Female	22	1.6	82	yes	yes		1	1	Sometime: no		2	no	0	2	Sometime: Public_Tra Obesity_Type_I		

Le Features sono dunque le seguenti:

Features relative alle informazioni personali:

- Gender: identifica il sesso [Male/ Female];
- Age: identifica l'età [14 – 61];
- Height: identifica l'altezza [1.45 - 1.98];
- Weight: identifica il peso [39 – 173];
- Family_history: storico utilizzato per indicare familiari che soffrono o hanno sofferto per sovrappeso [yes /no];

Features relative alle abitudini alimentari:

- FAVC: indica il consumo frequente di cibi ipercalorici [yes/ no];
- FCVC: indica la frequenza di consumo di verdure [Never/ Sometimes/ Always];
- NCP: indica il numero di pasti principali effettuati di solito [Between 1 or 2/ Three/ More than three];
- CAEC: indica, in termini di frequenza, il consumo di ulteriore cibo tra un pasto e l'altro [Never/ Sometimes/ Frequently/ Always];
- SMOKE: indica se il paziente fuma o meno [yes/ no];

- CH20: indica la frequenza del consumo di acqua giornaliero [Less than a liter/Between 1 and 2 L/ More than 2 L];
- CALC: indica la frequenza del consumo di alcohol [no/ Sometimes/ Frequently/ Always];

Features relative alla condizione fisica:

- SCC: indica se viene effettuato un monitoraggio del consumo calorico [yes/ no];
- FAF: indica la frequenza di attività fisica svolta [Never/ 1 or 2 days/ 2 or 4 days/ 4 or 5 days];
- TUE: indica la frequenza dell'uso di dispositivi elettronici [0–2 hours/ 3–5 hours/ More than 5 hours];
- MTRANS: indica i mezzi di trasporto utilizzati [Public transportation, Motorbike, Bike, Walking];

PRE-PROCESSING

Il Dataset è composto inizialmente da 17 features e 2111 istanze. Dopo aver verificato l'eventuale presenza di valori nulli, sono state rinominate alcune colonne per migliorare la leggibilità, e successivamente sono stati convertiti i valori delle features in valori numerici per poter effettuare la classificazione. Inoltre, è stato riportato il pie chart per visualizzare la distribuzione della classe target:



Come si evince dal pie chart, la distribuzione delle classi target è abbastanza bilanciata, dunque non c'è bisogno di effettuare quest'ultima operazione.

APPENDIMENTO SUPERVISIONATO

Con l'Apprendimento Supervisionato si cerca di costruire un modello partendo da dei dati di addestramento etichettati, con i quali si cerca di fare previsioni su dati non disponibili o futuri. Infatti, con il termine 'Supervisione' si intende che nell'insieme dei campioni, i segnali di output desiderati sono già noti poiché precedentemente etichettati. Nel contesto di interesse, l'obiettivo sarà quello di addestrare i classificatori per poter predire la categoria di obesità appartenente alle persone. Infatti, le classi iniziali erano le seguenti:

- Insufficient_Weight;
- Normal_Weight;
- Overweight_Level_I;
- Overweight_Level_II;
- Obesity_Type_I;
- Obesity_Type_II;
- Obesity_Type_III;

Le suddette classi sono state riunite in due grandi classi, sulle quali poi è stata applicata la *classificazione binaria*:

- Not_obese (Insufficient_Weight, Normal_Weight, Overweight_Level_I, Overweight_Level_II)
- Obese (Obesity_Type_I, Obesity_Type_II, Obesity_Type_III)

Sono stati implementati i seguenti classificatori addestrati sul Dataset:

- Random Forest
- K-Nearest Neighbors
- SVM

In particolare, una volta modificato il dataset, sono stati effettuati due task differenti utilizzando i suddetti classificatori per poter confrontare i risultati ottenuti:

- Split in train e test set pari a 75%-25%
- Split in train e test set pari a 60%-40%

Inoltre, per tutti i classificatori è stato effettuato il tuning degli Iperparametri tramite GridSearchCV con una stratified Kfold cross-validation (default) per poter sfruttare al massimo le capacità dei classificatori in questione. Infine, per ogni algoritmo di apprendimento supervisionato sono state prodotte le seguenti misure e i seguenti grafici:

- Accuratezza training e test
- Report di classificazione
- Matrice di confusione
- Curva Precision-Recall

CLASSIFICAZIONE

RANDOM FOREST (1° task)

Il Random Forest è un modello costituito da molti alberi di decisione, ognuno dei quali fornisce una predizione. Esse vengono poi, combinate allo scopo di ottenere una previsione complessiva della foresta per un dato esempio.

Sono stati selezionati i seguenti parametri per il classificatore Random Forest:

- *n_estimator*=5
- *max_depth*=2
- *random_state*=0

Il numero di stimatori (alberi) è dunque pari a 5, e si è deciso di selezionare una profondità iniziale bassa pari a 2.

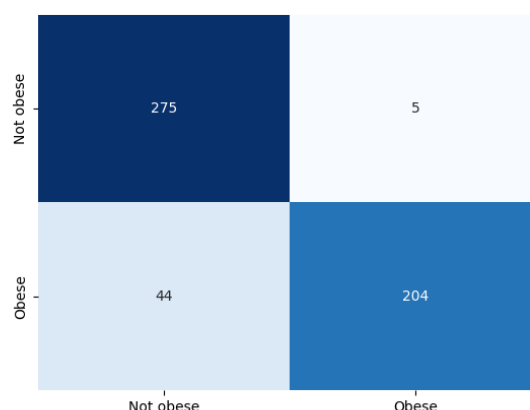
Le misure di valutazione e i grafici per il Random Forest sono invece i seguenti:

```
Accuracy train : 0.897
Accuracy test  : 0.907

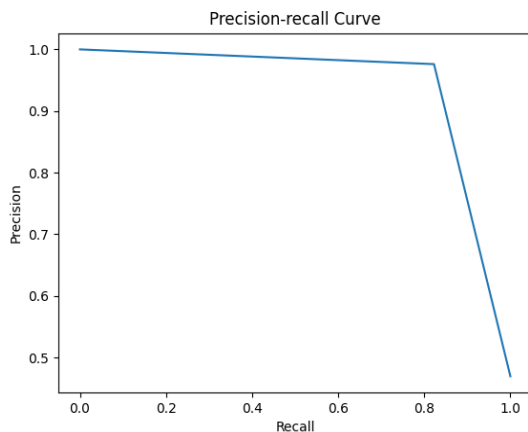
Classification Report:

```

	precision	recall	f1-score	support
0	0.86	0.98	0.92	280
1	0.98	0.82	0.89	248
accuracy			0.91	528
macro avg	0.92	0.90	0.91	528
weighted avg	0.92	0.91	0.91	528



Come si evince dal report l'accuratezza è del 91%, dunque un risultato abbastanza buono per questo primo task, soprattutto per quanto riguarda la precision per la classe 1 che riguarda gli obesi. Anche la matrice di confusione evidenzia dei buoni risultati ma con un numero un po' alto di falsi negativi (44), quindi li considera come non obesi ma in realtà sono effettivamente obesi.



Infine, la curva precision-recall evidenzia un buon rapporto tra le due metriche. Dunque è evidente che nella classificazione non sono riscontrati particolari problemi e i risultati sono abbastanza buoni.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stata applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'n_estimators': (2, 3, 4, 5, 10, 100),
              'max_depth': (1, 2, 3, 4, 5)
             }
```

Di seguito vengono riportati i risultati del tuning degli Iperparametri assieme ai migliori Iperparametri:

```
Best mean score (with cv): 0.955971 (Standard deviation: 0.053377) using {'max_depth': 5, 'n_estimators': 4}
Train-set score           : 0.975
Test-set score            : 0.989
```

Come si può notare dai risultati della GridSearchCV, nonostante sia stata scelta una profondità massima ottimale pari a 5 rispetto alla precedente (2) e un numero di stimatori ottimali ridotto di 1, c'è stato un buon incremento dell'accuratezza (media), così come anche gli score riguardanti il train-set e test-set.

RANDOM FOREST (2° task)

Dopo aver effettuato il secondo split (60%-40%) è stato applicato il classificatore Random Forest per una seconda valutazione. Le metriche utilizzate sono identiche alle precedenti:

- *n_estimator*=5
- *max_depth*=2
- *random_state*=0

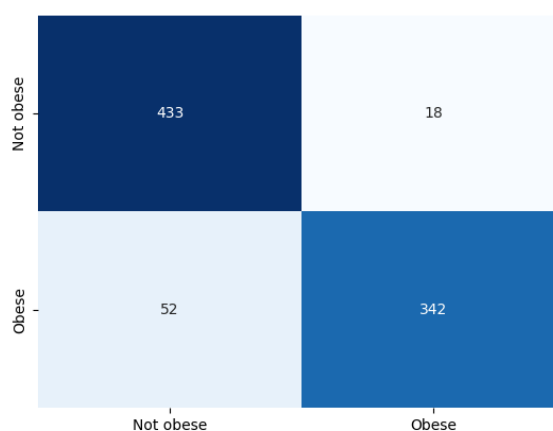
Le misure di valutazione e i grafici per il Random Forest sono invece i seguenti:

```
Accuracy train : 0.893
Accuracy test  : 0.917

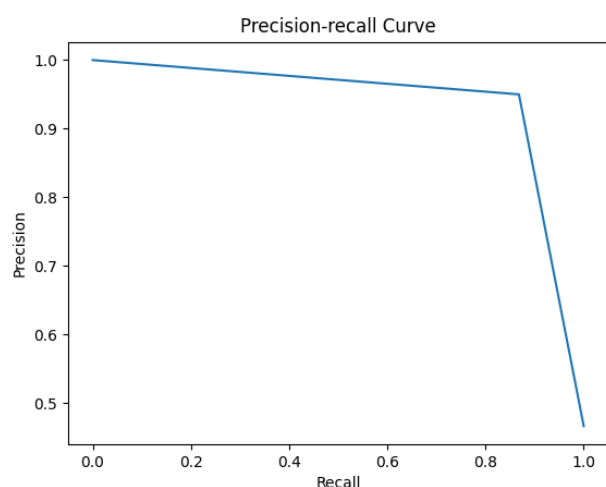
Classification Report:
              precision    recall  f1-score   support

      0       0.89       0.96       0.93       451
      1       0.95       0.87       0.91       394

   accuracy          0.92          0.92          0.92       845
  macro avg       0.92       0.91       0.92       845
 weighted avg       0.92       0.92       0.92       845
```



Come si evince da questa seconda valutazione, l'accuratezza del classificatore è aumentata dell'1% rispetto alla prima. Anche riguardo la matrice di confusione non sono stati rilevati grandi cambiamenti, ma il numero dei falsi positivi è comunque alto come la precedente.



Riguardo infine la curva precision-recall, considerando l'andamento delle precedenti metriche, è logico aspettarsi una curva simile alla precedente.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stata applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'n_estimators': (2, 3, 4, 5, 10, 100),  
              'max_depth': (1, 2, 3, 4, 5)  
              }
```

Dopo aver effettuato anche qui il tuning degli Iperparametri i risultati sono stati i seguenti:

```
Best mean score (with cv): 0.955971 (Standard deviation: 0.053377) using {'max_depth': 5, 'n_estimators': 4}  
Train-set score           : 0.971  
Test-set score            : 0.989
```

Come si nota lo score è sicuramente più alto del pre-tuning (2° task) e gli Iperparametri sono anche uguali a quelli 1° task. Dunque da questo 2° task effettuato su uno split differente si evince che il classificatore ha ottenuto dei risultati leggermente migliori ma comunque simili.

K-NEAREST NEIGHBORS (1° task)

Il K-Nearest Neighbors è un algoritmo di apprendimento supervisionato che consiste nell'individuare i k esempi più vicini a quello che si intende classificare, a quest'ultimo viene quindi attribuita la categoria "più ricorrente" tra i k esempi più vicini. Sono stati scelti i seguenti parametri per il classificatore KNN:

- *n_neighbors*=5,
- *weights*='distance'
- *metric*='minkowski'

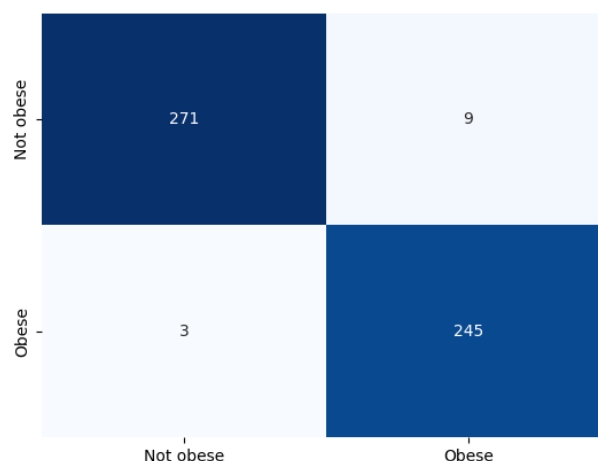
E' stato selezionato dunque un k iniziale pari a 5, e come peso una distanza, che è quella di mikowski, per non penalizzare i dati, non essendo il dataset normalizzato. Le misure di valutazione e i grafici per il KNN sono invece i seguenti:

```
Accuracy train : 1.000
Accuracy test  : 0.977

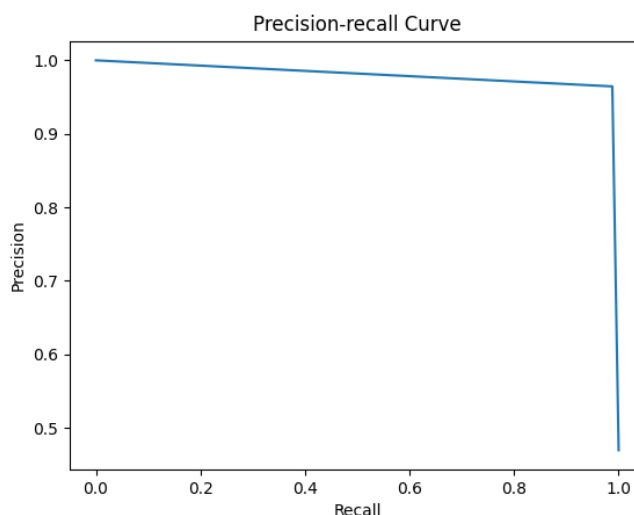
Classification Report:
              precision    recall  f1-score   support

     0       0.99       0.97       0.98        280
     1       0.96       0.99       0.98        248

   accuracy          0.98          0.98          0.98        528
  macro avg       0.98       0.98       0.98        528
 weighted avg       0.98       0.98       0.98        528
```



Come si evince dal report e dalla matrice di confusione, i risultati sono davvero ottimi (98% di accuratezza). Anche precision e recall sono ottimi per entrambe le classi. L'unico problema potrebbe essere la presenza di overfitting, infatti come si nota l'accuratezza sui dati di training è maggiore rispetto a quella sui dati di



test, e questo indica che il modello, essendo anche molto forte, si è adattato alla perfezione sui dati di training.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stato applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'n_neighbors': (2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                               18, 19, 20),
               'metric': ['euclidean', 'manhattan', 'minkowski'],
               'weights': ['uniform', 'distance']}
}
```

Di seguito vengono riportati i risultati del tuning degli Iperparametri assieme ai migliori Iperparametri:

```
Best mean score (with cv): 0.990054 (Standard deviation: 0.037742) using {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Train-set score           : 1.000
Test-set score            : 1.000
```

Si evince dunque che il problema dell'overfitting è stato risolto alla perfezione. Lo score migliore è del 99% e l'accuratezza sui due set di esempi è identica. Inoltre, la metrica scelta è questa volta quella di manhattan (che generalizza quella euclidea e di minkowski) anziché quella di minkowski iniziale, e con un k pari a 3, rispetto ai 5 iniziali.

K-NEAREST NEIGHBORS (2° task)

Dopo aver effettuato il secondo split (60%-40%) è stato applicato il classificatore KNN per una seconda valutazione. Le metriche utilizzate sono identiche alle precedenti:

- *n_neighbors=5*,
- *weights='distance'*
- *metric='minkowski'*

Le misure di valutazione e i grafici per il KNN sono invece i seguenti:

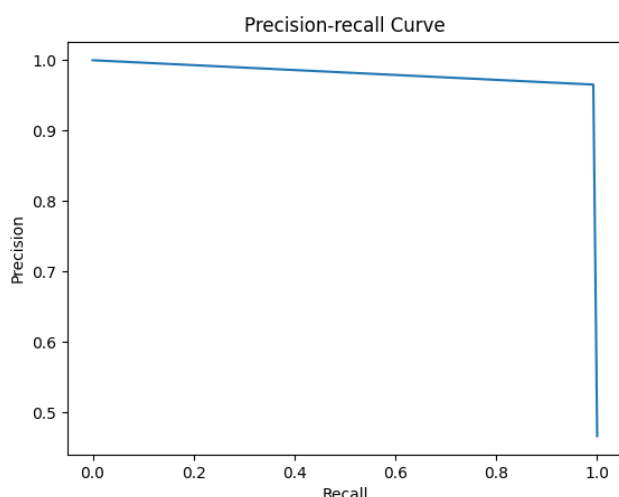
```
Accuracy train : 1.000
Accuracy test  : 0.980

Classification Report:
              precision    recall  f1-score   support

     0         0.99      0.97      0.98       451
     1         0.97      0.99      0.98       394

 accuracy          0.98          0.98          0.98      845
  macro avg         0.98          0.98          0.98      845
 weighted avg         0.98          0.98          0.98      845
```

Not obese	437	14
Obese	3	391
	Not obese	Obese



Si evince dunque che l'accuratezza è molto simile a quella iniziale, dunque i risultati sono ottimi per tutte le metriche, ma soprattutto che il problema dell'overfitting potrebbe ripresentarsi anche in questo caso. Anche la curva di precision-recall si presenta simile alla precedente, dunque con risultati ottimi.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stato applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'n_neighbors': (2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                                18, 19, 20),
               'metric': ['euclidean', 'manhattan', 'minkowski'],
               'weights': ['uniform', 'distance']}
}
```

Dopo aver effettuato anche qui il tuning degli Iperparametri i risultati sono stati i seguenti:

```
Best mean score (with cv): 0.990054 (Standard deviation: 0.037742) using {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Train-set score           : 1.000
Test-set score            : 1.000
SVM:
```

Come si può notare, con GridSearchCV non ci sono stati cambiamenti, ed è stato risolto ancora una volta l'overfitting. Non c'è stata neanche una variazione del k. Dunque, similmente al 1° task di KNN, i risultati sono ottimi anche con uno split differente.

SUPPORT VECTOR MACHINE (1° task)

Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi appartenenti alle due diverse categorie siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono quindi mappati nello stesso spazio e la predizione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricade.

Sono stati selezionati i seguenti parametri per il classificatore SVM:

- *kernel='linear'*
- *C=1.0 (default)*

E' stato scelto inizialmente un kernel semplice e lineare, con un C (parametro di regolarizzazione) di default.

Le misure di valutazione e i grafici per il SVM sono dunque i seguenti:

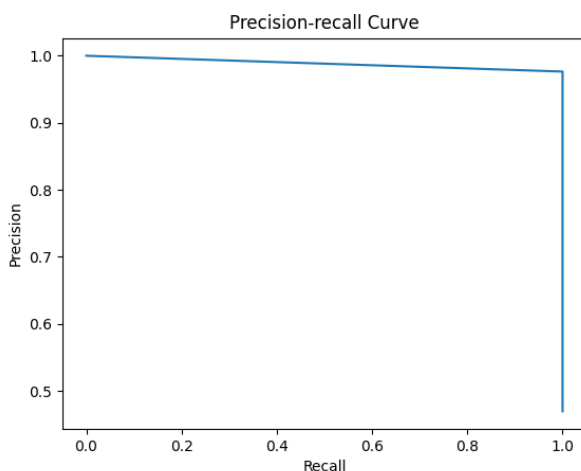
```
Accuracy train : 0.981
Accuracy test  : 0.989

Classification Report:
              precision    recall  f1-score   support

     0           1.00       0.98       0.99       280
     1           0.98       1.00       0.99       248

 accuracy              0.99              0.99       0.99       528
 macro avg           0.99       0.99       0.99       528
 weighted avg        0.99       0.99       0.99       528
```

Not obese	274	6
Obese	0	248
	Not obese	Obese



Come si evince dal report e dalle matrici di confusione, è evidente che nella classificazione non sono riscontrati problemi e i risultati sono ottimi. Si ha infatti un'accuratezza del 99%, così come per precision e recall i sono stati ottenuti risultati ottimi. Guardando la matrice di confusione infatti si può notare un numero

vicinissimo allo 0 sia per i falsi positivi che per i falsi negativi. Infine, anche la curva di precision-recall evidenzia ottimi risultati.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stato applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  
              'C': [50, 10, 1.0, 0.1, 0.01]}
```

Di seguito vengono riportati i risultati del tuning degli Iperparametri assieme ai migliori Iperparametri:

```
Best mean score (with cv): 0.988185 (Standard deviation: 0.001538) using {'C': 10, 'kernel': 'linear'}  
Train-set score           : 0.994  
Test-set score            : 0.998
```

Come si può notare, i risultati sono simili, cambia solo il fattore di regolarizzazione (10) che applica una maggiore penalizzazione al modello. Si sono ottenuti dunque ottimi risultati anche riguardo agli score sui due set di dati.

SUPPORT VECTOR MACHINE (2° task)

Dopo aver effettuato il secondo split (60%-40%) è stato applicato il classificatore SVM per una seconda valutazione. Le metriche utilizzate sono identiche alle precedenti:

- `kernel='linear'`
- `C=1.0 (default)`

Le misure di valutazione e i grafici per l'SVM sono invece i seguenti:

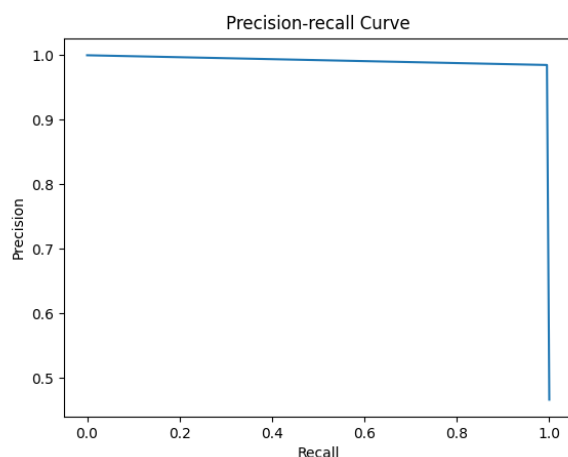
```
Accuracy train : 0.976
Accuracy test  : 0.991

Classification Report:
              precision    recall  f1-score   support

     0           1.00       0.99       0.99       451
     1           0.98       0.99       0.99       394

 accuracy          0.99          0.99          0.99       845
 macro avg         0.99          0.99          0.99       845
 weighted avg      0.99          0.99          0.99       845
```

	Not obese	Obese
Not obese	445	6
Obese	2	392



Rispetto alla precedente valutazione non c'è stata alcuna variazione significativa, e i risultati sono anche qui ottimi, con un'accuratezza pari al 99%. Naturalmente anche per quanto riguarda la matrice di confusione e la curva precision-recall i risultati sono molto simili a quelli del 1° task, e dunque quasi perfetti.

Successivamente sono stati selezionati i seguenti Iperparametri ed è stato applicato la GridSearchCV con stratified Kfold cross-validation con split di valore 10:

```
parameters = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
              'C': [50, 10, 1.0, 0.1, 0.01]}
```

Di seguito vengono riportati i risultati del tuning degli Iperparametri assieme ai migliori Iperparametri:

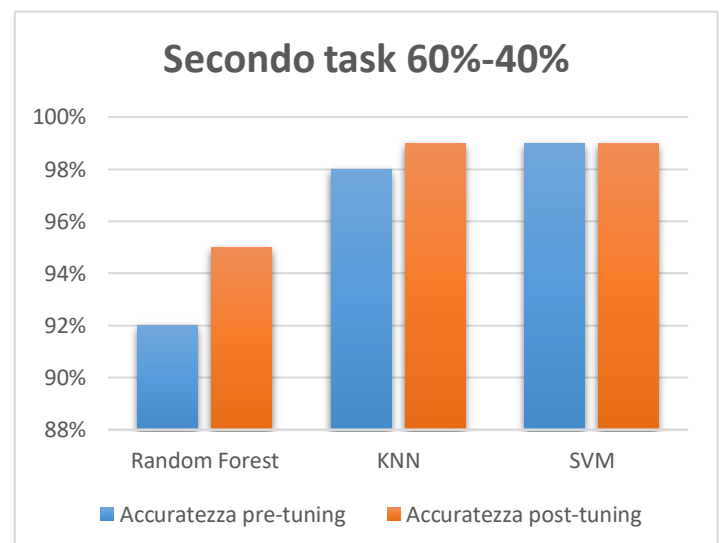
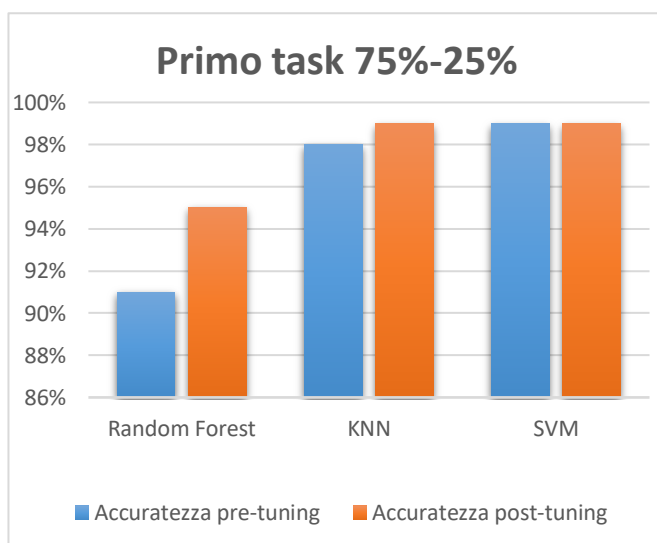
```
Best mean score (with cv): 0.988185 (Standard deviation: 0.001538) using {'C': 10, 'kernel': 'linear'}  
Train-set score           : 0.992  
Test-set score            : 0.999
```

Come per il primo task, sono stati selezionati come Iperparametri un C pari a 10 e un kernel di tipo lineare. Infatti, il modello ha lavorato quasi perfettamente su entrambi i set di dati e in entrambi i task, con risultati pressoché identici.

RISULTATI FINALI

I risultati finali che vengono fuori dai due differenti task di classificazione binaria sono i seguenti:

CLASSIFICATORI	PRIMO TASK 75%-25%		SECONDO TASK 60%-40%	
	ACCURATEZZA (PRE-TUNING)	ACCURATEZZA (POST-TUNING)	ACCURATEZZA (PRE-TUNING)	ACCURATEZZA (POST-TUNING)
RANDOM FOREST	91%	95%	92%	95%
KNN	98%	99%	98%	99%
SVM	99%	99%	99%	99%



INTERFACCIA

Dopo aver effettuato l'addestrato dei vari classificatori, è stato esportato il modello SVM (che ha l'accuratezza maggiore), e re-implementato all'interno dell'interfaccia, il cui scopo è predire una condizione di obesità in base ai dati inseriti dall'utente. Non avendo effettuato la classificazione multi-classe, è stata implementata anche una funzione per il calcolo delle fasce del BMI. Di seguito sono riportati due esempi:

```
Benvenuto/a! Ti dirò a quale livello di peso appartieni in base al tuo stile di vita.
Inoltre, ti restituirò il BMI associato!
Inserisci i seguenti valori:
-----
Inserisci il sesso: (1.Femmina 0.Maschio) 1
Inserisci l'età: 35
Inserisci l'altezza: 1.80
Inserisci il peso: 100
Hai familiari obesi o che lo sono stati? (0.No 1.Yes ) 1
Mangi spesso cibi molto calorici? (0.No 1.Sì) 1
Ogni quanto mangi vegetali? (1.Sempre 2.A volte 3.Raramente) 1
Inserisci il numero di pasti quotidiani: (0.Tra 1 e 2 1.Tre 2.Più di tre) 2
Fumi? (0.No 1.Sì) 0
Quanta acqua bevi giornalmente? (1.Meno di un litro 2.Tra uno e due litri 3.Più di due litri) 2
Quanto spesso mangi tra i pasti principali? (1.Sempre 2.Frequentemente 3.A volte 4.Raramente) 1
Conti le calorie che consumi durante il giorno? (0.No 1.Sì) 0
Quanto attività fisica svolgi settimanalmente? (1.Da 1 a 2 giorni 2.Da 3 a 4 giorni 3.Da 5 a 6 giorni 4.Nessuna attività fisica) 4
Quanto tempo passi sui dispositivi elettronici? (0.Da 0 a 2 ore 1.Da 3 a 5 ore 2.Più di 5 ore) 2
Bevi alcohol? (1.Non consumo 2.Raramente 3.Settimanalmente 4.Giornalmente) 4
Quale mezzo di trasporto usi per spostarti? (0.Automobile 1.Motorbike 2.Bici 3.Mezzo pubblico 4.A piedi) 0
Livello di obesità: obeso
Sei obeso di tipo 1
BMI: 30.094959824689553
```

Predizione per un individuo obeso

```
Benvenuto/a! Ti dirò a quale livello di peso appartieni in base al tuo stile di vita.
Inoltre, ti restituirò il BMI associato!
Inserisci i seguenti valori:
-----
Inserisci il sesso: (1.Femmina 0.Maschio) 1
Inserisci l'età: 34
Inserisci l'altezza: 1.75
Inserisci il peso: 70
Hai familiari obesi o che lo sono stati? (0.No 1.Yes ) 0
Mangi spesso cibi molto calorici? (0.No 1.Sì) 0
Ogni quanto mangi vegetali? (1.Sempre 2.A volte 3.Raramente) 2
Inserisci il numero di pasti quotidiani: (0.Tra 1 e 2 1.Tre 2.Più di tre) 2
Fumi? (0.No 1.Sì) 0
Quanta acqua bevi giornalmente? (1.Meno di un litro 2.Tra uno e due litri 3.Più di due litri) 2
Quanto spesso mangi tra i pasti principali? (1.Sempre 2.Frequentemente 3.A volte 4.Raramente) 2
Conti le calorie che consumi durante il giorno? (0.No 1.Sì) 0
Quanto attività fisica svolgi settimanalmente? (1.Da 1 a 2 giorni 2.Da 3 a 4 giorni 3.Da 5 a 6 giorni 4.Nessuna attività fisica) 1
Quanto tempo passi sui dispositivi elettronici? (0.Da 0 a 2 ore 1.Da 3 a 5 ore 2.Più di 5 ore) 2
Bevi alcohol? (1.Non consumo 2.Raramente 3.Settimanalmente 4.Giornalmente) 2
Quale mezzo di trasporto usi per spostarti? (0.Automobile 1.Motorbike 2.Bici 3.Mezzo pubblico 4.A piedi) 2
Livello di obesità: non obeso
Sei normopeso
BMI: 23.510204081632654
```

Predizione per un individuo non obeso

CONCLUSIONI

Il progetto è stato un banco di prova importante, nel quale si sono presentate numerose difficoltà riguardanti principalmente l'utilizzo di nuovi strumenti come Python, Protégé e Prolog. Nonostante ciò, sono stati raggiunti gli obiettivi principali, e si è tratto il meglio da questa esperienza. Inoltre, questo progetto nasce principalmente come task di classificazione ma potrebbe essere anche esteso come task di regressione. E' possibile poi estendere il progetto effettuando la classificazione multi-classe per predire le singole classi relative al BMI.