

Università di Torino

Dipartimento di

Informatica



Modellazione concettuale per il

Web Semantico

Relazione di Progetto

Classificazione per il dominio
Machine Learning

Studenti:
Michele Metta
Pietro Sangermano

Anno accademico 2021/2022

1) Motivazioni

Il machine Learning, da parecchi anni a questa parte, sta diventando sempre di più uno dei maggiori settori di ricerca nell'informatica e non.

Volendo fare alcuni esempi, uno di questi è la sua applicazione in biologia; difatti questo viene usato per riconoscere ad esempio le cellule tumorali nel sangue (ovviamente unito a delle tecniche di laboratorio) o ancora viene usato per l'individuazione di nuovi farmaci, partendo da una serie di composti, di cui un esempio tangibile è uno studio del MIT in questo senso nel caso della tubercolosi.

Un altro esempio che sta spopolando in questo periodo soprattutto nell'ambito della grafica è DLSS (Deep Learning Super Sampling) che è una tecnologia di rendering IA all'avanguardia che aumenta le prestazioni grafiche utile all'aumento di frame rate e alla nitidezza dei video e delle immagini.

Un ultimo esempio che di questi tempi è molto in voga e che ha generato più di qualche situazione imbarazzante è la tecnologia del deepfake. Essa, infatti, utilizza una speciale rete neurale detta GAN (generative Adversarial Network) che è in grado in base a uno speciale meccanismo di addestramento di simulare l'aspetto e le movenze facciali di una persona e poi, tramite appositi strumenti, adattarli ad un altro volto facendo sì che si pensi che il soggetto di un certo video sia una certa persona invece che un'altra.

L'ultimo esempio che si vuole portare, vuol far sentire ancora di più quanto il machine learning possa influire sulla nostra vita, in maniera ancora più schiacciante, è il progetto del Fraunhofer Institute per Optronics di Lemgo in Germania atto alla regolazione del traffico. Utilizzando algoritmi di machine learning di tipo DRL (Deep Reinforced Learning), molto utilizzato nei problemi di controllo, si è visto che applicandolo ai semafori ad un incrocio e abilitando il cambio di luce intelligente e predittivo hanno constatato che ci sono sensibili miglioramenti sotto diversi aspetti:

- Miglioramento del traffico del 10-15%;
- Diminuzione dell'inquinamento;
- Diminuzione dell'inquinamento acustico;
- Spostamenti più veloci per i viaggiatori (nello specifico Lemgo è una città universitaria piena di pendolari).

Adesso lo stesso team sta lavorando per una soluzione simile ma per i pedoni cercando di creare una mistura fra IA e un sistema embedded di sensori atti alla soddisfazione dei bisogni di chi attraversa.

Si spera che nei prossimi anni si possa ridurre del 30% il tempo in coda e di diminuire il numero di incidenti di attraversamento di circa il 25%.

Detto ciò si può constatare come non solo, come accennato prima, il machine learning sta diventato molto popolare nell'ambito della ricerca, ma inoltre esso sta già influenzando le nostre vite e continuerà a farlo in maniera sempre più impattante.

2) Requisiti

- a. Uno dei problemi più comuni del machine learning è capire come far sì che tutte le sue componenti si incastrino perfettamente rispetto al contesto del problema, cioè, dato un certo problema di machine learning (o task), capire quale algoritmo sia meglio utilizzare e di quali tipi di dati si ha bisogno per estrarre le giuste features.

Purtroppo porre ciò in questi termini è un po' riduttivo, in quanto è facile trovare più algoritmi che appartengono a categorie diverse e che risolvono perfettamente la nostra problematica.

A questo si aggiunge anche l'orientamento dell'utente verso un tipo di algoritmo rispetto ad un altro o la disponibilità di un certo dataset o di certe features, il che rende la scelta molto più complicata.

Per questo si vuole creare un'ontologia atta a modellare nei suoi aspetti più importanti il machine learning in modo da riuscire a mettere l'utente nelle condizioni di divincolarsi su cosa fare e in che modo farlo. Ovviamente qui non si ha la pretesa di creare qualcosa di completo e onnisciente (anche alla luce del natura fortemente personalizzabile della materia) ma di gettare una base che sia in grado di dare un indizio o una traccia da cui cominciare.

- b. Descrizione delle classi utilizzate nell'ontologia:
 - i. Algoritmo, superclasse rappresentante le tipologie di algoritmi possibili nell'apprendimento automatico. È composta da 3 sottoclassi:
 - 1. Algoritmo_ML, indica le tipologie di algoritmi di machine learning attualmente più utilizzati. Essi sono:
 - a. Albero: algoritmo machine learning con struttura ad albero binario;
 - b. Distanza: algoritmo di machine Learning basati sulle misure di distanza fra esempi di training ed esempi di test;
 - c. Lineare: algoritmi di machine learning basati sulla combinazione lineare di parametri atta all'approssimazione della migliore retta/iperpiano possibile, che ci consenta di separare linearmente 2 classi di oggetti(o più)
 - d. Ensemble Learning: algoritmi di machine learning basati sull'utilizzo di weak learners in grado di ottenere prestazioni, dal punto di vista dell'accuratezza, molto elevate. Esso si divide in 2 sottoclassi: Bagging e Boosting.
 - e. Probabilistico: algoritmi di machine learning basati sulla stima di probabilità. Si divide in 2 sottoclassi: generativo e discriminativo.
 - f. Regole: algoritmo di machine learning basati sul ragionamento logico tramite costruzione di regole.
 - g. Rete Neurale: algoritmi di machine Learning che approssimano il comportamento delle sinapsi e dei neuroni umani nell'elaborazione delle informazioni e nell'apprendimento. Ha una sottoclasse che è la Rete neurale profonda (in cui i livelli della rete sono molto maggiori di quelli della superclasse)
 - 2. Apprendimento: specifica il tipo di apprendimento che perpestra l'algoritmo. Ha 2 sottoclassi: Supervisionato e Non Supervisionato (le 2 tipologie di apprendimento).
 - 3. Trasformatore features: algoritmi di supporto all'apprendimento automatico in grado di trasformare feature di un certo tipo in un altro. Si compone di 3 sottoclassi:
 - 1) Trasformatore feature Booleana, la cui sottoclasse (unica effettiva operazione) è la calibrazione booleana.
 - 2) Trasformatore feature categorica, con 3 sottoclassi: Binarizzazione, calibrazione categorica e raggruppamento
 - 3) Trasformatore feature quantitativa, con 3 sottoclassi: discretizzazione, Normalizzazione e Thresholding.

- ii. Dataset: struttura dati semi-strutturata contenente esempi di training e di test utilizzati dagli algoritmi di machine learning. Ne esistono di svariato tipo, noi abbiamo considerato:
 - 1. Audio
 - 2. Firma Digitale
 - 3. Immagine
 - 4. Impronta Digitale
 - 5. Numerico_Categorico
 - 6. Retina
 - 7. Video
- iii. Feature: caratteristiche distintive estratte da un dataset, atte ad agevolare e descrivere sinteticamente il learning problem. Ne esistono principalmente 3 tipi:
 - 1. Numerico
 - 2. Categorico
 - 3. Booleano
- iv. Libreria: insieme delle librerie più usate nell'ambito del machine learning.
- v. Linguaggio di programmazione: classe rappresentante quelli che sono i linguaggi di programmazione più utilizzati nell'implementazione delle librerie all'interno delle quali sono implementati gli algoritmi di machine learning.
- vi. Task: sono i learning problems da risolvere tramite un algoritmo di apprendimento automatico.
Sono di 2 tipi:
 - 1) Descrittivi: atti a descrivere i dati in ordine di trovare un pattern nascosto negli stessi (nell'ontologia abbiamo descritto 2 di questi, ovvero il clustering e l'anomaly detection).
 - 2) Predittivi: in cui il problema è predire la classe di appartenenza o l'andamento numerico di un dato oggetto (anche qui ne abbiamo descritti 2, la classificazione e la regressione).

- c. Gli utenti a cui è rivolta la nostra ontologia è un qualcuno a cui il mondo dell'apprendimento automatico non è del tutto sconosciuto, si presuppone che essi sappiano cosa sia un dataset o delle features e che ne conoscano il dominio in maniera più o meno superficiale. Difatti ci si aspetta che gli utilizzatori conoscano almeno una delle classi in maniera abbastanza approfondita e che sappiano almeno superficialmente cosa siano le altre.

In realtà, però nulla nega ad un utente completamente inesperto sull'argomento di consultare l'ontologia, perché ciò lo porterebbe a scoprire, anche se in modo sommario, le componenti del machine learning e di come le sue componenti sia connesse.

Riassumendo, ci aspettiamo che l'ontologia possa essere utilizzata sia da utenti alquanto esperti in cerca di un punto di partenza o di una conferma per un loro eventuale lavoro e sia da utenti inesperti che vogliono scrostare la superficie di questa materia.

3) Descrizione del dominio

Il Machine Learning è un ramo dell'intelligenza artificiale che si preoccupa di eseguire uno studio sistematico degli algoritmi e dei sistemi che migliorano la propria conoscenza e le proprie performance con l'esperienza. L'apprendimento automatico quindi, comprende un insieme di tecniche che permettono di poter addestrare un algoritmo in modo tale che questo possa apprendere dai dati.

Si può praticamente dire che il machine learning non è altro che una serie di meccanismi atti ad infondere conoscenza ad una "macchina" in modo da migliorare le sue prestazioni su un certo task.

Il padre fondatore del machine learning è sicuramente Alan Turing, che già nella prima metà del novecento, palesò la necessità di realizzare algoritmi specifici per la realizzazione di macchine che fossero in grado di apprendere, anche se le sue parole vennero accolte con scetticismo.

Il processo di apprendimento si compone di diverse fasi:

- Studio del dominio del problema e raccolta dei dati:
 - Questa parte (forse la più importante), è atta allo studio del nostro problema di learning e a capire quali dati sia meglio sfruttare per creare il nostro modello di learning. Per la prima parte abbiamo diverse possibilità: classificazione, regressione, anomaly detection e così via. Per quanto riguarda la scelta dei dati, essa dipende molto dalla disponibilità dell'utente. Infatti, i dati non sono esplicitamente legati ad un certo tipo di problema ma alcuni sono preferibili ad altri a seconda dell'algoritmo che si vuole utilizzare e soprattutto, rispetto alle features che si vuole usare.
- Estrazione delle feature:
 - Una feature è una caratteristica discriminante di un tipo di dato. Queste vengono estratte dai dati e servono principalmente a descrivere in maniera sintetica ma estremamente esplicativa i dati, in modo da non dover addestrare i modelli su dati grezzi e quindi avere un'alta imprecisione (potenzialmente) sugli stesso. Da ogni tipo di dato si possono estrarre differenti features, sta a noi capire cosa ci è più utile sia in base al dominio del problema e sia in base al grado di espressività delle features che desideriamo estrarre.
- Addestramento del modello:
 - Addestrare il modello è un processo pressoché automatico, ciò a cui si deve stare attenti è la parametrizzazione dei vari metodi che ancora una volta dipende da ciò che vogliono ottenere.
Il processo di addestramento serve a creare un modello basato su un set di esempi che riesce ad approssimare il comportamento o rappresentare un qualcosa in modo da poter poi prendere decisioni in maniera intelligente, basandosi su cosa quello che è stato appreso.
- Sperimentazione (o testing) del modello:
 - Il testing del modello è un processo molto delicato che se va male potrebbe comportare una rianalisi del dominio o la scelta di altre features. Tecnicamente si testa il modello somministrandogli un set di dati che lui non conosce e studiando come esso si comporta.

Se il nostro modello creato nella fase di addestramento è stato abbastanza preciso nel catturare il comportamento della soluzione al nostro problema di learning, allora la sperimentazione vuol dire che sarà andata bene, altrimenti avremo risultati scarsi.

Da sottolineare come per risultati “buoni” non si intende la precisione nel 100% dei casi, soprattutto con problemi più complessi, ma che sia migliore del 50% (altrimenti in un problema binario, sarebbe come tirare una moneta).

Fissata questa struttura generale, negli anni si sono avvicinati diversi metodi di machine learning come il k-means per il clustering o la one class SVM per l'anomaly detection. Uno di questi metodi che ultimamente sta andando per la maggiore sono le reti neurali, essi sono complessi sistemi di apprendimento in grado di trovare applicazione in moltissimi problemi di learning. Da loro è nata tutta una branca della materia chiamata deep learning che studia l'utilizzo di reti neurali dense, strumenti estremamente potenti che però comportano un grosso carico computazionale e richiedono una grossa quantità di dati (problema ormai superato dall'avvento dei Big Data).

Riferimenti bibliografici:

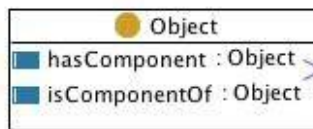
- <https://www.innovationpost.it/2018/02/14/intelligenza-artificiale-deep-learning-e-machine-learning-quali-sono-le-differenze/>
- https://it.wikipedia.org/wiki/Big_data
- [Machine learning \(apprendimento automatico\) - Cos'è e come funziona? \(intelligenzaartificiale.it\)](http://intelligenzaartificiale.it)

4) Documentazione del dominio

Per creare la documentazione dell'ontologia è stato utilizzato **WIDOCO**. Esso estende il framework **LODE** che viene utilizzato per descrivere le classi, le proprietà e tutti gli altri dati dell'ontologia. Il file generato da Widoco è presente nella cartella chiamata **Machine-Learning-Ontology**. Una volta entrati al suo interno basta cliccare sul file chiamato “index-en.html” per ottenere la documentazione dell'ontologia.

a. Requisiti

Per modellare la classe ensemble learning abbiamo usato il content pattern Composition che è la specificazione di un altro pattern: PartOf.



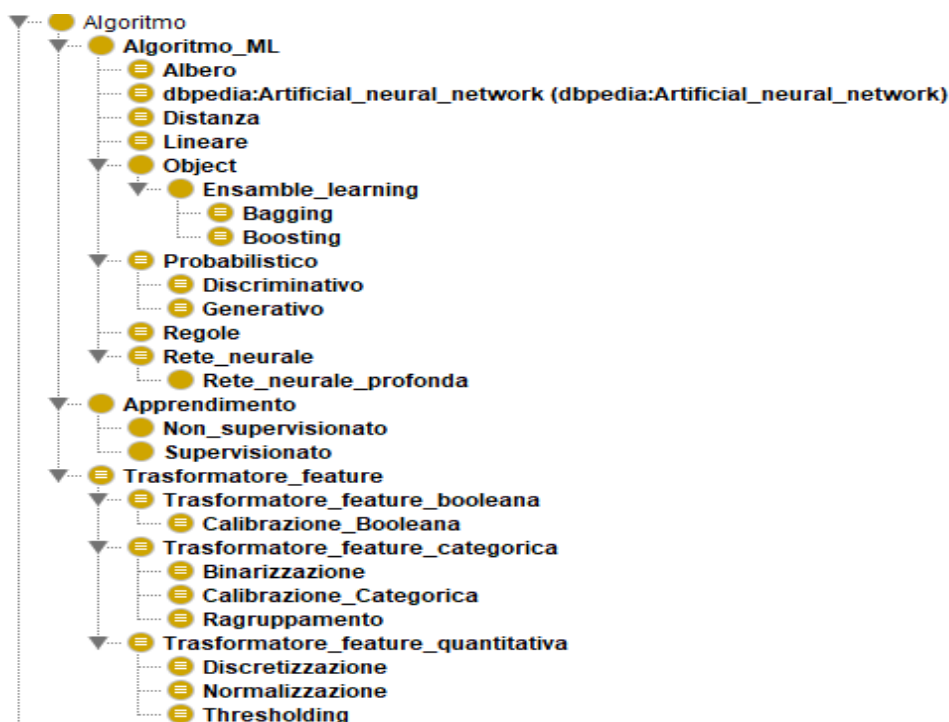
L'obiettivo di questo pattern è rappresentare (in modo non transitivo) degli oggetti che siano componenti di altri oggetti e sia oggetti che invece sono composti da altri oggetti. Nel nostro caso è stato utilizzato poiché l'ensemble learning fa uso di una composizione di vari algoritmi in forma 'depotenziata', in modo da boostare le prestazioni di questi ultimi mediante la loro combinazione.

5) Tassonomia

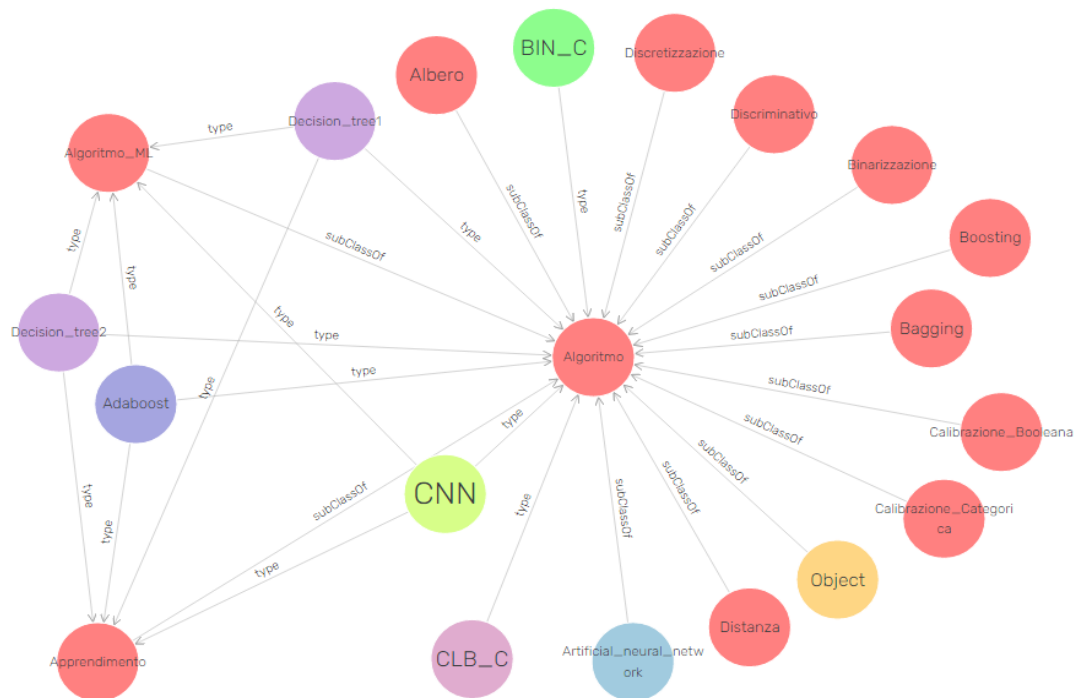
a. La classe: Algoritmo

La tassonomia della classe algoritmo va a modellare tutti gli aspetti prettamente procedurali riguardanti il machine learning. Nel nostro dominio la classe algoritmo ha 3 sottoclassi:

- **Algoritmo_ML**, le cui sottoclassi sono: Albero, Distanza, Lineare, Ensemble Learning (distinto in Bagging e Boosting), Probabilistico (distinto in generativo e discriminativo), Regole e Rete Neurale (la cui sottoclasse è Rete neurale profonda).
- **Apprendimento**, diviso in 2 sottoclassi: Supervisionato e Non Supervisionato.
- **Trasformatore Feature**, diviso in 3 sottoclassi: Trasformatore Feature Booleano (con sottoclasse calibrazione Booleana), Trasformatore Feature categorico (con sottoclassi Binarizzazione, Calibrazione categorica e Raggruppamento) e Trasformatore Feature quantitativo (con 3 sottoclassi Discretizzazione, Normalizzazione e Thresholding)



Visualizzazione a grafo della classe Algoritmo (utilizzando GRAPH-DB):



Adesso entriamo più nello specifico delle suddette classi. Algoritmo_ML ha 7 sottoclassi e alcune di queste a loro volta sono specificate ulteriormente da altre sottoclassi.

Abbiamo gestito queste ulteriori sottoclassi andandole a definire tramite delle caratteristiche uniche che le contraddistinguono nell'equivalentTo. Portiamo un paio di esempi:

1. Distanza

Gli algoritmi basati sulla distanza basano il loro funzionamento sul calcolo della distanza fra gli esempi di test e i centroidi(o simili) trovati durante l'addestramento. Abbiamo quindi deciso di creare la data property **distanze** di un tipo creato ex-novo che abbiamo chiamato **misure**, il quale può essere di 3 tipi (semplificando): Euclidea, Manhattan e Mahalanobis .

Annotations: **misure**

Annotations

[rdfs:comment](#)

Tipi di distanza adottabili

Description: **misure**

Datatype Definitions

{"Euclidea", "Malhanobis", "Manhattan"}

La definizione della classe Distanza è la seguente:

Description: Distanza

Equivalent To

(utilizza some Quantitativa)
and (distanze some misure)

SubClass Of

Algoritmo_ML

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

k_means

k_medoid

knn

Come ulteriore appunto, si è voluto specificare che il tipo di features che utilizza questo tipo di algoritmo sono le features quantitative. A questo proposito, vogliamo introdurre la object property **utilizza**, in quanto è presente in ognuna delle definizioni dei tipi di algoritmi.

Essa mette in relazione algoritmo_ML e Feature e implica, appunto, l'utilizzo di un certo tipo di feature da parte del primo.

Annotations: utilizza

Annotations

[rdfs:comment](#)

Implica l'utilizzazione di un certo tipo di feature da parte di un algoritmo di machine learning

Characteristics: utilizza

☐ Functional

☐ Inverse functional

☐ Transitive

☐ Symmetric

☐ Asymmetric

☐ Reflexive

☐ Irreflexive

Description: utilizza

Equivalent To

SubProperty Of

Inverse Of

Domains (intersection)

Algoritmo_ML

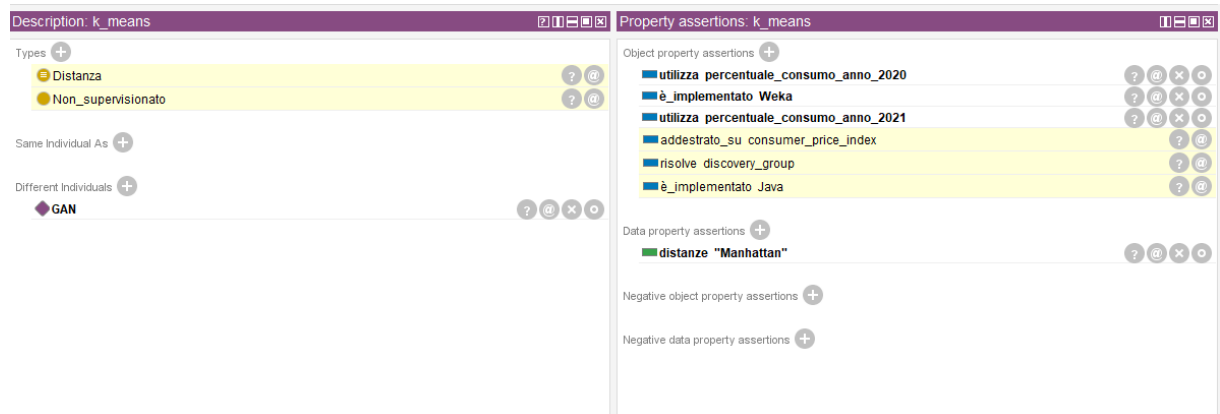
Ranges (intersection)

Feature

Disjoint With

SuperProperty Of (Chain)

Per concludere vediamo cosa inferisce il reasoner per l'individuo k_means:



Come si può vedere siccome k_means usa la distanza di Manhattan, i 2 **utilizza** fanno riferimento a 2 features quantitative. Grazie a queste due informazioni, il k_means viene inferito come algoritmo basato sulla distanza.


2. Rete neurale


Per definire la classe **Rete neurale** abbiamo invece fatto uso delle regole SWIRL (sia per la suddetta e sia per la sua sottoclasse **Rete neurale profonda**). Abbiamo pensato che la definizione operativa per una rete neurale, fosse la presenza stessa dei neuroni e che ciò che contraddistinguesse la superclasse dalla sottoclasse fosse il numero dei livelli della rete (che ne indica appunto la profondità). Creando due data property (neuroni di tipo booleana che indica la presenza o meno di neuroni nell'algoritmo e numero_di_livelli di tipo integer) abbiamo strutturato le 2 swirl nel seguente modo:


- **Rete Neurale:** $ML:Quantitativa(?f) \wedge ML:Algoritmo_ML(?a) \wedge ML:utilizza(?a, ?f) \wedge ML:neuroni(?a, ?e) \wedge swrlb:equal(?e, true) \wedge ML:numero_di_livelli(?a, ?n) \wedge swrlb:greaterThan(?n, 0) \rightarrow ML:Rete_neurale(?a)$
- **Rete Neurale Profonda:** $ML:Quantitativa(?f) \wedge ML:Algoritmo_ML(?a) \wedge ML:utilizza(?a, ?f) \wedge ML:neuroni(?a, ?e) \wedge swrlb:equal(?e, true) \wedge ML:numero_di_livelli(?a, ?n) \wedge swrlb:greaterThan(?n, 10) \rightarrow ML:Rete_neurale_profonda(?a)$


Da quello che si può notare la swirl per entrambe le classi è praticamente identica poiché diciamo che se un algoritmo utilizza features quantitative, ha neuroni e il numero di livelli è maggiore strettamente di 0, allora è una rete neurale ma andando a fissare una soglia arbitraria (noi l'abbiamo fissata a 10 ma difatti non c'è un numero esatto) allora la rete neurale è anche profonda.


Come ultimo appunto, abbiamo voluto aggiungere l'allineamento con dbpedia per la classe rete neurale tramite l'equivalent class poiché lo abbiamo inteso come una spiegazione di cosa sia una rete neurale e non da cosa sia derivato, come si potrebbe intendere tramite subclass.

Equivalent To 


 **dbpedia:Artificial_neural_network** (dbpedia:Artificial_neural_network)


SubClass Of 



 **Algoritmo_ML**

General class axioms 

SubClass Of (Anonymous Ancestor)

 **Rete_neurale**

Instances 


 belief_network
 multilayer_perceptron


La seconda sottoclasse che descriviamo è la classe **Apprendimento**. Con essa si vuole fare riferimento alla classica distinzione fra **apprendimento supervisionato e non supervisionato**. Per modellare questo concetto ci siamo rifatti al come si comportano gli algoritmi in base a questa definizione e abbiamo associato che la differenza si basa sul fatto che appunto l'addestramento venga effettuato su un dataset etichettato o non etichettato. Pertanto, abbiamo creato 2 swirl, una l'opposta dell'altra per modellare ciò:


- Supervisionato:** $ML:Algoritmo_ML(?a) \wedge ML:Dataset(?d) \wedge ML:etichettato(?d, ?e) \wedge swrlb:equal(?e, true) \wedge ML:addestrato_su(?a, ?d) \rightarrow ML:Supervisionato(?a)$
- Non Supervisionato:** $ML:Algoritmo_ML(?a) \wedge ML:Dataset(?d) \wedge ML:etichettato(?d, ?e) \wedge swrlb:equal(?e, false) \wedge ML:addestrato_su(?a, ?d) \rightarrow ML:Non_supervisionato(?a)$


Unica aggiunta che si vuole fare è sulla object property `addestrato_su`, creata tramite il meccanismo di property chain che mette in relazione algoritmo e dataset tramite feature.


Description: addestrato_su


Equivalent To 


SubProperty Of 


Inverse Of 


Domains (intersection) 


 **Algoritmo_ML**

Ranges (intersection) 

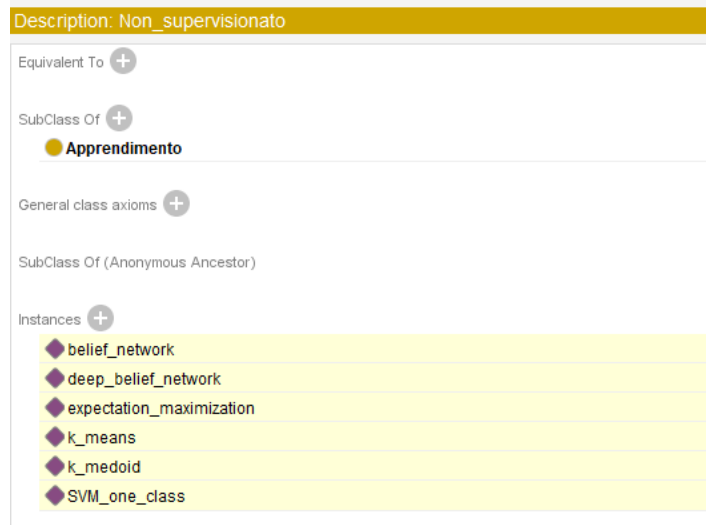
 **Dataset**

Disjoint With 

SuperProperty Of (Chain) 

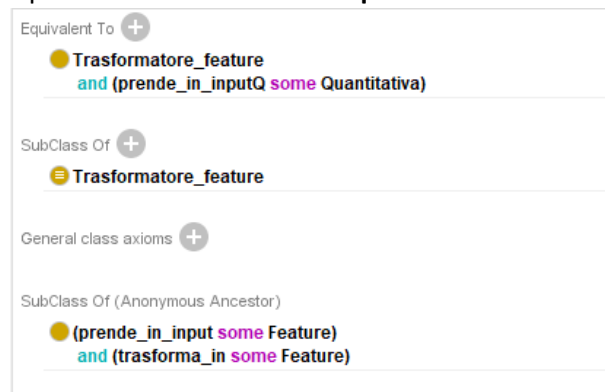
 **utilizza o è estratta** SubPropertyOf: `addestrato_su`

Mostriamo adesso le inferenze fatte dal reasoner per la classe **non supervisionato**:

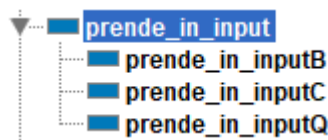


L'ultima sotto classe di algoritmo è **Trasformatore Feature**. Questa classe rappresenta un insieme di algoritmi di supporto all'apprendimento automatico e servono a trasformare un certo tipo di feature in un altro tipo.

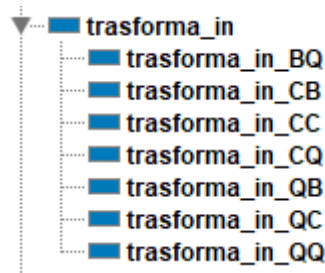
Prendiamo ad esempio **Trasformatore feature quantitativa**:



Come si può osservare esso è un trasformatore feature che prende in input features Quantitative e a proposito di questa object property facciamo una piccola digressione:



Abbiamo creato una super property chiamata **prende_in_input** che ha come dominio Trasformatore feature e range Feature e dopodichè abbiamo creato 3 sotto-proprietà il cui range cambia in base al tipo di feature presa in input appunto. Allo stesso modo abbiamo creato la proprietà **trasforma_in** creando una sotto proprietà per ogni coppia significativa di feature:



Detto questo prendiamo l'individuo MinMax un algoritmo di normalizzazione che effettua una trasformazione da quantitativa a quantitativa, pertanto:

The screenshot shows a Semantic Web editor interface. On the left, under 'Equivalent To', there is a logical expression: $(\text{prende_in_inputQ some Quantitativa}) \text{ and } (\text{trasforma_in_QQ some Quantitativa})$. Below this, under 'SubClass Of', is the class `Trasformatore_feature_quantitativa`. Further down, under 'General class axioms', there are two axioms: $\text{Trasformatore_feature and } (\text{prende_in_inputQ some Quantitativa})$ and $(\text{prende_in_input some Feature}) \text{ and } (\text{trasforma_in some Feature})$. Under 'Instances', there are two instances: `MinMax` and `statistical_normalization`. On the right, under 'Object property assertions', there are four assertions: `prende_in_inputQ altezza`, `trasforma_in_QQ altezza_tra_0_e_1`, `prende_in_input altezza`, and `trasforma_in altezza_tra_0_e_1`. The bottom right panel shows 'Data property assertions', 'Negative object property assertions', and 'Negative data property assertions', all of which are currently empty.

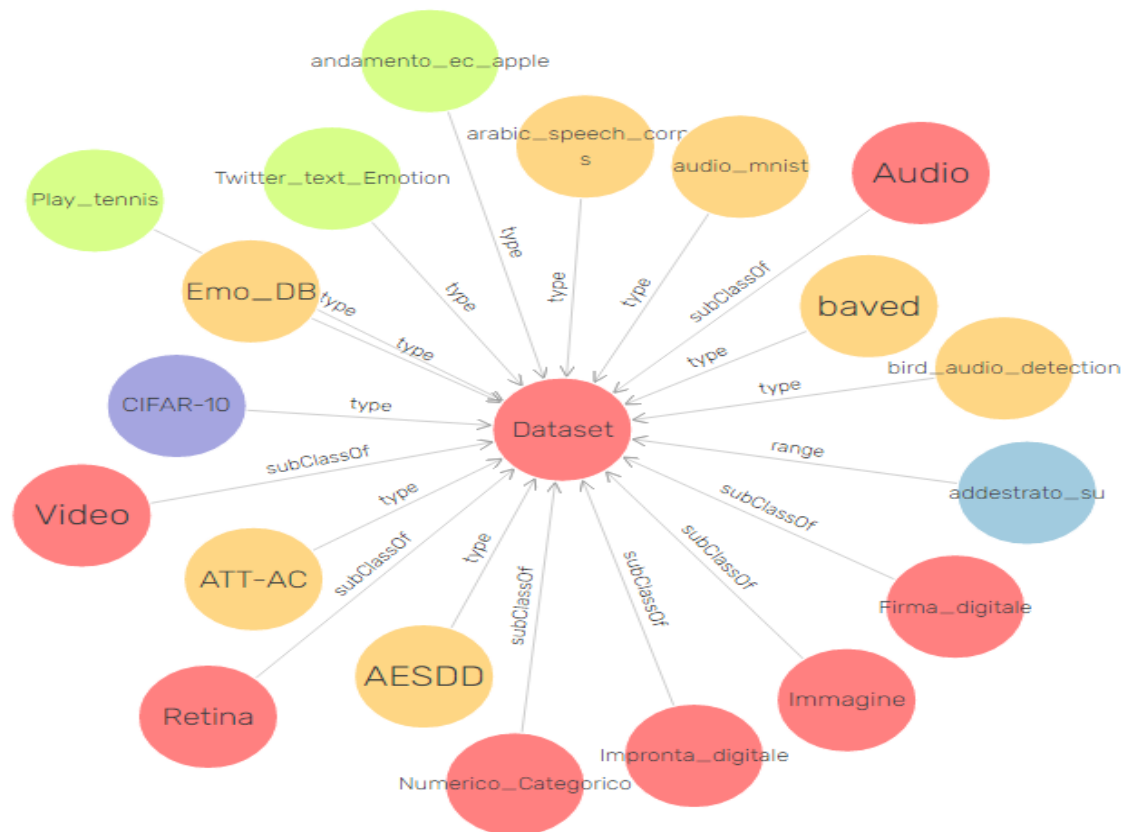
b. La classe: Dataset

La classe Dataset modella alcune delle tipologie di dataset utilizzate per la risoluzione di determinati learning task.

La classe Dataset ha 7 sottoclassi: Audio, Firma Digitale, immagine, Impronta digitale, Numerico Categorico, Retina e Video.



Visualizzazione a grafo della classe Dataset:



Bisogna specificare 2 cose su questa classe: la prima è che il dataset impronta digitale è stato definito tramite enumerazione dei datasets più conosciuti in circolazione.

Equivalent To **{mnist_dataset , n2n , nist , socothing}**

SubClass Of **Dataset**

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

◆ mnist_dataset
◆ n2n
◆ nist
◆ socothing

La seconda riguarda la data property booleana **etichettato**, che ci dice se gli esempi di un certo dataset ad esempio sono accompagnati dalla propria classe target e la object property **viene_estratta** che mette in relazione Dataset e Feature (da un Dataset viene estratta una certa Feature), inoltre esiste anche la sua inversa che è chiamata **è_estratta**). Si porta un esempio di uno di questi dataset:

Types

Numerico_Categorico

Same Individual As

Different Individuals

Object property assertions

viene_estratta nome_arbitro

viene_estratta home_team

viene_estratta tiri_squadra_fuori_casa

viene_estratta away_team

viene_estratta risultato_fine_primo_tempo

viene_estratta goals_totali_squadra_fuori_casa

viene_estratta tiri_squadra_casa

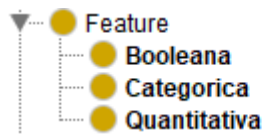
è_utilizzato_per risultati_partite_di_calcio

Data property assertions

etichettato true

c. La classe: Feature

La classe feature riassume i tipi di feature utilizzabili dagli algoritmi di apprendimento automatico. La classe feature ha 3 sottoclassi: Booleana, Categorica e Quantitativa.



Visualizzazione a grafo della classe Feature:



Si riporta come esempio la definizione della classe Categorica:

description: Categorica

Equivalent To +

SubClass Of +

● Feature

General class axioms +

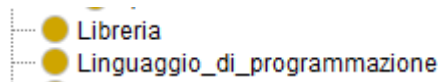
SubClass Of (Anonymous Ancestor)

Instances +

- away_team
- clima
- grandezza_regione
- home_team
- Industrializzazione
- Intervalli_di_altezza
- intervallo_peso
- nome_arbitro
- obeso
- parola
- parola_TW
- risultato_fine_primo_tempo
- tag_parola
- temperatura
- temperatura_acqua
- umidità

d. Le classi: Libreria e linguaggi di programmazione

Le classi Libreria e linguaggi di programmazione servono a modellare le librerie che implementano gli algoritmi di machine learning e i linguaggi di programmazione in cui essi sono scritti.



Su queste due classi è stata implementata una proprietà transitiva chiamata **è_implementato** con la seguente logica:

“Un algoritmo è implementato in una libreria la quale è implementata in un linguaggio di programmazione”. Riportiamo un esempio:

dbpedia:Artificial_neural_network (dbpedia:Artificial_neural_network) ? @

● Non_supervisionato ? @

⇒ Rete_neurale ? @

Same Individual As +

Different Individuals +

utilizza popolazione_bf

è_implementato Neuroph

utilizza pil

addestrato_su pop_cw

risolve appartenenza_fascia_tasse

è_implementato Java

Data property assertions +

neuroni true

numero_dilivelli 4

e. La classe: Task

La classe task modella le tipologie di learning problems che possono essere risolti tramite un algoritmo di machine learning.

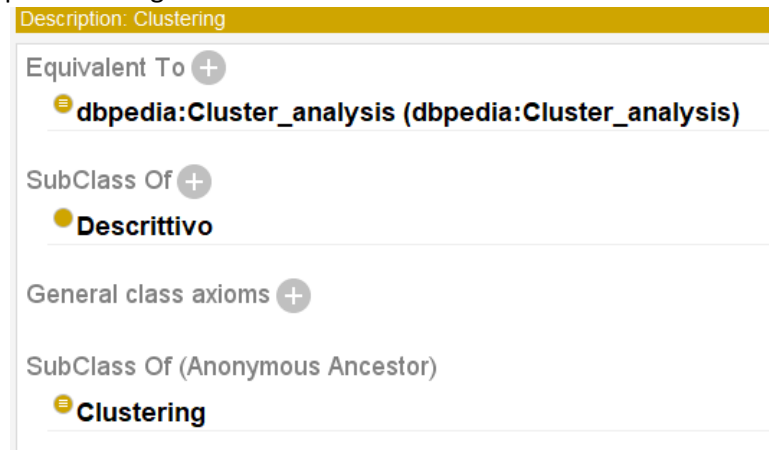
La classe **Task** ha 2 sottoclassi: **Descrittivo** (con 2 sottoclassi anomaly detection e clustering) e **Predittivo** (con 2 sottoclassi Regressioni e Classificazione, dove quest'ultima è composta a sua volta da 2 sottoclassi Binaria e multiclasse).



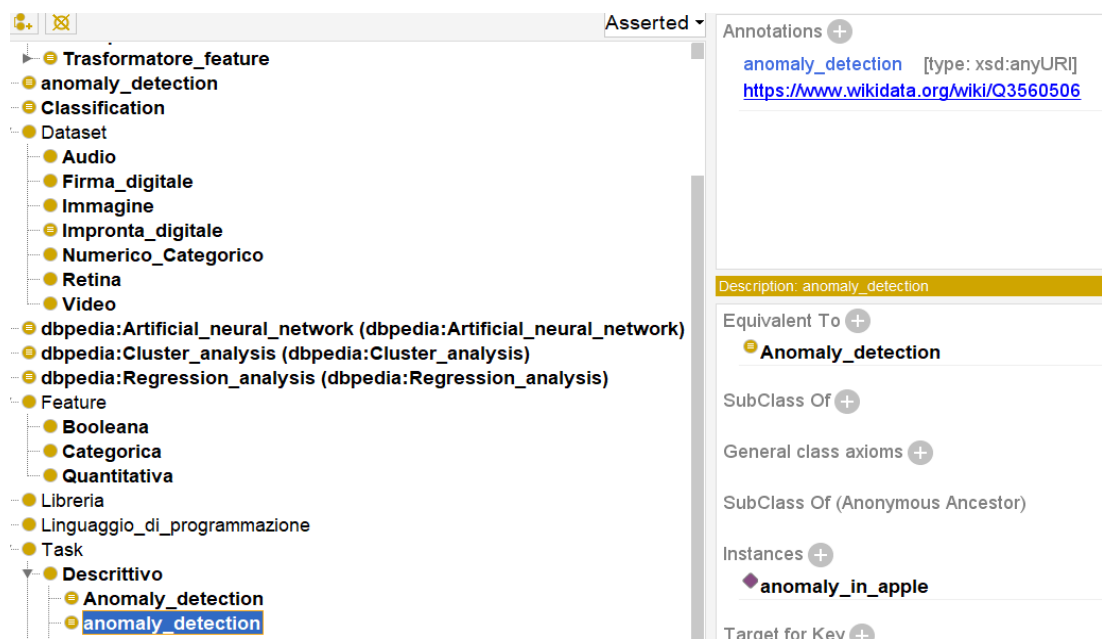
Visualizzazione a grafo della classe Task:



Per quanto riguarda la classe **Clustering**, essa è stata allineata con la classe Cluster_analysis di Dbpedia, questo è stato fatto modificando l'equivalent to della classe Clustering come mostrato in questa immagine:



La classe **Anomaly_detection** è stata invece allineata con la classe anomaly_detection presente su Wikidata. Per fare ciò è stato inserito nelle annotazioni nella classe anomaly_detection l'URI facente riferimento a tale classe presente su Wikidata come mostrato nella figura sottostante:



Dopodichè nell'equivalent to della nostra classe Anomaly_detection è stata inserita la classe anomaly_detection nominata precedentemente:

- Retina
- Video
- dbpedia:Artificial_neural_network (dbpedia:Artificial_neural_network)
- dbpedia:Cluster_analysis (dbpedia:Cluster_analysis)
- dbpedia:Regression_analysis (dbpedia:Regression_analysis)
- Feature
- Booleana
- Categorica
- Quantitativa
- Libreria
- Linguaggio_di_programmazione
- Task
- Descrittivo
 - Anomaly_detection

Description: Anomaly_detection

Equivalent To +

- anomaly_detection

SubClass Of +

- Descrittivo

General class axioms +

SubClass Of (Anonymous Ancestor)

- Anomaly_detection

Instances +

Per quanto riguarda invece la classe **Classificazione**, essa è stata allineata con la classe Classification presente su Wikidata. Il metodo utilizzato per l'allineamento è identico a quello fatto per la classe anomaly_detection detta pocanzi.

Immagini che mostrano l'allineamento appena citato:

Asserted

- Classification
- Dataset
- Audio
- Firma_digitale
- Immagine
- Impronta_digitale
- Numerico_Categorico
- Retina
- Video
- dbpedia:Artificial_neural_network (dbpedia:Artificial_neural_network)
- dbpedia:Cluster_analysis (dbpedia:Cluster_analysis)
- dbpedia:Regression_analysis (dbpedia:Regression_analysis)
- Feature
- Booleana
- Categorica
- Quantitativa
- Libreria
- Linguaggio_di_programmazione
- Task
- Descrittivo
 - Anomaly_detection
 - anomaly_detection
 - Clustering
 - dbpedia:Cluster_analysis (dbpedia:Cluster_analysis)
- Predittivo
 - Classification
 - Classificazione

Annotations +

Classification [type: xsd:anyURI]
<https://www.wikidata.org/wiki/Q13582682>

Description: Classification

Equivalent To +

- Classificazione

SubClass Of +

General class axioms +

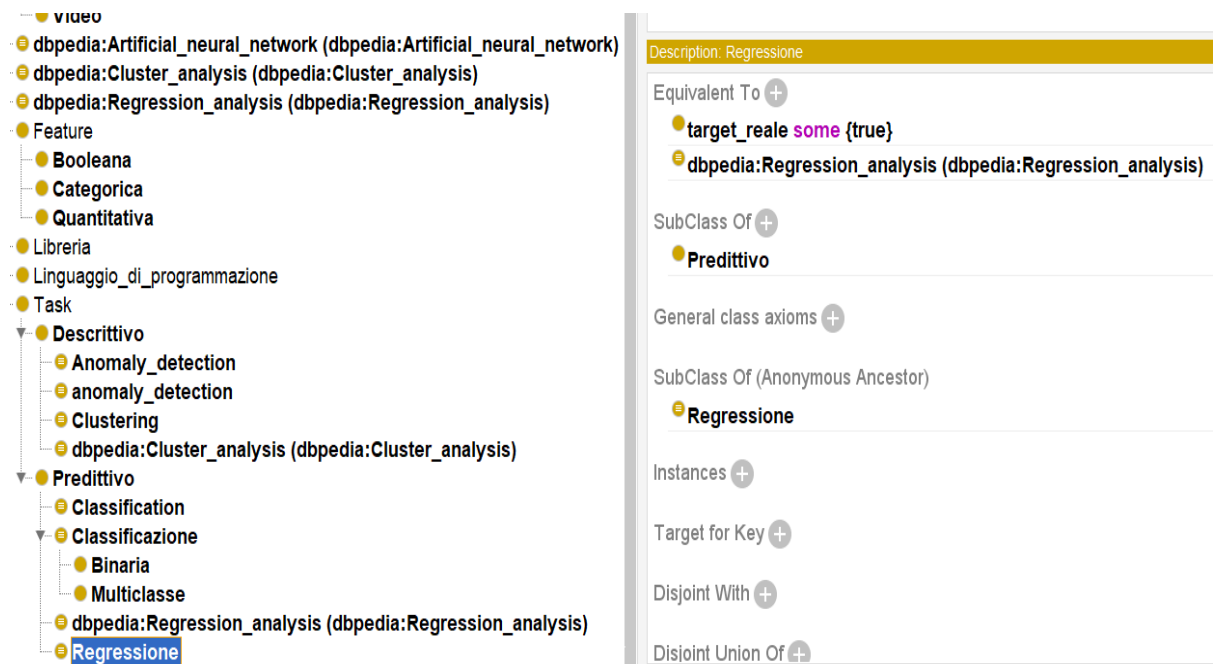
SubClass Of (Anonymous Ancestor)

Instances +

Target for Key +

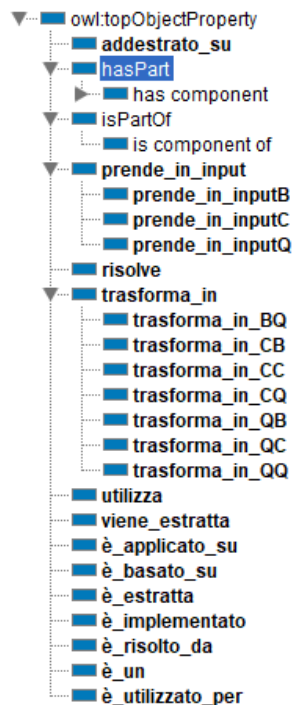
Disjoint With +

Disjoint Union Of +



f. Object Property

Attraverso la definizione delle varie object properties sono state create le associazioni tra le varie classi dell'ontologia. Esse sono le seguenti:



Nota di merito va a **risolve**, una property chain che mette in relazione la classe **Algoritmo_ML** e la classe **Task** attraverso la classe **dataset**, utilizzando l'altra property chain **addestrato_su**, esplicitata prima, e la proprietà **è_utilizzato_per** che mette in relazione **dataset** e **Task**. La proprietà **risolve** ha un'inversa che è chiamata **è_risolta_da**.

Description: resolve
Equivalent To
SubProperty Of
owl:topObjectProperty
Inverse Of
è_risolto_da
Domains (intersection)
Algoritmo_ML
Ranges (intersection)
Task
Disjoint With
SuperProperty Of (Chain)
addestrato_su o è_utilizzato_per SubPropertyOf: resolve

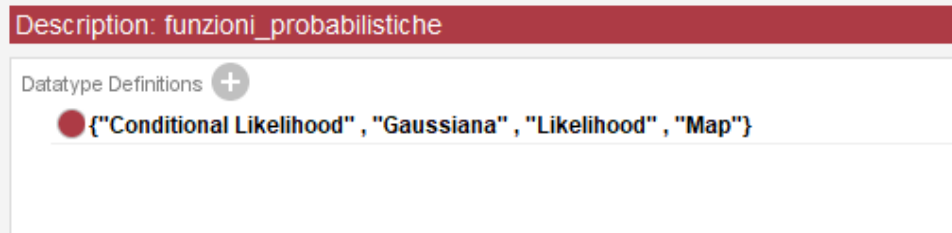
g. Data property

Qui vengono presentate le data properties utilizzate per arricchire la nostra ontologia e, per certi versi, definire molte delle nostre classi:

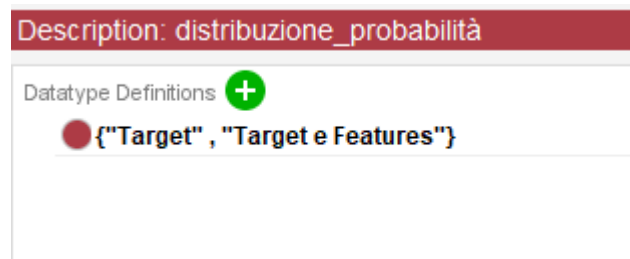


Anche qui bisogna fare due precisazioni:

- Massimizzazione, il cui tipo è chiamato **funzioni probabilistiche**, è stato creato ex novo e indica su quale tipo di funzione un algoritmo probabilistico esegue la massimizzazione.



- Restituzione di probabilità: è associato sempre agli algoritmi di probabilità ed esso specifica cosa restituisce l'algoritmo dopo essere stato addestrato. Anche qui abbiamo creato un nuovo tipo chiamato **distribuzione di probabilità**, definito come nell'immagine qui sotto.



h. Individui

Di seguito vengono mostrati alcuni degli individui da noi introdotti (non vengono messi tutti per l'elevata mole di individui creati):

- | | |
|---------------------------------------|-----------------------------------|
| ◆ AESDD | ◆ drive_dataset |
| ◆ altezza | ◆ Emo_DB |
| ◆ altezza_tra_0_e_1 | ◆ energia_segnaie |
| ◆ andamento_ec_apple | ◆ Equal_depth_partitioning |
| ◆ andamento_mercato_finanziario_Apple | ◆ Equal_width_partitioning |
| ◆ anomaly_in_apple | ◆ Euclidean |
| ◆ appartenenza_fascia_tasse | ◆ expectation_maximization |
| ◆ arabic_speech_corpus | ◆ fdd |
| ◆ ATT-AC | ◆ GAN |
| ◆ audio_mnist | ◆ gdp |
| ◆ away_team | ◆ goals_totali_squadra_casa |
| ◆ baved | ◆ goals_totali_squadra_fuori_casa |
| ◆ belief_network | ◆ google_open_image |
| ◆ BIN_C | ◆ gpds |
| ◆ bird_audio_detection | ◆ grandezza_regione |
| ◆ boss_dataset | ◆ GROUPING |
| ◆ C++ | ◆ harris_corner_detection |
| ◆ Caffe | ◆ hidden_markov_model |
| | ◆ home_team |
| | ◆ hopfield_nets |
| | ◆ icdar_2011 |
| | ◆ id_rid |
| | ◆ imagenet |
| | ◆ independent_component_analysis |
| | ◆ Industrializzazione |
| | ◆ Intervalli_di_altezza |
| | ◆ intervallo_peso |
| | ◆ iris_flower_dataset |
| | ◆ isolation_forest |
| | ◆ Java |
| | ◆ Joy_emotion_text_recognition |
| | ◆ k_means |
| | ◆ k_medoid |
| | ◆ Keras |

6) Swrl

Oltre alle regole swirl precedentemente citate (ovvero le due per le reti neurali e le altre due per l'apprendimento supervisionato e non supervisionato) abbiamo creato altre regole che sono mostrate di seguito:

- a. *Se una certa istanza di Task t ha un certo numero di clusters n e questo n è maggiore di 0, allora quell'istanza è un task di Clustering:*

*ML:Task(?t) ^ ML:numero_clusters(?t, ?n) ^ swrlb:greaterThan(?n, 0) ->
ML:Clustering(?t)*

- b. *Se un task di classificazione ha un numero di classi esattamente uguale a 2, allora quello è un task di classificazione binaria:*

*ML:Classificazione(?c) ^ ML:numero_classi(?t, ?n) ^ swrlb:equal(?n, 2) ->
ML:Binaria(?t)*

- c. *Se un task di classificazione ha un numero di classi maggiore di 2, allora quello è un task di classificazione multiclasse:*

*ML:Classificazione(?c) ^ ML:numero_classi(?t, ?n) ^ swrlb:greaterThan(?n, 2) ->
ML:Multiclasse(?t)*

- d. *Se un algoritmo utilizza una certa feature e questo algoritmo ha dei weak learners, allora anche i weak learners utilizzeranno quella feature:*

*ML:Algoritmo(?a) ^ ML:utilizza(?a, ?f) ^ ML:ha_weak_learner(?e, ?a) ->
ML:utilizza(?e, ?f)*

7) GRAPH DB

Per creare le query ed eseguirle in modo da poter estrarre le informazioni contenute nel knowledge graph è stata utilizzata la Linked data platform Graph DB.

QUERY 1):

La query seleziona tutti gli algoritmi di apprendimento supervisionato presenti nell'ontologia.

```
▼ 1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2
3 select *
▼ 4 WHERE {
5     ?inst a ML:Supervisionato.
6 }
7
```

Risultati query 1):

	inst
1	ML:CNN
2	ML:Decision_tree1
3	ML:Decision_tree2
4	ML:GAN
5	ML:Random_forest
6	ML:SVM
7	ML:hidden_markov_model
8	ML:knn
9	ML:linear_regression
10	ML:multilayer_perceptron
11	ML:regressione_logistica
12	ML:som
13	ML:Adaboost
14	ML:SVM_1
15	ML:SVM_2

QUERY 2):

La query seleziona tutti gli algoritmi di apprendimento non supervisionato presenti nell'ontologia.

```
1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2
3 select *
4 WHERE {
5     ?inst a ML:Non_supervisionato.
6 }
7
```

Risultati query2):

	inst
1	ML:SVM_one_class
2	ML:belief_network
3	ML:deep_belief_network
4	ML:expectation_maximization
5	ML:k_means
6	ML:k_medoid

QUERY 3:

La query seleziona tutti i modelli lineari presenti nell'ontologia.

```
1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6
7 select distinct ?a
8 WHERE {
9     ?a ML:utilizza ?q.
10     ?a ML:combinazione_lineare true.
11     ?q rdf:type ML:Quantitativa.
12 }
13
```

Risultati query 3):

	Algoritmi_lineari
1	ML:Least_mean_squares
2	ML:SVM
3	ML:SVM_one_class
4	ML:linear_regression
5	ML:SVM_1
6	ML:SVM_2

QUERY 4:

La query seleziona tutti gli algoritmi di clustering presenti.

```
▼ 1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6
7
8 select distinct ?a
▼ 9 WHERE {
10     ?a ML:risolve ?t.
11     ?t ML:numero_clusters ?n.
12     FILTER(?n>0).
13 }
```

Risultati query 4):

	algoritmi_di_clustering
1	ML:k_means
2	ML:deep_belief_network
3	ML:k_medoid

QUERY 5:

La query seleziona tutti gli algoritmi presenti nell'ontologia che permettono di trasformare una certa feature da quantitativa a categorica.

```
▼ 1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6
7
8 select ?Nome_Algoritmo
▼ 9 WHERE {
10     ?Nome_Algoritmo a ML:Trasformatore_feature_quantitativa.
11     ?Nome_Algoritmo ML:trasforma_in_QC ?f.
12     ?f a ML:Categorica.
13 }
14
```

Risultati query 5):

	Nome_Algoritmo
1	ML:Equal_depth_partitioning
2	ML:Equal_width_partitioning

QUERY 6:

la query restituisce tutti i tasks che sono risolti da una certa rete neurale profonda che è addestrata su un certo dataset.

```
▼ 1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6
7 select ?task ?algoritmo ?dataset
▼ 8 WHERE {
9     ?task ML:è_risolto_da ?algoritmo.
10     ?algoritmo ML:addestrato_su ?dataset.
11     ?algoritmo ML:numero_dilivelli ?n.
12     FILTER(?n>10)
13 }
14
```

Risultati query 6):

	task	algoritmo	dataset
1	ML:riconoscimento_immagini	ML:CNN	ML:CIFAR-10
2	ML:paesi_con_pil_simile	ML:deep_belief_network	ML:gdp
3	ML:appartenenza_fascia_tasse	ML:deep_belief_network	ML:gdp
4	ML:paesi_con_pil_simile	ML:deep_belief_network	ML:pop_cw
5	ML:appartenenza_fascia_tasse	ML:deep_belief_network	ML:pop_cw
6	ML:riconoscimento_immagini	ML:som	ML:CIFAR-10

QUERY 7):

La query restituisce tutti gli algoritmi presenti in una certa libreria che è implementata in Python.

```
1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6
7 select ?algoritmo ?libreria
8 WHERE {
9     ?algoritmo ML:è_implementato ?libreria.
10    ?libreria a ML:Libreria.
11    ?libreria ML:è_implementato ML:Python.
12
13 }
14
```

Risultati query 7):

	algoritmo	libreria
1	ML:CNN	ML:Keras
2	ML:GAN	ML:Keras
3	ML:SVM	ML:Scikit_learn
4	ML:SVM_one_class	ML:Scikit_learn
5	ML:expectation_maximization	ML:Scikit_learn
6	ML:linear_regression	ML:Scikit_learn
7	ML:Adaboost	ML:Scikit_learn
8	ML:SVM_1	ML:Scikit_learn
9	ML:SVM_2	ML:Scikit_learn

Query 8):

La seguente query restituisce i nomi dei sistemi operativi sui quali è possibile utilizzare la libreria Scikit-learn, questa informazione è presa direttamente da DBPEDIA.

```
1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6 PREFIX dbo: <http://dbpedia.org/ontology/>
7 PREFIX dbr: <http://dbpedia.org/resource/>
8 select ?os
9 WHERE {
10     SERVICE <https://dbpedia.org/sparql>
11     {
12         dbr:Scikit-learn dbo:operatingSystem ?os.
13     }
14 }
```

Risultati query 8):

	OS
1	http://dbpedia.org/resource/Microsoft_Windows
2	http://dbpedia.org/resource/MacOS
3	http://dbpedia.org/resource/Linux

Query 9):

Questa query restituisce i nomi di tutti i linguaggi di programmazione in cui è implementata la libreria Scikit-learn, questa informazione è presa direttamente da DBPEDIA.

```
▼ 1 PREFIX ML:<http://www.semanticweb.org/miky9/ontologies/2022/3/untitled-ontology-34#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6 PREFIX dbo: <http://dbpedia.org/ontology/>
7 PREFIX dbr: <http://dbpedia.org/resource/>
8 select ?linguaggi_di_programmazione
▼ 9 WHERE {
10     SERVICE <https://dbpedia.org/sparql>
▼11     {
12         ?concept dbo:wikiPageID 33490859;
13         dbo:programmingLanguage ?linguaggi_di_programmazione
14     }
```

























Risultati query 9):

	linguaggi_di_programmazione
1	http://dbpedia.org/resource/C_(programming_language)
2	http://dbpedia.org/resource/C++
3	http://dbpedia.org/resource/Python_(programming_language)
4	http://dbpedia.org/resource/Cython


























8) Omeka S

E' stata utilizzata la piattaforma open source Omeka S, che permette di creare e pubblicare collezioni digitali, per sviluppare delle schede descrittive utilizzando come vocabolario l'ontologia creata. Quello che è stato fatto inizialmente è stato caricare l'ontologia creata per poterla utilizzare come vocabolario e dopodichè sono stati creati i resources template, successivamente sono stati creati gli item sets e infine ognuno di essi è stato mappato su un certo template.

I resources template creati sono mostrati in questa immagine:



































Resource templates			Import	A
1	of 1	< > 1-8 of 8		Lab
Label		Class	Owner	
Albero	  	Albero	Machine Learning	
Algoritmo Machine Learning	  	Algoritmo_ML	Machine Learning	
Apprendimento Non Supervisionato	  	Non_supervisionato	Machine Learning	
Apprendimento Supervisionato	  	Supervisionato	Machine Learning	
Base Resource	  		[no owner]	
Dataset	  	Dataset	Machine Learning	
Feature	  	Feature	Machine Learning	
Task	  	Task	Machine Learning	

Gli item sets creati sono i seguenti:

Item sets				A
1	of 1	< > 1-6 of 6	Q Advanced search	Created De
Batch actions  Go				
<input type="checkbox"/> Title		Class	Owner	Created
<input type="checkbox"/>  Apprendimento Non Supervisionato	  	Non_supervisionato	Machine Learning	Jun 20, 2022
<input type="checkbox"/>  Apprendimento Supervisionato	  	Supervisionato	Machine Learning	Jun 20, 2022
<input type="checkbox"/>  Task	  	Task	Machine Learning	Jun 20, 2022
<input type="checkbox"/>  Feature	  	Feature	Machine Learning	Jun 20, 2022
<input type="checkbox"/>  Dataset	  	Dataset	Machine Learning	Jun 20, 2022
<input type="checkbox"/>  Algoritmi di Machine Learning	  	Algoritmo_ML	Machine Learning	Jun 20, 2022

Successivamente sono stati creati 17 diversi items che in una fase immediatamente successiva sono stati mappati su uno o più item sets precedentemente creati.

Gli items creati sono mostrati in questa immagine:

Items				
	Probabilistico		Probabilistico	Machine Learning Jun 20, 2022
	Ensamble Learning		Ensamble_learning	Machine Learning Jun 20, 2022
	Lineare		Lineare	Machine Learning Jun 20, 2022
	Distance based algorithm		Distanza	Machine Learning Jun 20, 2022
	Rete neurale		Rete_neurale	Machine Learning Jun 20, 2022
	Video		Video	Machine Learning Jun 20, 2022
	Firma digitale		Firma_digitale	Machine Learning Jun 20, 2022
	Impronta digitale		Impronta_digitale	Machine Learning Jun 20, 2022
	Audio		Audio	Machine Learning Jun 20, 2022
	Regressione		Regressione	Machine Learning Jun 20, 2022
	Classificazione		Classificazione	Machine Learning Jun 20, 2022
	Clustering		Clustering	Machine Learning Jun 20, 2022
	Anomaly detection		Anomaly_detection	Machine Learning Jun 20, 2022
	Booleana		Booleana	Machine Learning Jun 20, 2022
	Categorica		Categorica	Machine Learning Jun 20, 2022
	Quantitativa		Quantitativa	Machine Learning Jun 20, 2022
	Albero		Albero	Machine Learning Jun 20, 2022

Ad ogni item è stato:

- 1) Assegnato un titolo.
- 2) Assegnata una descrizione che fornisce informazioni generali sull'item.
- 3) Assegnato un certo insieme di proprietà derivanti direttamente dall'ontologia creata (ad esempio specificando nel caso di un algoritmo di machine learning quali siano i tasks che quell'algoritmo rappresentato da quel determinato item riesce a risolvere, quali sono le features che quell'algoritmo utilizza, ecc..)
- 4) Assegnato eventualmente un insieme di data property derivanti sempre dall'ontologia creata.
- 5) Assegnato uno o più item sets di riferimento (Ad esempio ci sono algoritmi di machine learning che possono eseguire sia l'apprendimento supervisionato che non supervisionato e quindi per questo motivo faranno riferimento sia all'item set dell'apprendimento supervisionato e sia a quello non supervisionato).
- 6) Associato un media (in genere un'immagine).

In questa immagine è possibile visionare l'item Ensemble Learning:

ITEMS

Ensamble Learning

Metadata

Linked resources

Class

Ensamble_learning

Title

Ensamble Learning

Description

Algoritmi che sfruttano la combinazione di altri algoritmi di machine learning per boostare le prestazioni.

addestrato_su

Video

Audio

Firma digitale

Impronta digitale

risolve

Regressione

Classificazione

Clustering

utilizza

Quantitativa

Categorica

Booleana

ha_weak_learner

E' un modello addestrato singolarmente che può essere di qualsiasi tipo.

Per decidere la predizione finale, l'algoritmo di Ensamble Learning usa le predizioni di ogni weak learner.

Has Part

Lineare

Albero

Distance based algorithm

ID

22

Visibility

Public

Item sets

Algoritmi di Machine Learning

Apprendimento Supervisionato

Apprendimento Non Supervisionato

Sites

Apprendimento Automatico

Created

Jun 20, 2022

Owner

Machine Learning

Media (1)

ensemble-learning.png

Infine sono state create le pagine del sito web e ogni item set è stato assegnato a ciascuna pagina, infine è stata creata la navigazione. Le pagine create sono 8 e sono le seguenti:

- 1) Index: è la prima pagina che compare appena si accede al sito ed è quella che mostra semplicemente l'insieme di tutti gli items creati.
- 2) Algoritmi di Machine Learning: è la pagina che contiene tutti gli algoritmi di apprendimento automatico, a ciascuno di essi è assegnata una certa immagine rappresentativa.
- 3) Features: tipi di features in genere utilizzate per gli algoritmi di apprendimento automatico.
- 4) Datasets: alcuni dei tipi di datasets sui quali vengono addestrati gli algoritmi.
- 5) Tasks: insieme di tasks principali risolti mediante l'utilizzo di algoritmi di machine learning.

E infine sono state create 3 pagine contenenti i risultati di 3 queries semantiche create mediante l'ausilio dello strumento interno ad Omeka S che permette di poter prelevare items con caratteristiche specifiche:

- 6) La prima query è quella che restituisce tutti gli algoritmi di ML che utilizzano features categoriche.
- 7) La seconda query è quella che invece restituisce tutti gli algoritmi che eseguono l'Anomaly Detection utilizzando anche features quantitative.
- 8) La terza ed ultima query è quella che restituisce tutti i datasets che vengono utilizzati per il clustering e che contengono features sia quantitative che categoriche.

Immagine che racchiude tutte le sezioni del sito web:

Apprendimento Automatico

	Algoritmi di Machine Learning	Features	Datasets	Tasks	Algoritmi di ML che utilizzano features categoriche	Algoritmi che eseguono l'Anomaly Detection utilizzando anche features quantitative	Datasets utilizzati per il clustering cor features quantitative e categoriche
Index							

Qui di seguito vengono mostrate le pagine contenenti i risultati delle 3 queries semantiche citate precedentemente:

Pagina 6):

Apprendimento Automatico

Algoritmi di Machine Learning

Index

Features

Datasets

Tasks

Algoritmi di ML che utilizzano features categoriche

Algoritmi che eseguono l'Anomaly Detection utilizzando anche features quantitative

Datasets utilizzati per il clustering con features quantitative e categoriche


Search

Q

Algoritmi di ML che utilizzano features categoriche


Ensamble Learning

Algoritmi che sfruttano la combinazione di altri algoritmi di machine learning per boostare le prestazioni.



Albero

Algoritmo di Machine Learning che ha una struttura ad albero binario.



Pagina 7):

Apprendimento Automatico

Algoritmi di Machine Learning

Index

Features

Datasets

Tasks

Algoritmi di ML che utilizzano features categoriche

Algoritmi che eseguono l'Anomaly Detection utilizzando anche features quantitative

Datasets utilizzati per il clustering con features quantitative e categoriche


Search

Q

Algoritmi che eseguono l'Anomaly Detection utilizzando anche features quantitative


Probabilistico

Algoritmo di Machine Learning basato sulla stima di probabilità.



Lineare

Algoritmi di Machine Learning atti a trovare l'iperpiano/retta che separa linearmente due classi.



Apprendimento Automatico

Datasets utilizzati per il clustering con features quantitative e categoriche

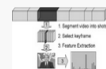
Search

Q

Datasets utilizzati per il clustering con features quantitative e categoriche

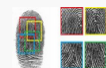
Video

Dataset contenente cortometraggi, solitamente di ugual durata, di svariato genere.



Impronta digitale

Dataset che contiene immagini ritraenti impronte digitali.



Audio

Dataset che contiene tracce audio di vario genere.

