

Disturbo dello Spettro Autistico

Gruppo di lavoro

- Mattia Marinelli, 737220, m.marinelli36@studenti.uniba.it
- Kristi Dashaj, 738055, k.dashaj@studenti.uniba.it

Link GitHub:

<https://github.com/kristiMcBacon/iCon.git>

AA 2022-2023

INDICE:

INTRODUZIONE:

1. DATASET:

1.1. Descrizione dataset (dominio):

1.2 Osservazione grafica dei dati:

2. ONTOLOGIA:

2.1 Analisi Dominio:

2.1.1 Classi:

2.1.2 Object property:

2.1.3 Data property:

2.1.4 Individuals:

2.2 Software per la realizzazione dell'Ontologia:

3. QUERY:

3.1 DL Query:

3.2 OwlReady:

4. APPRENDIMENTO SUPERVISIONATO:

4.1 Decisioni di progetto:

4.2 Metriche di valutazione:

4.3 Selezione delle Feature:

4.4 Preprocessing dei Dati:

4.4.1: Dummification

4.4.2: Smote

4.5 Divisione dei Dati:

4.6 K-Nearest Neighbors (KNN):

4.6.1 Decisioni di progetto:

4.6.2 Ottimizzazione numero di vicini:

4.6.3 Addestramento e predizione:

4.6.4 Valutazione Finale:

4.7 Random Forest:

4.7.1 Decisioni di progetto:

4.7.2 Ottimizzazione scelta numero alberi decisionali:

4.7.3 Valutazione finale

4.8 Support Vector machines:

4.8.1. Decisioni di progetto

4.8.2 Ottimizzazione parametro gamma:

4.8.3 Valutazione Finale:

4.9 Neural Network:

4.9.1 Decisioni di progetto:

4.9.2 Creazione modello e addestramento:

4.9.3 Valutazione Finale:

4.10 Conclusioni:

5 CLUSTERING:

5.1 Decisioni di progetto:

5.2 K-means:

5.3 Valutazione Finale:

6 CONCLUSIONE

6.1 Sviluppi futuri

INTRODUZIONE:

Questo progetto nasce con l'idea di predire una diagnosi di spettro autistico sfruttando un dataset disponibile online. A tal fine, sono state implementate metodologie di apprendimento supervisionato e non supervisionato. Inoltre, è stata sviluppata anche un'ontologia di riferimento che permette di adottare un quadro formale della realtà oggetto di studio.

Il disturbo dello spettro autistico (ASD) è una condizione del neurosviluppo. Sfortunatamente, i tempi di attesa per una diagnosi di ASD sono lunghi e le procedure non sono convenienti. L'impatto economico dell'autismo e l'aumento del numero di casi di ASD in tutto il mondo rivela un urgente bisogno di sviluppare metodi di screening efficaci e facilmente implementabili. Pertanto, uno screening ASD efficiente in termini di tempo e accessibile è imminente per aiutare gli operatori sanitari e informare le persone se devono perseguire una diagnosi clinica formale.

1. DATASET:

Proponiamo un set di dati relativo allo screening dell'autismo dei bambini piccoli, degli adolescenti e degli adulti, tratto dal sito www.archive.ics.uci.edu, che contiene caratteristiche influenti da utilizzare per l'analisi, in particolare per determinare i tratti autistici e migliorare la classificazione dei casi di ASD. In questo set di dati, registriamo dieci caratteristiche comportamentali più altre caratteristiche individuali che si sono dimostrate efficaci nel rilevare i casi di ASD dai controlli nella scienza del comportamento.

1.1. Descrizione dataset (dominio):

Questo dataset si basa su una raccolta longitudinale di 854 soggetti. A ciascun soggetto del database è stato sottoposto un test composto da 21 feature.

Se un soggetto segna un punteggio ('screening_score') maggiore di 6 è alta la possibilità che questo presenti dei tratti dell'ASD.

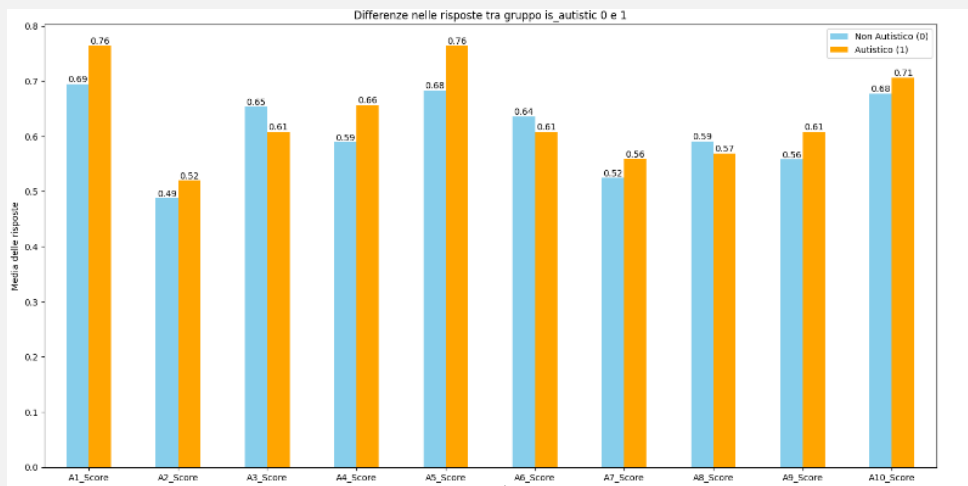
In questo set di dati, registriamo dieci caratteristiche comportamentali più altre caratteristiche individuali che si sono dimostrate efficaci nel rilevare i casi di ASD dai controlli nella scienza del comportamento.

Feature	Type	Description
age	Integer	Anni d'età
gender	Binary (0 / 1)	Mascho o Femmina (1=M, 0=F)
ethnicity	Integer (1 - 10)	Elenco delle etnie comuni in formato testo
jaundice	Binary (0 / 1)	Born with jaundice: Se il caso è nato con ittero (colorito giallastro)
is_autistic	Binary (0 / 1)	Is_Autism: valore booleano che indica se il soggetto è affetto da autismo
PDD_parent	Binary (0 / 1)	Family member with ASD: Se un membro della famiglia è affetto da autismo
test_compiler	Integer (1 - 4)	Who completing the test: 1) Family member, 2) Health Care Professional, 3) Self, 4) Others
country_of_res	String	Country of residence: nazione di residenza del paziente
used_app_before	Binary (0 / 1)	Used the screening app before: Se l'utente ha utilizzato un'app di screening
A1_Score	Binary (0 / 1)	Question 1 Answer: Tuo/a figlio/a ti guarda quando chiami il suo nome?
A2_Score	Binary (0 / 1)	Question 2 Answer: È facile per te avere un contatto visivo con tuo/a figlio/a?
A3_Score	Binary (0 / 1)	Question 3 Answer: Tuo/a figlio/a indica per indicare ciò che vuole? (ad esempio un gioco che non riesce e a raggiungere)
A4_Score	Binary (0 / 1)	Question 4 Answer: Tuo/a figlio/a punta a condividere interesse con te?
A5_Score	Binary (0 / 1)	Question 5 Answer: Tuo/a figlio/a finge? (ad esempio prendersi cura delle bambole, parlare su un telefono giocattolo)
A6_Score	Binary (0 / 1)	Question 6 Answer: Tuo/a figlio/a segue dove stai guardando?
A7_Score	Binary (0 / 1)	Question 7 Answer: Se tu o qualcun altro nella famiglia è visibilmente agitato, tuo/a figlio/a mostra segni di voler confortarli? (ad esempio, accarezzando i capelli, abbracciandoti)
A8_Score	Binary (0 / 1)	Question 8 Answer: Tuo/a figlio/a mostra interesse a partecipare a giochi di gruppo con altri bambini?
A9_Score	Binary (0 / 1)	Question 9 Answer: Tuo/a figlio/a usa semplici gesti (ad esempio il saluto)
A10_Score	Binary (0 / 1)	Question 10 Answer: Tuo/a figlio/a fissa il nulla senza uno motivo apparente?
screening_score (result)	Integer (0-10)	Screening Score: Il punteggio finale ottenuto sulla base dell'algoritmo di punteggio del metodo di screening utilizzato. Questo è stato calcolato in modo automatizzato
class/asd	Binary (0 / 1)	class/asd: Con screening score ≤ 6 no tratti asd, con screening score > 6 tratti asd

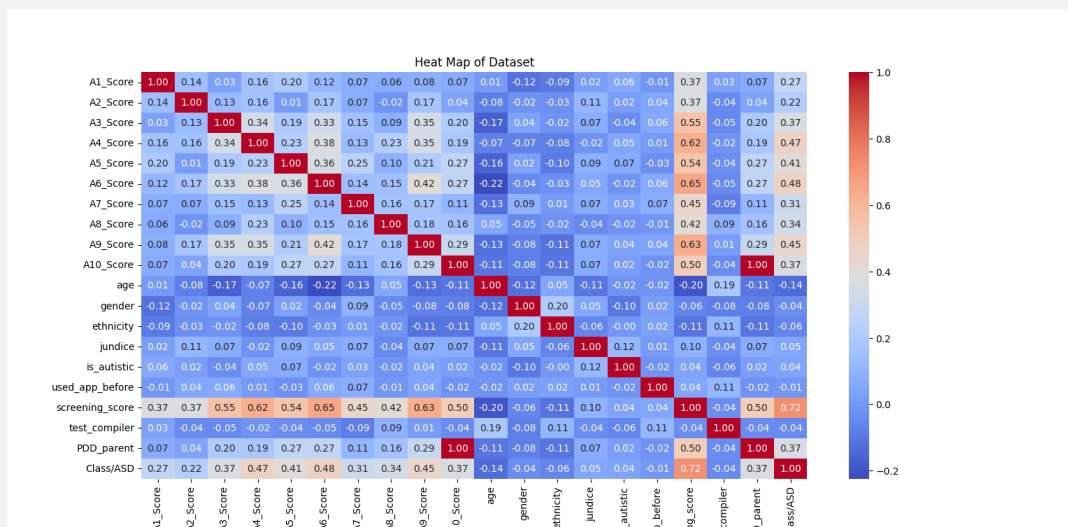
1.2 Osservazione grafica dei dati:

Abbiamo svolto delle osservazioni grafiche che ci permettessero di valutare la correlazione dei dati e soprattutto per capire in che modo la diagnosi fosse influenzata da essi. Qui di seguito riportiamo alcuni dei grafici che abbiamo realizzato in linguaggio python:

- Il primo grafico descrive la differenza nella media delle risposte tra chi si è definito affetto da autismo e chi no.



- Il secondo grafico mostra invece una matrice di correlazione, che è una rappresentazione tabulare delle correlazioni tra variabili. È una rappresentazione grafica che aiuta a comprendere come le diverse variabili in un dataset numerico sono correlate tra loro. La correlazione misura quindi la relazione statistica tra due variabili, indicando se e come variano insieme.



2. ONTOLOGIE:

Un'ontologia è un modello di conoscenza che consente di rappresentare in modo univoco le informazioni relative ad un determinato dominio. Durante l'analisi del dominio si indentificano i concetti principali, le relazioni e le proprietà che caratterizzano il dominio.

Successivamente questi concetti, relazioni e proprietà possono essere formalizzate mediante un linguaggio di rappresentazione formale come ad esempio OWL (Ontology Web Language).

2.1 Analisi Dominio:

Dato il set di dati utilizzato e le informazioni sugli attributi, abbiamo deciso di dividere gli attributi del dataset in:

-*"ciò che deve essere rappresentato"*: gli attributi fondamentali e cruciali che devono essere rappresentati per catturare le informazioni essenziali dal dataset. Nel contesto dell'autismo, include: domande, test e pazienti

-*"ciò che caratterizza ciò che deve essere rappresentato"*: le proprietà delle entità rappresentate, (nel caso del paziente ad esempio possono essere età, sesso...)

-*"ciò che può essere ricavato"*: ad esempio gli attributi 'Screening Score' e 'Used the screening app before'

-*"ciò che può essere scartato"*: ad esempio l'attributo 'Country of residence'.

2.1.1 Classi:

Analizzando il dominio abbiamo deciso di rappresentare come classi dell'ontologia:

• **Paziente**: La classe Paziente, rappresenta l'insieme dei pazienti sottoposti al test. A questa classe abbiamo associato diverse proprietà:

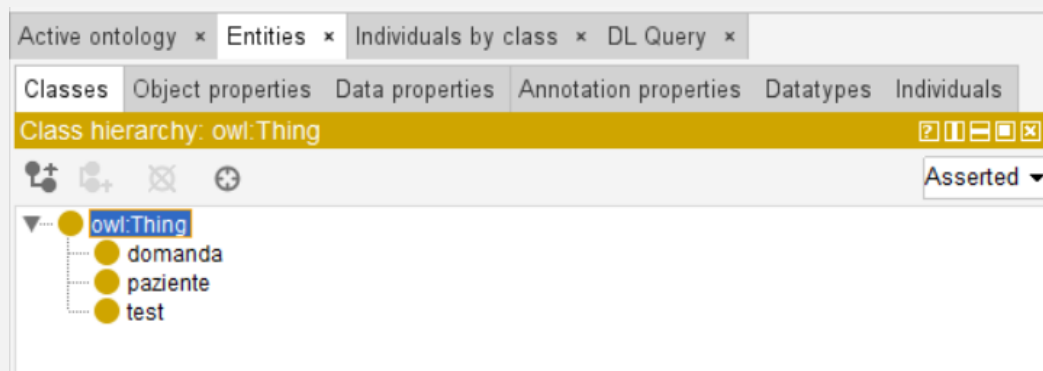
- Age: rappresenta l'età del paziente al momento del test.
- Gender: rappresenta il genere del paziente.
- isAutistic: vera se paziente è stata già riscontrata una forma di autismo.
- Ittero: vera se il paziente è nato con una forma di ittero.
- PDDparent: vera se una parente stretto del paziente presenta forme di PDD.
- Etnia: rappresenta l'etnia del paziente.
- IdPaziente: rappresenta il paziente in modo univoco.

• **Test:** La classe Test, rappresenta l'insieme dei test svolti da tutti i pazienti. A questa classe sono associati tre proprietà:

- **CompilatoreTest:** rappresenta chi ha compilato svolto il test con il paziente
- **IdTest:** identifica univocamente un test

• **Domanda:** rappresenta la classe delle domande svolte. Abbiamo associato a questa classe due proprietà:

- **Risposta:** vera se è vera la risposta del test.
- **NumQuestion:** numero della domanda all'interno del test.

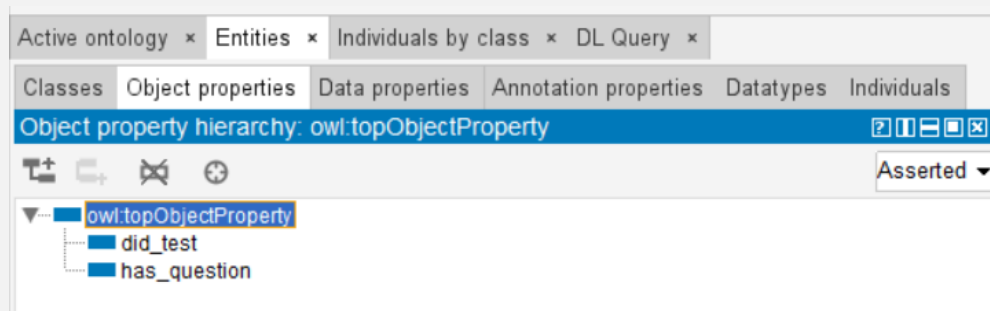


2.1.2 Object property:

Una object property permette di mettere in relazione due individui, siano essi di classi distinte o della stessa classe.

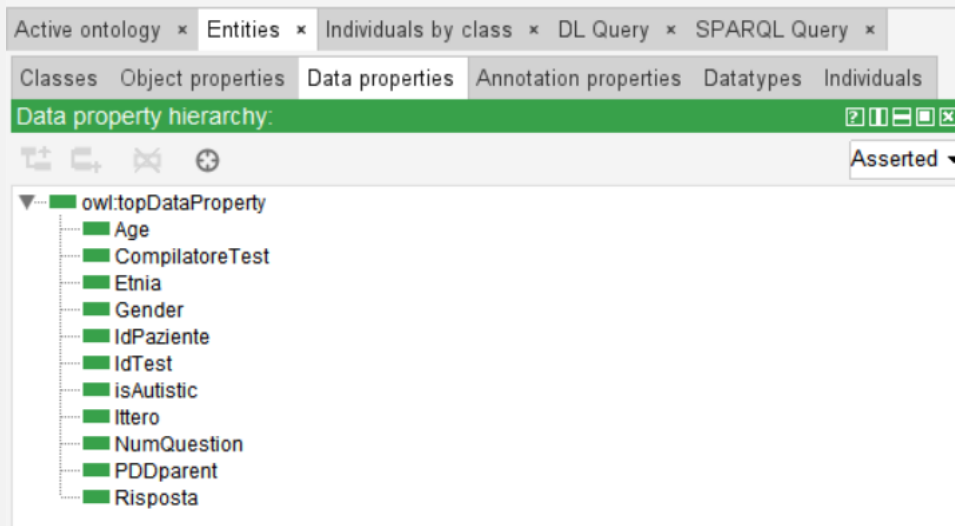
Tra le classi abbiamo definito delle relazioni che rappresentano come queste interagiscono tra di loro. Le relazioni definite sono:

- **did_Test(Paziente) -> Test** : permette di individuare i test svolti da un paziente.
- **has_Question(Test) -> Domanda**: permette di individuare le risposte date dal paziente. Informazioni come 'Screening Score' e 'Used the screening app before' possono essere derivate dall'ontologia realizzata con semplici query in Sparql.



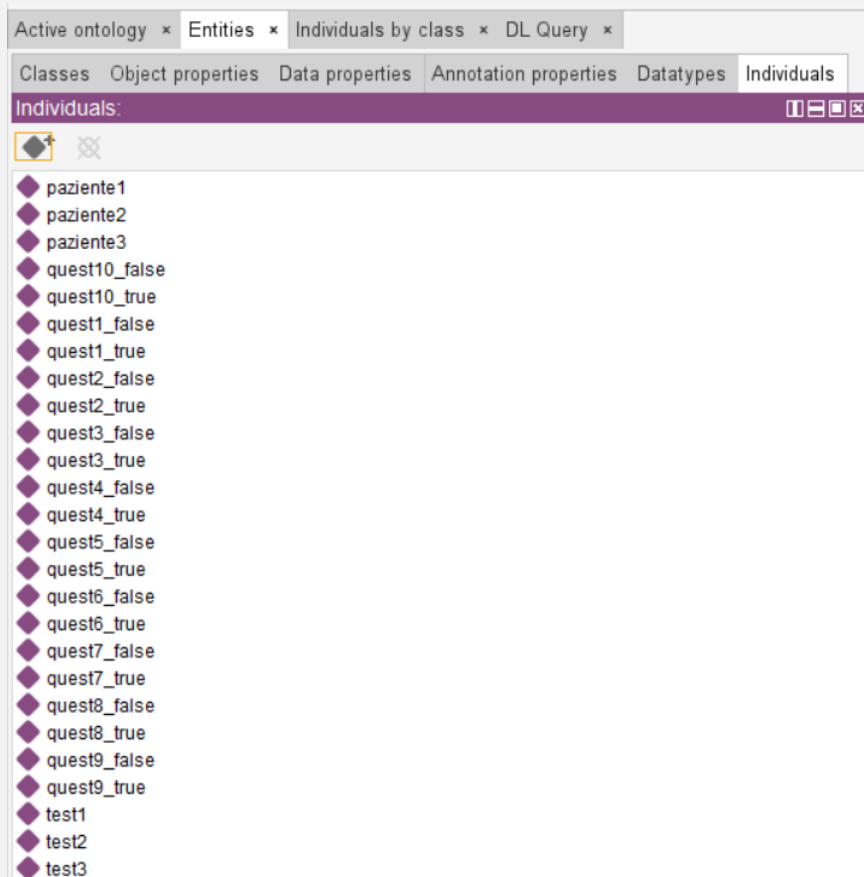
2.1.3 Data property:

Una data property permette di mettere in relazione un individuo con un valore di tipo primitivo.



2.1.4 Individuals:

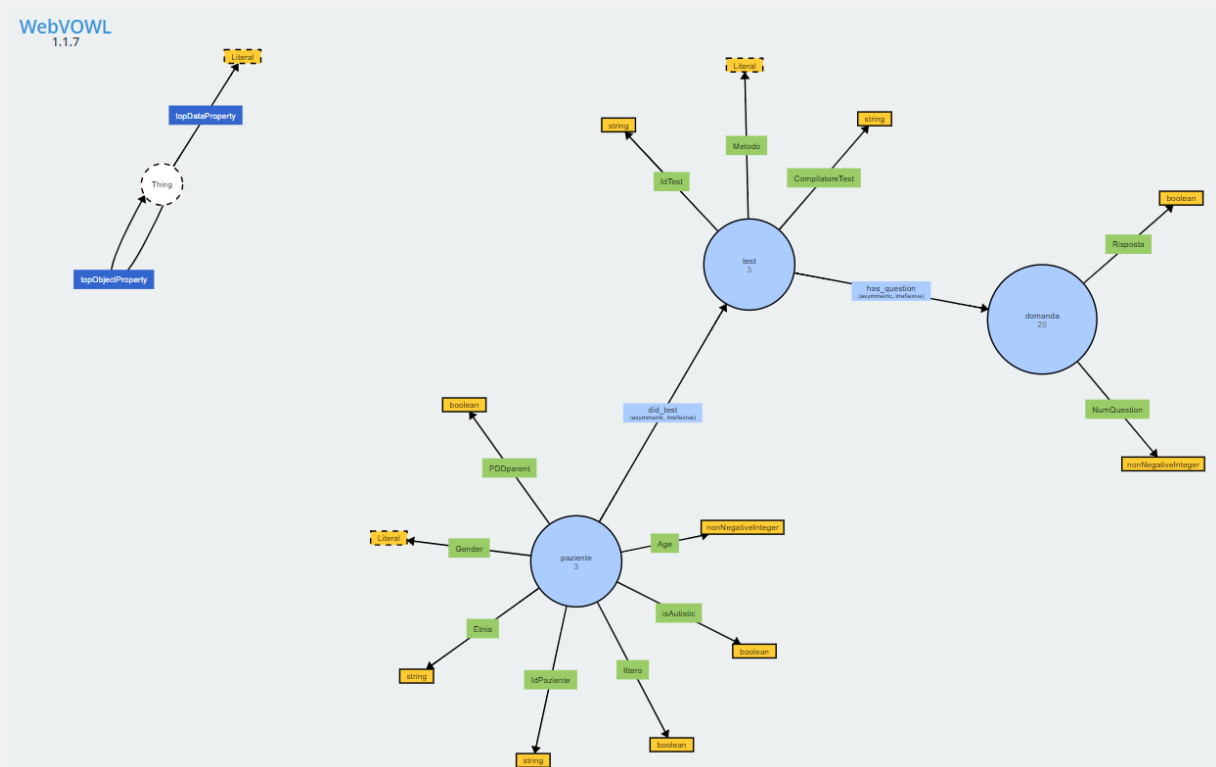
Per alcune entità si sono create delle istanze, ad esempio per individuare istanze di pazienti a cui attribuire un test e le relazioni con esso. Individui inseriti nella nostra ontologia:



2.2 Software per la realizzazione dell'Ontologia:

L'ontologia è stata creata mediante il software Protege, che è uno strumento per creare e gestire ontologie e che permette di definire concetti, classi e relazioni in ontologie basate su standard come OWL.

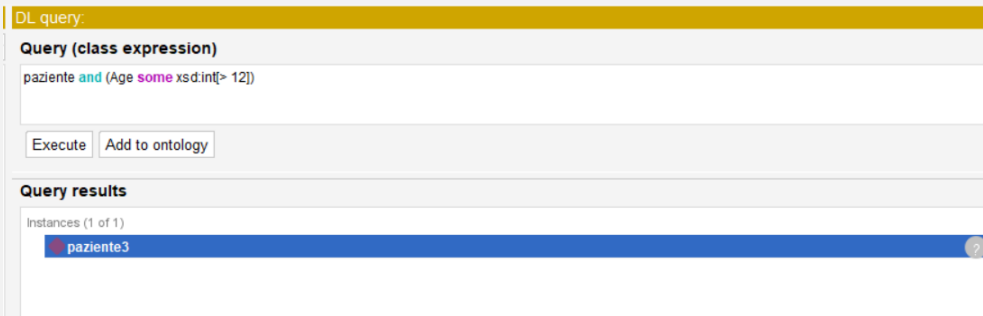
Per la rappresentazione grafica dell'ontologia è stato usato WebVOWL 1.1.7 (Visual Notation for OWL Ontologies) che è uno strumento online che consente di visualizzare e interagire con ontologie OWL in modo grafico e intuitivo.



3. QUERY:

Successivamente sono state formulate delle query per interrogare l'ontologia.

3.1 DL Query:



The screenshot shows a web interface for a DL query. At the top, there's a yellow header with the text "DL query:". Below it, a section titled "Query (class expression)" contains the query: "paziente and (Age some xsd:int[> 12])". There are two buttons: "Execute" and "Add to ontology". Below this, a section titled "Query results" shows "Instances (1 of 1)" and a single instance named "paziente3" in a blue bar.

Una query che restituisce tutti i pazienti con età maggiore di 12 anni, scritta in linguaggio DL (Descriptions logics), che è un linguaggio formale utilizzato principalmente per la modellazione concettuale e l'ontologia nelle discipline dell'intelligenza artificiale e della rappresentazione della conoscenza.

3.2 OwlReady:

È stato poi creato il file "ontologia.py" con il quale è possibile consultare l'ontologia direttamente in Python grazie alla libreria OwlReady2 per la manipolazione di ontologie e il ragionamento.

- Con il seguente codice è quindi possibile estrarre dall'ontologia la lista delle classi, delle object property, dei data property e degli individui che appartengono alle relative classi.

```
1  # -*- coding: utf-8 -*-
2  from owlready2 import *
3
4  print("ONTOLOGIA\n")
5  onto = get_ontology("ontologia.owl").load()
6
7  #stampa il contenuto principale dell'ontologia
8  print("-----Class list in ontology:-----\n")
9  print(list(onto.classes()), "\n")
10
11 #stampa le proprietà dell'oggetto
12 print("-----Object property in ontology:-----\n")
13 print(list(onto.object_properties()), "\n")
14
15 #stampa le proprietà dei dati
16 print("-----Data property in ontology:-----\n")
17 print(list(onto.data_properties()), "\n")
18
19 #stampa gli individui della classe paziente
20 print("-----paziente list in ontology:-----\n")
21 paziente = onto.search(is_a = onto.paziente)
22 print(paziente, "\n")
23
24 #stampa gli individui della classe test
25 print("-----test list in ontology:-----\n")
26 test = onto.search(is_a = onto.test)
27 print(test, "\n")
28
29 #stampa gli individui della classe domanda
30 print("-----domanda list in ontology:-----\n")
31 domanda = onto.search(is_a = onto.domanda)
32 print(domanda, "\n")
33
```

Il codice precedente produce i seguenti risultati:

```
-----Class list in ontology:-----
[ontologia.paziente, ontologia.test, ontologia.domanda]

-----Object property in ontology:-----
[ontologia.did_test, ontologia.has_question]

-----Data property in ontology:-----
[ontologia.Age, ontologia.CompilatoreTest, ontologia.Etnia, ontologia.Gender, ontologia.IdPaziente,
ontologia.IdTest, ontologia.Ittero, ontologia.NumQuestion, ontologia.PDDparent, ontologia.Risposta,
ontologia.isAutistic]

-----paziente list in ontology:-----
[ontologia.paziente, ontologia.paziente1, ontologia.paziente2, ontologia.paziente3]

-----test list in ontology:-----
[ontologia.test, ontologia.test1, ontologia.test2, ontologia.test3]

-----domanda list in ontology:-----
[ontologia.domanda, ontologia.quest10_false, ontologia.quest10_true, ontologia.quest1_false,
ontologia.quest1_true, ontologia.quest2_false, ontologia.quest2_true, ontologia.quest3_false,
ontologia.quest3_true, ontologia.quest4_false, ontologia.quest4_true, ontologia.quest5_false,
ontologia.quest5_true, ontologia.quest6_false, ontologia.quest6_true, ontologia.quest7_false,
ontologia.quest7_true, ontologia.quest8_false, ontologia.quest8_true, ontologia.quest9_false,
ontologia.quest9_true]
```

- Attraverso il seguente codice è stato invece possibile interrogare l'ontologia.

```
34 # QUERY-----
35 print("                QUERY\n")
36 # Esegui la query per ottenere i pazienti che hanno effettuato il test
37 pazientiConTest = onto.search(is_a = onto.paziente, haTest = onto.Test)
38
39 print("- Lista dei pazienti che hanno effettuato un test:\n")
40 for paziente in pazientiConTest:
41     print(paziente)
42
43 # Query per estrarre i pazienti con isAutistic impostato a True
44 query_result = list(onto.search(type=onto.paziente, isAutistic=True))
45
46 # Stampa i risultati
47 print("\n\n- Pazienti che hanno effettuato un test e sono autistici:\n")
48 for paziente in query_result:
49     print(paziente.name)
```

Il codice precedente restituisce come risultato la lista dei pazienti che hanno effettuato un test, come mostrato nella prima parte dell'esempio, e la lista dei pazienti autistici che hanno effettuato il test, come mostrato nella seconda parte dell'esempio.

```
                QUERY

- Lista dei pazienti che hanno effettuato un test:

ontologia.paziente
ontologia.paziente1
ontologia.paziente2
ontologia.paziente3

- Pazienti che hanno effettuato un test e sono autistici:

paziente2
paziente3
```

4. APPRENDIMENTO SUPERVISIONATO:

L'apprendimento supervisionato è una tecnica di apprendimento automatico che mira a istruire un sistema informatico in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input sulla base di una serie di esempi ideali, costituiti da coppie di input e di output, che gli vengono inizialmente forniti.

4.1 Decisioni di progetto:

In questo progetto, l'obiettivo dell'apprendimento supervisionato è risolvere un problema di classificazione, in particolare utilizzando la colonna 'Class/ASD' come variabile target da predire basandosi sulle altre caratteristiche relative ai dati dei pazienti.

Inizialmente è stato necessario decidere quali modelli utilizzare. Sono stati scelti gli algoritmi con le migliori prestazioni, in quanto la sensibilità del tema necessitava una grande precisione nelle valutazioni. Per questo motivo è stato scelto ad esempio random forest piuttosto che decision tree, in quanto è stata preferita la precisione del random forest alla minore complessità del decision tree, che potrebbe garantire prestazioni inferiori nel caso, ad esempio, in cui l'albero diventa più profondo e complesso.

Sono stati scelti 4 altri modelli (implementati con le librerie '*sklearn*' e '*keras*') e ne è stata valutata l'efficacia e l'efficienza:

- **K-nearest-neighbors (KNN)**
- **Random Forest**
- **Support Vector Machine (SVM)**
- **Neural Network**

4.2 Metriche di valutazione:

Andiamo adesso ad analizzare le metriche con cui sono stati valutati i modelli ed i risultati. Per ogni algoritmo implementato sono stati prodotti 4 tipologie di grafici differenti:

- **ROC Curve (Receiver Operating Characteristic Curve):** La curva ROC è un grafico in cui l'asse delle ordinate rappresenta il tasso di veri positivi (True Positive Rate), mentre l'asse delle ascisse rappresenta il tasso di falsi positivi (False Positive Rate).
- **Precision-Recall Curve:** Questa curva rappresenta il trade-off tra la precisione e il recall di un modello di classificazione binaria.
- **Bar Chart di Varianza e Deviazione Standard:** Questo tipo di grafico a barre rappresenta la varianza e la deviazione standard dei punteggi di cross-validation. La varianza e la deviazione standard aiutano a comprendere quanto i risultati siano consistenti o variabili su diverse iterazioni della cross-validation.

- **Matrice di Confusione:** La matrice di confusione è una tabella che mostra il numero di previsioni corrette e errate fatte da un modello di classificazione in termini di veri positivi (TP), falsi positivi (FP), veri negativi (TN) e falsi negativi (FN). .

La **cross-validation** è un'altra tecnica utilizzata per valutare le prestazioni dei modelli in modo accurato e affidabile. Abbiamo scelto di dividere i dati in 5 "fold" uguali, in modo da addestrare e testare il modello 5 volte, ognuna delle quali usa una fold diversa come set di test e l'unione delle rimanenti come set di addestramento.

4.3 Selezione delle Feature:

Nella selezione delle feature abbiamo scelto di selezionare:

- Le risposte al questionario (*'A1_Score', 'A2_Score', ..., 'A10_Score'*) sono le nostre feature principali, in quanto rappresentano il metro di giudizio più evidente.
- Età (*'age'*) e Genere (*'gender'*) sono feature che potrebbero essere rilevanti, poiché l'età e il genere possono influenzare la prevalenza e la manifestazione dell'autismo.
- Il punteggio di Screening (*'screening_score'*), al pari delle risposte al questionario, è una delle feature principali in quanto rappresenta la somma delle risposte al questionario.
- *'PDD_parent'* può essere un'altra feature importante per comprendere la trasmissibilità dello spettro autistico, quindi come un genitore affetto da autismo possa influenzare la possibilità che questa diagnosi si ripercuota sul proprio figlio.

'Ethnicity', 'contry_of_res', 'used_app_before' e *'test_compiler'* sono i campi del nostro dataset che non sono stati inseriti tra le nostre feature in quanto nonostante potessero avere un impatto sulla diagnosi, non le abbiamo ritenute rilevanti per l'addestramento del modello a tal punto da considerarle feature.

4.4 Preprocessing dei Dati:

L'addestramento di ogni modello inizia con l'esecuzione del preprocessing dei dati.

4.4.1 : Dummification

Questa operazione include la normalizzazione delle feature attraverso la funzione *'get_dummies()'* per eseguire la codifica one-hot delle variabili categoriche in quanto questi modelli non supportano questo tipo di variabili.

La codifica one-hot è un processo che converte le variabili categoriche in una rappresentazione binaria, in modo che ciascuna categoria venga rappresentata come una colonna binaria separata

```
data_dummies = pd.get_dummies(dataset, columns=feature_dummied)
data_dummies = data_dummies.drop(["Class/ASD"], axis=1)
```

Dato che il target della nostra predizione è Class/ASD, la colonna relativa a questo campo viene eliminata dal dataframe con il metodo `'drop()'`.

La colonna Class/ASD viene inserita invece in una nuova variabile.

```
y = pd.get_dummies(dataset["Class/ASD"], columns=["Class/ASD"])
y = y["1"]
```

4.4.2: SMOTE

È stata utilizzata la funzione «SMOTE», implementata grazie all'uso della libreria imblearn, con lo scopo di ridimensionare la classe di esempi. SMOTE è utile quando si lavora con dataset sbilanciati, dove una classe è rappresentata in modo significativamente minore rispetto all'altra classe.

```
oversample = SMOTE()
X1, y1 = oversample.fit_resample(X, y)
```

Applichiamo quindi la tecnica SMOTE ai parametri ``X`` e ``y`` che rappresentano le feature e il target. X1 conterrà le nuove feature sintetiche create da SMOTE per bilanciare le classi e y1 conterrà le corrispondenti etichette target.

4.5 Divisione dei Dati:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, random_state = 13)
```

Nel contesto del problema specifico, abbiamo scelto di suddividere il nostro dataset in 75% per l'addestramento, e il restante 25% per il test. Questa suddivisione è sembrata un compromesso ragionevole tra la necessità di avere abbastanza dati per l'addestramento e la necessità di avere abbastanza dati nel set di test per valutare accuratamente le prestazioni del modello.

Se avessimo scelto una proporzione più alta per il set di test, avremmo avuto meno dati disponibili per l'addestramento del modello, ciò avrebbe potuto comportare una minore capacità del modello di apprendere le complessità dei dati e avrebbe potuto portare a una generalizzazione meno accurata.

D'altro canto, se avessimo scelto una proporzione più alta per il set di addestramento, saremmo potuti incorrere in un rischio di overfitting, acquisendo anche il rumore nei dati e risultando meno in grado di generalizzare su nuovi dati.

4.6 K-Nearest Neighbors (KNN):

Il KNN è un algoritmo di classificazione che mira a determinare la classe di appartenenza di un dato di input cercando tra tutti gli esempi di addestramento quello più vicino al dato di input in base alla metrica desiderata, come ad esempio la distanza euclidea.

4.6.1 Decisioni di progetto:

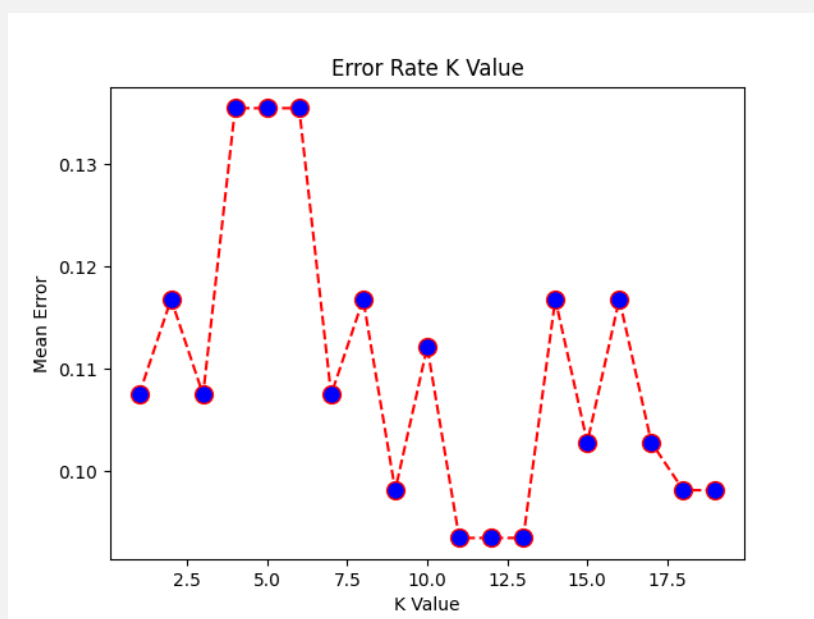
Questo modello è stato scelto perché semplice da implementare, versatile, in quanto funziona bene con dataset di qualsiasi dimensione, e robusto anche in presenza di dati rumorosi o valori anomali. Dato che il modello si basa sui vicini più prossimi, un singolo valore anomalo avrà meno impatto sull'output finale.

4.6.2 Ottimizzazione numero di vicini:

L'obiettivo di questo codice è quello di valutare come l'errore varia al variare del numero di vicini utilizzati in un range di valutazione da 1 a 20. Si sta cercando il valore ottimale di numero di vicini che minimizza l'errore di previsione e migliora la capacità di generalizzazione del modello.

```
for i in range(1, 20):  
    knn = KNeighborsClassifier(n_neighbors=i)  
    knn.fit(X_train, y_train)  
    pred_i = knn.predict(X_test)  
    error.append(np.mean(pred_i != y_test))
```

L'obiettivo è scegliere il valore di k che equilibra la complessità del modello con la sua capacità di generalizzazione.



In questo caso l'errore è più basso quando k è tra 11 e 13 perciò un k intorno a 11 potrebbe essere una scelta ragionevole per ottenere un buon equilibrio tra overfitting e underfitting, dato che un k troppo piccolo potrebbe causare overfitting, mentre un k troppo grande potrebbe causare underfitting.

4.6.3 Addestramento e predizione:

Scelto il numero di vicini, inizia l'addestramento del modello. Durante questo processo, il modello regola i suoi parametri in modo che le previsioni siano il più vicine possibile ai valori reali.

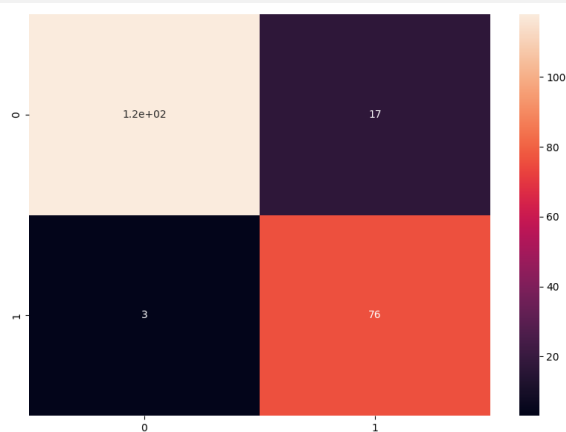
```
neigh = KNeighborsClassifier(n_neighbors = 11)
knn = neigh.fit(X_train, y_train)
```

Il modello viene addestrato cercando di trovare la relazione tra le feature e le etichette di classe. Il codice crea quindi un modello di classificazione addestrato pronto per essere utilizzato per fare previsioni su nuovi dati, cercando di assegnare loro l'etichetta di classe appropriata in base alle sue vicinanze rispetto ai dati di addestramento.

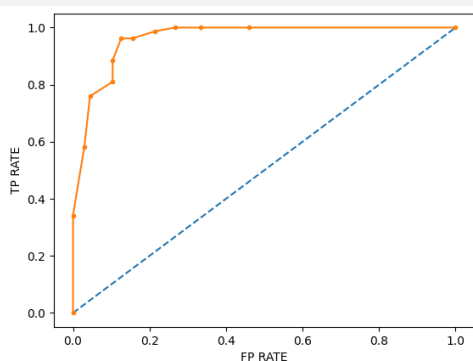
```
prediction = knn.predict(X_test)
accuracy = accuracy_score(prediction, y_test)
```

Il classificatore k-NN addestrato viene quindi usato per effettuare previsioni sul set di dati di test. Il metodo `predict` restituisce un array contenente i valori delle previsioni, che vengono confrontate con le etichette di classe effettive per calcolare il valore dell'accuracy,

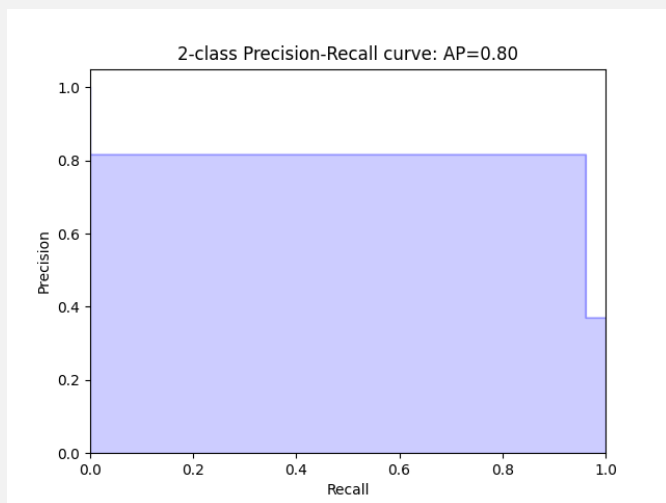
4.6.4 Valutazione Finale:



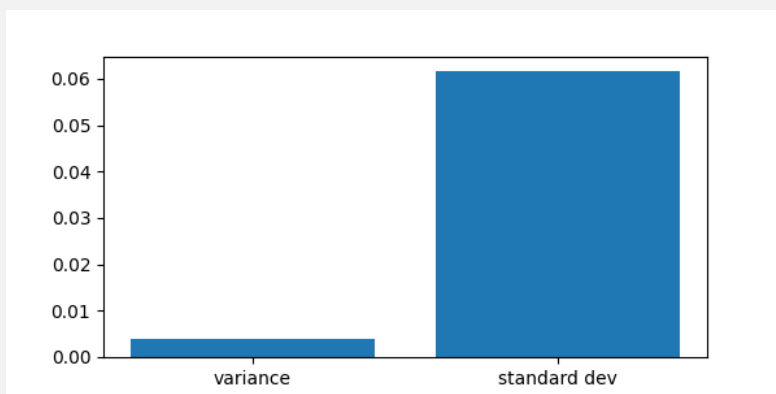
Matrice di Confusione: rivela che il modello ha una buona capacità di fare previsioni corrette per entrambe le classi. Si nota che i falsi positivi (17) sono leggermente più elevati dei falsi negativi (3), ciò potrebbe suggerire che il modello è leggermente incline a effettuare previsioni corrette quando la classe reale è negativa.



ROC Curve: L'Area Under the Curve (AUC) nella ROC Curve è estremamente positiva, con un valore di 0.962. Questo indica che il modello KNN ha un'eccellente capacità di discriminare tra le classi positive e negative.



Precision-Recall Curve: Il grafico mostra delle imperfezioni evidenti dall'analisi delle metriche di precision, recall. La precisione relativamente bassa potrebbe significare che il modello sta etichettando erroneamente alcuni campioni.



Bar Chart di Varianza e Deviazione Standard:

La bassa varianza nelle iterazioni di cross-validation indica che il modello è stabile e coerente nelle sue prestazioni. Inoltre, la deviazione standard suggerisce che il modello non è soggetto a fluttuazioni significative nelle performance durante le diverse iterazioni. Questo è un indicatore positivo della coerenza del modello e della sua capacità di mantenere risultati affidabili.

```

Clasification report:
      precision    recall  f1-score   support

   False      0.98      0.87      0.92       135
    True      0.82      0.96      0.88        79

 accuracy      0.91       214
 macro avg      0.90      0.92      0.90       214
weighted avg      0.92      0.91      0.91       214

Confusion matrix:
[[118  17]
 [   3  76]]

cv_scores mean:0.8841073271413828

cv_score variance:0.003802600249992634

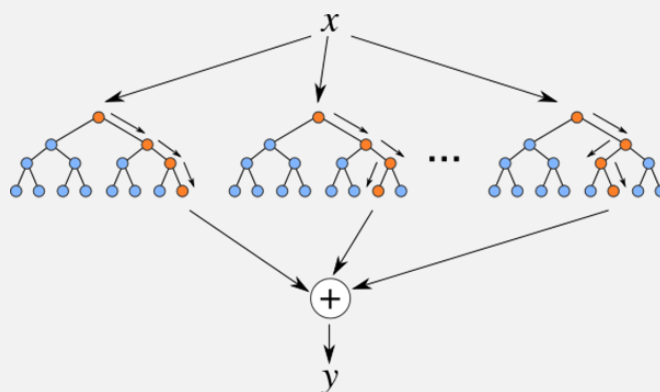
cv_score dev standard:0.06166522723539283

AUC: 0.962

```

4.7 Random Forest:

L'algoritmo Random Forest rappresenta un tipo di modello che combina molti alberi decisionali in un unico modello. Individualmente, le previsioni fatte dagli alberi decisionali potrebbero non essere accurate, ma combinate insieme, le previsioni saranno in media più vicine al risultato.



4.7.1 Decisioni di progetto:

L'algoritmo Random Forest è stato scelto poichè tende ad avere una buona precisione e stabilità nei problemi di classificazione. Inoltre, la sua capacità di addestrare gli alberi in modo parallelizzato consente una rapida esecuzione anche con risorse computazionali limitate.

- **'n_estimators=20'**: si è scelto di impostare a 20 questo parametro, che indica il numero di alberi decisionali che vengono creati nell'ensemble, in modo da trovare un equilibrio tra prestazioni e efficienza computazionale, dato che un numero maggiore di alberi può portare a prestazioni migliori, ma aumenta anche il tempo di addestramento.
- **'max_depth=30'**: questo parametro indica la profondità massima degli alberi decisionali all'interno dell'ensemble. È stato scelto 30 in quanto è un valore che permette agli alberi di adattarsi ai dati diminuendo il rischio di overfitting che si avrebbe avuto con una maggiore profondità.
- **'random_state=0'**: questo parametro controlla la generazione dei numeri casuali all'interno del modello. Fornendo un valore specifico ci si assicura che l'addestramento e la previsione del modello siano riproducibili.

```
clf = RandomForestClassifier(max_depth=30, random_state=0, n_estimators=20)
clf1 = clf.fit(x1_train, y1_train)
```

4.7.2 Ottimizzazione scelta numero alberi decisionali:

L'esecuzione del modello con i parametri impostati come in precedenza ha portato ad ottenere i seguenti risultati:

```
accuracy_score: 0.9066147859922179

Clasification report:
      precision    recall  f1-score   support

 False      0.91      0.91      0.91      138
  True      0.90      0.90      0.90      119

 accuracy      0.91      0.91      0.91      257
 macro avg      0.91      0.91      0.91      257
weighted avg      0.91      0.91      0.91      257

Confussion matrix:
[[126  12]
 [ 12 107]]

cv_scores mean:0.9407719630594364

cv_score variance:0.004197807289770974

cv_score dev standard:0.06479048764881287

AUC: 0.966
```

Abbiamo poi capito che scegliendo i valori più adatti per le variabili *'max_depth'* e *'random_state'*, i risultati, molto validi già di per sé, potevano essere migliorati ulteriormente. Per questo abbiamo deciso di usare un algoritmo che ci aiutasse proprio a scegliere questi valori.

```
# Definire un intervallo di valori da testare per max_depth
max_depth_values = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Definire un intervallo di valori per verificare lo random_state
random_state_values = [0, 4, 16, 64, 256, 1024, 4096]

# Memorizza i punteggi medi di cross-validation per ogni combinazione di max_depth e random_state
scores = []
for max_depth in max_depth_values:
    for random_state in random_state_values:
        RFC = RandomForestClassifier(max_depth=max_depth, random_state=random_state)
        cv_scores = cross_val_score(RFC, X, y, cv=5)
        scores.append((max_depth, random_state, cv_scores.mean()))
```

L'algoritmo parte definendo un intervallo di valori da testare per *max_depth* e *random_state*. Viene quindi eseguita una valutazione di cross-validation su ogni combinazione di valori di *'max_depth'* e *'random_state'* e viene calcolata la media delle valutazioni ottenute, scegliendo la combinazione che restituisce i risultati migliori.

```
Valore migliore max_depth: 6.0
Valore migliore random_state: 4096.0
```

Infine, viene addestrato un classificatore random forest, utilizzando i valori di *'max_depth'* e *'random_state'* appena ottenuti, e viene eseguita una valutazione di cross-validation sul nuovo classificatore.

L'immagine a sinistra riporta i valori precedenti all'uso dell'algoritmo, mentre quella a destra mostra i valori ottimizzati.

```
accuracy_score: 0.9066147859922179

Clasification report:
      precision    recall  f1-score   support

   False      0.91      0.91      0.91      138
    True      0.90      0.90      0.90      119

 accuracy      0.91      0.91      0.91      257
  macro avg      0.91      0.91      0.91      257
 weighted avg      0.91      0.91      0.91      257

Confussion matrix:
[[126  12]
 [ 12 107]]

cv_scores mean:0.9407719630594364

cv_score variance:0.004197807289770974

cv_score dev standard:0.06479048764881287
```

```
accuracy_score: 0.9377431906614786

Clasification report:
      precision    recall  f1-score   support

   False      0.97      0.91      0.94      138
    True      0.91      0.97      0.93      119

 accuracy      0.94      0.94      0.94      257
  macro avg      0.94      0.94      0.94      257
 weighted avg      0.94      0.94      0.94      257

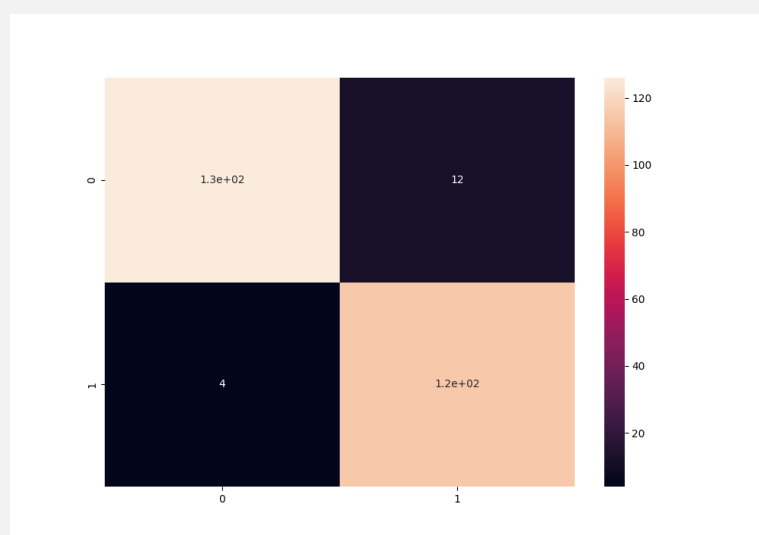
Confussion matrix:
[[126  12]
 [  4 115]]

cv_scores mean:0.928117452048307

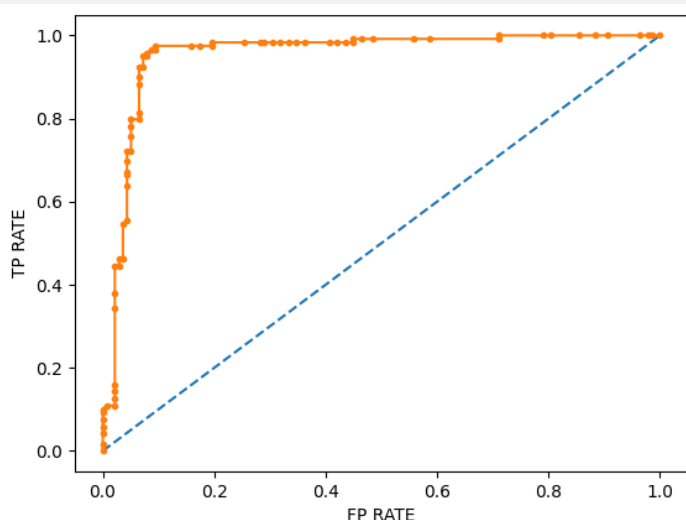
cv_score variance:0.005589393581118116

cv_score dev standard:0.07476224703095885
```

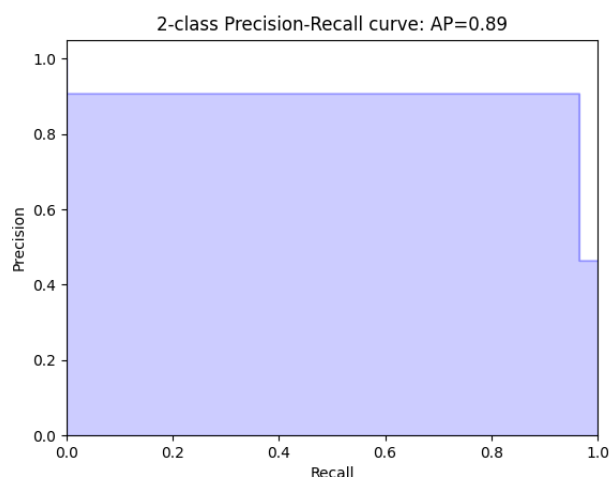
4.7.3 Valutazione Finale:



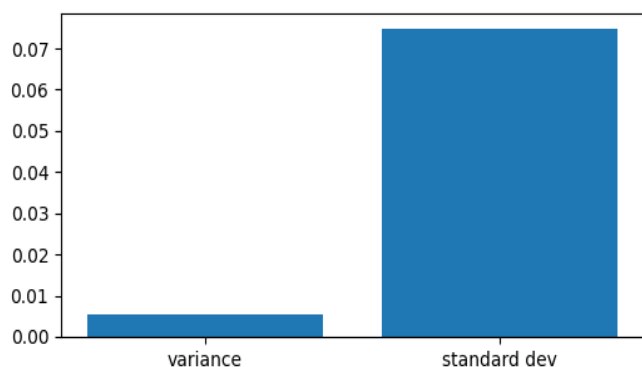
Matrice di confusione: mostra che ci sono pochi falsi positivi, solo 12 a fronte di 126 classificazione corrette, e ancor meno falsi negativi, solo 4 su un totale di 119 previsioni, il che indica un'eccellente capacità del modello nel limitare gli errori di classificazione.



ROC curve: L'Area Under the Curve (AUC) indica una forte capacità del modello di discriminare tra le classi. Questo punteggio suggerisce che il modello ha una buona capacità di separazione delle classi e un'ottima performance complessiva.



Precision-recall curve: l'analisi della precision-recall curve presenta dei valori molto bilanciati. Questi risultati indicano che il modello è abile a prevedere le classificazioni, suggerendo che la quasi totalità delle previsioni sia stata corretta.



Bar Chart di Varianza e Deviazione Standard: l'analisi delle metriche di validazione incrociata (cv_scores mean, cv_score variance, cv_score dev standard) suggerisce che il modello mantiene un livello di coerenza nelle prestazioni quando viene applicato a diverse parti del dataset. La bassa varianza e la deviazione standard delle cv_scores indicano una certa stabilità nelle prestazioni.

4.8 Support Vector Machines (SVM):

L'SVM è basato sull'idea di trovare un iperpiano che divida al meglio un set di dati in due classi, i Support Vector sono i punti dati più vicini all'iperpiano e tali punti dipendono dal set di dati che si sta analizzando.

4.8.1 Decisioni di progetto:

Questo modello è stato scelto in quanto potente per la classificazione binaria, in particolare quando le classi sono ben separate. È adatto per problemi con un numero moderato di feature e può gestire relazioni non-lineari attraverso l'uso di kernel.

- Abbiamo scelto di usare il **kernel RBF** (Radial Basis Function) per questo modello perché con il kernel RBF è possibile comprendere strutture complesse e non lineari che potrebbero essere presenti nei dati, migliorando le probabilità di ottenere un modello di classificazione più accurato e generalizzabile.

4.8.2 Ottimizzazione parametro gamma:

L'iperparametro gamma nel kernel RBF controlla quanto l'influenza di un singolo esempio di addestramento si estenda. Valori più alti di gamma rendono le decisioni più locali, mentre valori più bassi danno una maggiore influenza alle istanze circostanti. L'obiettivo del seguente codice è quello di trovare il miglior valore per il parametro gamma.

```
# Definisco i valori di gamma da testare
param_grid = {'gamma': [1e-4, 1e-3, 0.01, 0.1, 1, 10, 100]}

# Creazione del modello SVC
svm_model = svm.SVC()

# Ricerca a griglia con cross-validation
grid_search = GridSearchCV(svm_model, param_grid, cv=5)
grid_search.fit(X1_train, y1_train)
```

Per fare ciò, si utilizza la tecnica della "grid search" insieme alla cross-validation. Si inizia definendo una griglia di valori di gamma, e successivamente, dopo aver creato un modello SVM, si esegue la ricerca a griglia che addestrerà e valuterà il modello su diverse combinazioni di parametri gamma utilizzando la cross-validation con 5 fold.

Infine, con il metodo fit, si otterrà il modello SVM ottimizzato con i migliori parametri gamma individuati dalla grid search. Il risultato del codice è il seguente:

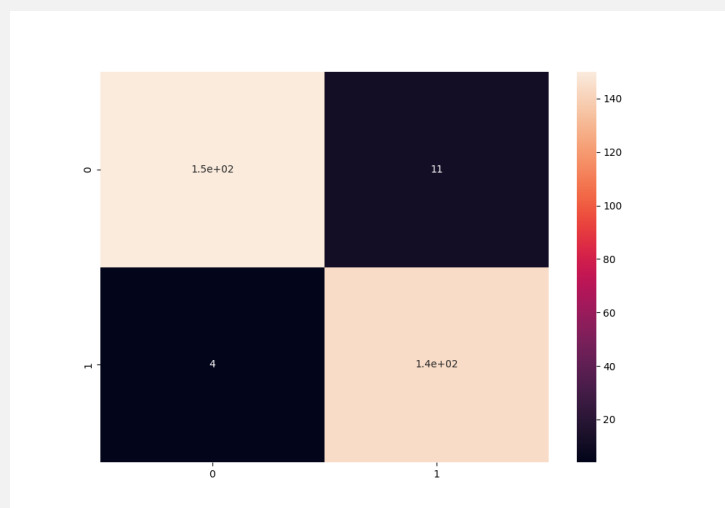
```
Gamma: 0.0001    Mean Score: 0.5090423465423466
Gamma: 0.001     Mean Score: 0.8943181818181817
Gamma: 0.01      Mean Score: 0.9582944832944833
Gamma: 0.1       Mean Score: 0.9569055944055943
Gamma: 1         Mean Score: 0.7482517482517482
Gamma: 10        Mean Score: 0.7440753690753691
Gamma: 100       Mean Score: 0.7440753690753691

Miglior valore di gamma: 0.01
```

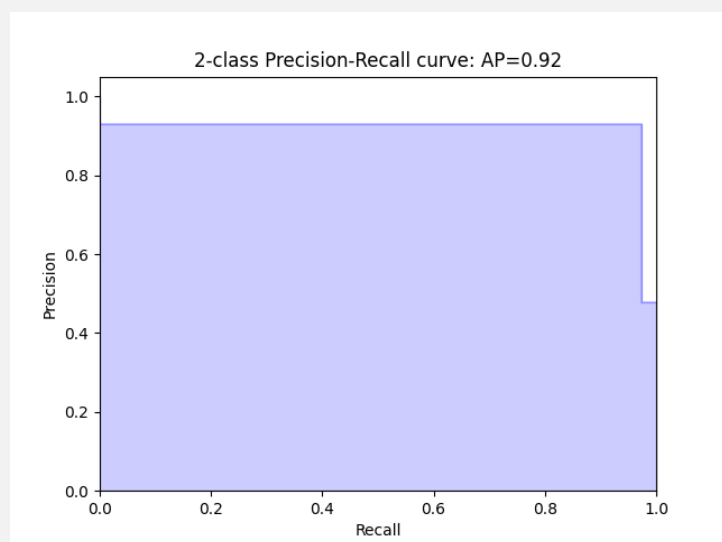
Questi risultati ci fanno capire come il modello risulta ottimizzato con il valore di gamma impostato a 0.01.

```
clf = svm.SVC(kernel='rbf', gamma=0.01)
clf.fit(X1_train, y1_train)
```

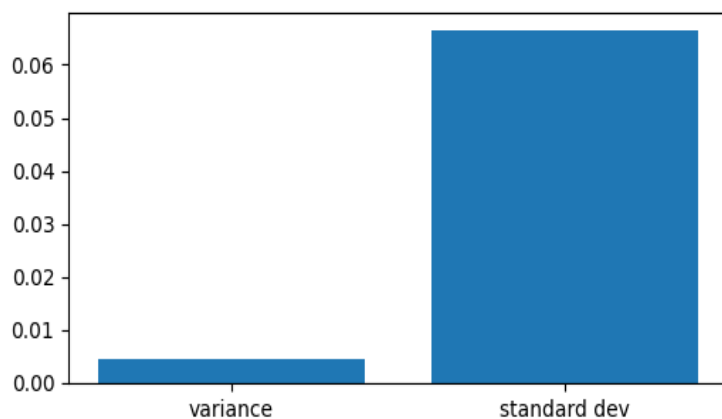
4.8.3 Valutazione Finale:



Matrice di Confusione: mostra come il modello abbia effettuato nel complesso delle ottime classificazioni, nonostante qualche errore, giustificabile in parte però dalla quantità di previsioni effettuate.



Precision-Recall Curve: il grafico mostra un ottimo bilanciamento tra precision e recall, confermando l'affidabilità delle previsioni.



Bar Chart di Varianza e Deviazione Standard: i risultati evidenziano una bassa varianza e una leggermente maggiore deviazione standard. Ciò suggerisce coerenza nelle prestazioni medie del modello, ma con alcune variazioni significative in alcuni fold.

```

Clasification report:
              precision    recall  f1-score   support

     False      0.98      0.93      0.96       161
     True       0.93      0.98      0.95       148

 accuracy              0.95       309
 macro avg      0.95      0.96      0.95       309
weighted avg      0.96      0.95      0.95       309


Confussion matrix:
[[150  11]
 [  3 145]]

cv_scores mean:0.9553303338858632

cv_score variance:0.004697144068971779

cv_score dev standard:0.06853571382112963

```


4.9 Neural Network:

Le reti neurali simulano neuroni artificiali collegati tra loro, elaborando stimoli attraverso pesi e soglie. L'attivazione avviene se il risultato supera la soglia, i pesi quantificano l'importanza degli input.

4.9.1 Decisioni di progetto:

Il modello è stato scelto per la sua capacità di affrontare problemi di apprendimento profondi e complessi e per la sua capacità di raggiungere prestazioni di livello superiore in termini di accuratezza e previsione.

Abbiamo scelto di definire una rete neurale sequenziale con due livelli. Il primo livello, con 30 neuroni nascosti, utilizza la funzione di attivazione ReLU, il secondo livello ha un singolo neurone con una funzione di attivazione sigmoide, che produce un valore compreso tra 0 e 1 per la classificazione binaria.

4.9.2 Creazione modello e addestramento:

Il seguente codice definisce e crea un modello di rete neurale sequenziale utilizzando il framework Keras.

```
def create_model():  
    network = Sequential()  
    network.add(Dense(30, input_dim=184, activation="relu"))  
    network.add(Dense(1, activation='sigmoid'))  
    network.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])  
    return network
```

La funzione 'create_model()' inizia definendo un oggetto di tipo Sequential, che è la base per costruire reti neurali sequenziali in Keras. Si aggiunge poi un livello denso (fully connected) con 30 unità nascoste, un numero moderato di unità nascoste è spesso utilizzato per evitare l'overfitting.

Si sceglie poi la funzione di attivazione '**ReLU**' per introdurre non-linearità. Il secondo livello denso ha un'unica unità di output e utilizza una funzione di attivazione '**sigmoide**' per comprimere l'output in un intervallo tra 0 e 1.

La funzione termina con la compilazione del modello, utilizzando la funzione '**binary_crossentropy**', che è appropriata per problemi di classificazione binaria e l'ottimizzatore '**adam**' che viene utilizzato per addestrare la rete, ed è noto per la sua efficacia in una varietà di scenari.

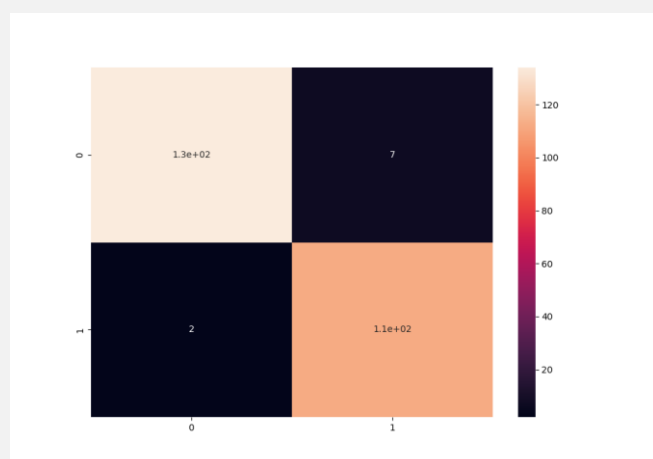
Si crea quindi un'istanza del modello e inizia l'addestramento sui dati.

```
model = create_model()  
model.fit(X_train, y_train, epochs=30, batch_size=64)
```

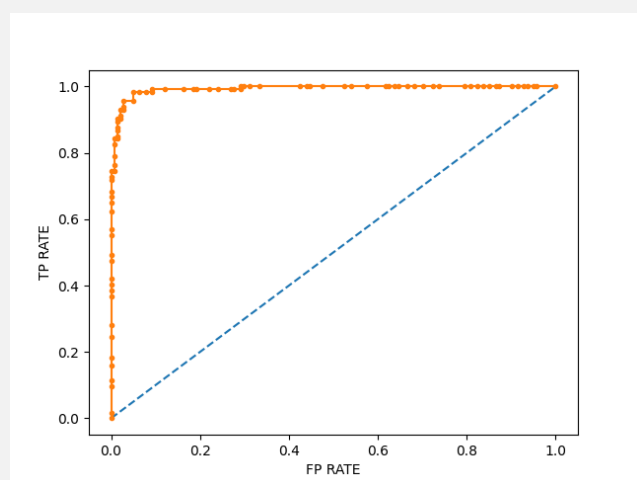
-**"epochs"** specifica quante volte l'intero dataset di addestramento verrà utilizzato per aggiornare i pesi del modello. Abbiamo scelto di addestrare il modello per "30 epochs", dato che maggiori epoche potrebbero consentire al modello di adattarsi meglio ai dati di addestramento, ma potrebbero anche portare all'overfitting.

-**"batch_size"** definisce la dimensione del batch, ovvero quante istanze di addestramento vengono utilizzate prima di aggiornare i pesi del modello. Abbiamo scelto di utilizzare un batch di dimensione "64", perchè rappresenta un compromesso tra l'accuratezza del modello e l'efficienza di addestramento. Potrebbe fornire un'adeguata quantità di dati per apprendere modelli complessi senza richiedere troppo tempo computazionale.

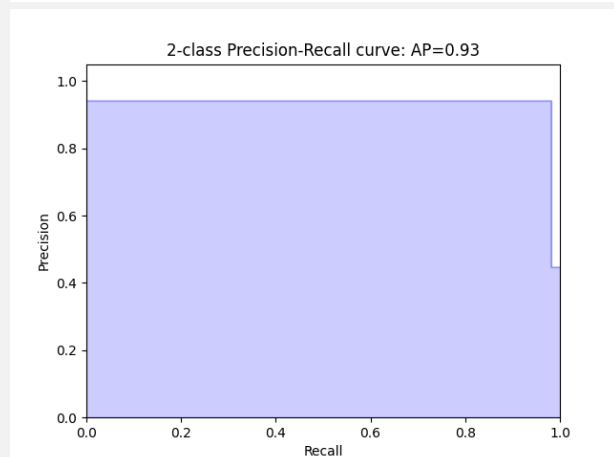
4.9.3 Valutazione Finale:



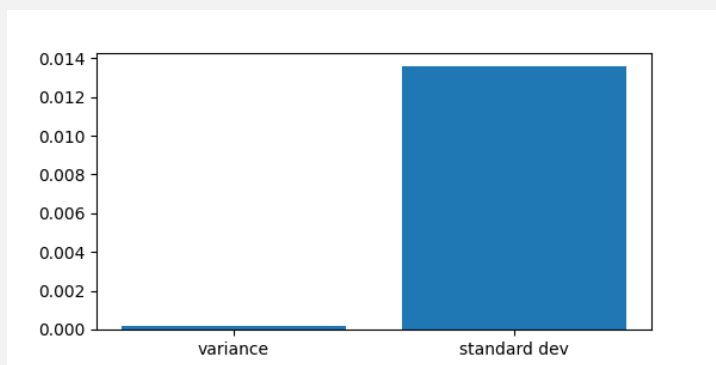
Matrice di Confusione: il grafico mostra che il modello ha previsto correttamente la maggior parte delle istanze sia per la classe "False" che per la classe "True", garantendo affidabilità nonostante qualche piccolo errore.



ROC curve: il grafico indica un'ottima capacità del modello nel discriminare tra le due classi, fornendo previsioni accurate e bilanciate tra true positive e false positive



Precision-Recall Curve: il grafico indica che il modello riesce a fare previsioni accurate e a individuare la maggior parte delle istanze positive reali. Questo suggerisce che il modello è efficace nel separare le classi e nel produrre risultati coerenti.



Bar Chart di Varianza e Deviazione Standard:

Questo grafico indica una ottima coerenza e consistenza nei risultati, che sottolinea grande stabilità e affidabilità nel comportamento durante il processo di addestramento e valutazione.

```

Clasification report:
              precision    recall  f1-score   support

     False      0.99      0.95      0.97      141
     True       0.94      0.98      0.96      114

 accuracy              0.96              0.96      255
 macro avg              0.96      0.97      0.96      255
 weighted avg           0.97      0.96      0.96      255

Confussion matrix:
[[134   7]
 [   2 112]]

cv_scores mean:0.9514877880976952

cv_score variance:0.00018467102349180477

cv_score dev standard:0.01358937171070851

AUC: 0.992

```

4.10 Conclusioni:

Come metriche per confrontare i nostri modelli abbiamo usato la media di accuratezza, varianza e deviazione standard generata attraverso il processo di cross-validation, la media pesata del f1-score, la precisione media e l'AUC.

MODELLO	ACCURATEZZA	VARIANZA	DEV.STANDAD	F1-SCORE	AVERAGE-PRECISION	AUC
KNN	0.884	0.0038	0.062	0.91	0.80	0.962
RANDOM FOREST	0.928	0.0056	0.075	0.94	0.89	0.954
SVM	0.955	0.0047	0.069	0.95	0.92	/
NEURAL NETWORK	0.951	0.0002	0.014	0.96	0.93	0.992

Analizzando le prestazioni generali dei vari algoritmi di apprendimento senza entrare nei dettagli di ogni singolo campo, possiamo osservare le seguenti tendenze:

- **Neural Network** presenta dei valori che indicano un alto livello di precisione e coerenza nei risultati. L'F1-Score, l'Average-Precision e l'AUC sono i più alti tra tutti i modelli considerati, dimostrando una notevole capacità predittiva e discriminativa del modello.
- **Support vector machines (SVM)** presenta un'accuratezza ancora più elevata dimostra un'elevata competenza nel classificare i dati. La bassa varianza sottolinea la capacità di generalizzare su diverse partizioni dei dati, ma la leggermente maggiore deviazione standard potrebbe indicare alcune variazioni nelle prestazioni tra i fold.
- **Random Forest** ha dimostrato generalmente prestazioni quasi eccellenti. L'accuratezza del modello, è un risultato notevole e suggerisce che il modello sta facendo un buon lavoro nella classificazione complessiva. L'algoritmo è coerente nel corso delle esecuzioni e non dimostra sostanziali disuguaglianze di prestazioni tra le classi.
- **K-Nearest Neighbor (KNN)** i risultati indicano che questo modello ha una buona capacità di discriminare tra le classi, garantendo una buona performance globale, ci sono però segnali di disuguaglianza di prestazioni tra le classi e una variazione nelle prestazioni quando viene applicato a diversi set di dati. La varianza e la deviazione standard confermano la mancanza di piena coerenza del modello nelle iterazioni di cross-validation.

In conclusione, i dati evidenziano un chiaro ordine di performance tra i modelli considerati. Il Neural Network si pone come il modello più affidabile e preciso, seguito da SVM e Random Forest. KNN si attesta a un livello inferiore di accuratezza e prestazioni complessive.

5 CLUSTERING:

Dopo aver esplorato i metodi di apprendimento supervisionato abbiamo ritenuto che l'applicazione del clustering potesse offrire un ulteriore approccio per comprendere meglio la struttura intrinseca dei dati.

Questa tecnica ci consente di identificare naturali raggruppamenti nei dati, anche senza etichette di classe, e ciò potrebbe rivelare relazioni sottostanti tra le risposte al questionario e fornirci una prospettiva diversa sulla distribuzione dei dati.

Il clustering può essere di due tipi:

- Clustering rigido (Hard clustering): prevede l'assegnazione statica di ciascun esempio a una classe di appartenenza.
- Clustering morbido (Soft clustering): utilizza distribuzioni di probabilità per assegnare le classi associate a ciascun esempio.

5.1 Decisioni di progetto:

Questa tecnica è stata scelta per raggruppare i lavoratori in categorie basate sulle caratteristiche presenti nel file 'Autism-Dataset.csv'. L'obiettivo è individuare dei cluster, ovvero gruppi di esempi simili a centroidi calcolati in modo automatico.

L'utilizzo di questa tecnica nel progetto mira a creare una nuova colonna da aggiungere alle features relative ai dati di ciascun lavoratore, in un nuovo file chiamato 'Autism-Dataset+clusters.csv'. Questo potrebbe aprire nuove prospettive sulla varietà delle risposte e potenzialmente portare a nuove intuizioni sull'autismo e sulla sua diagnosi e contribuire a una diagnosi più accurata e comprensiva dell'autismo.

Nel nostro caso, abbiamo scelto di utilizzare una tecnica di clustering rigido, in particolare il k-means, implementato tramite la libreria scikit-learn.

5.2 K-means:

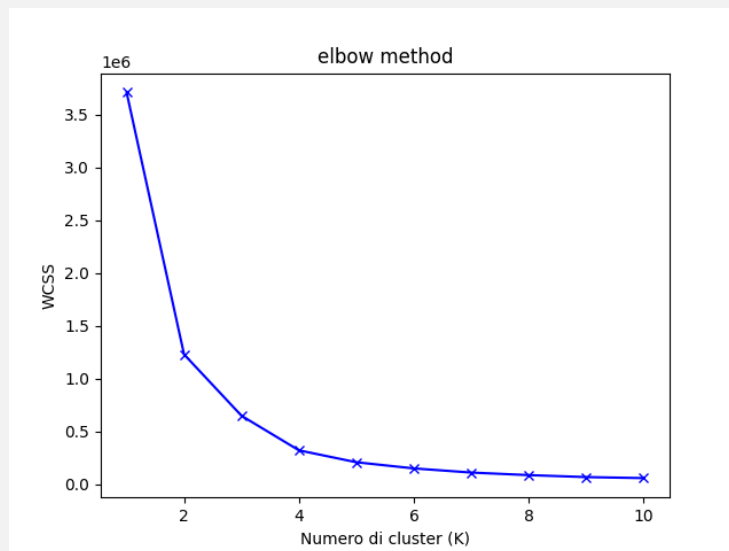
K-means è un algoritmo di clustering che suddivide un insieme di dati in K-cluster, dove K è un valore predefinito. Inizia posizionando casualmente K centroidi nel dataset e assegna ciascun punto al centroide più vicino. Successivamente, aggiorna la posizione dei centroidi in base alla media dei punti assegnati e ripete il processo fino a convergenza. L'obiettivo è minimizzare la somma dei quadrati delle distanze tra i punti e i centroidi assegnati, creando cluster di dati simili.

Dato che parte delle caratteristiche nel dataset sono di natura categorica e l'algoritmo k-means accetta solo caratteristiche numeriche, è stato necessario convertirle. A tal fine, abbiamo scelto di utilizzare la tecnica di codifica one-hot encoding. Questo approccio crea una nuova variabile binaria univoca per ogni categoria.

Una volta codificate le variabili categoriche, abbiamo dovuto determinare il numero ottimale di cluster.

```
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=40)
    kmeans.fit(dataset)
    wcss.append(kmeans.inertia_)
```

A tal scopo, abbiamo utilizzato la nota tecnica del metodo del gomito (elbow method), in cui sull'asse delle ordinate sono rappresentati i valori delle somme dei quadrati intra-cluster (WCSS), mentre sull'asse delle ascisse sono rappresentati i k cluster.



Come è possibile evincere dal grafico il gomito è piuttosto evidente; dunque, è stato deciso di prendere un valore di k cluster pari a 3.

Successivamente, il modello k-means è stato addestrato sui 3 cluster.

```
kmeans = KMeans(n_clusters=3, n_init=10, random_state=40)
kmeans.fit(dataset)
```

Le metriche scelte per la configurazione del modello sono state:

- **'n_cluster=3'**: il numero di cluster in cui il dataset deve essere diviso.
- **'random_state=40'**: questo parametro controlla la riproducibilità dei risultati. Fissando un valore alto per random_state, il modello fornirà sempre gli stessi risultati quando viene addestrato con gli stessi dati.
- **'n_init=10'**: specifica il numero di volte che l'algoritmo viene eseguito con diverse inizializzazioni casuali dei centroidi. L'algoritmo seleziona la soluzione migliore tra i tentativi basandosi sulla somma dei quadrati delle distanze. Un valore maggiore di n_init aumenta le probabilità di ottenere una soluzione di migliore qualità a scapito di un maggior tempo di calcolo. È stato scelto un valore non troppo alto, ma sufficiente per massimizzare la qualità dell'output.

5.3 Valutazione Finale:

Sono state valutate le prestazioni del modello utilizzando due metriche:

- **WCSS** (Within-Cluster Sum of Squares) che calcola la somma dei quadrati delle distanze di ogni punto rispetto al proprio centroide di cluster. Un valore più basso di WCSS indica una maggiore coesione all'interno dei cluster.
- **Silhouette Score**: valuta la coesione all'interno dei cluster e la separazione tra i cluster. È calcolato per ciascun punto e rappresenta il rapporto tra la distanza media al proprio cluster e la distanza media ai cluster più vicini. Un punteggio più alto del Silhouette Score indica una migliore separazione dei cluster.

```
WCSS: 1227706.6914349133  
Silhouette Score: 0.5865641889402072
```

In generale, con uno Silhouette Score pari a 0.586 è possibile affermare che i risultati presentano una separazione abbastanza buona tra i cluster, con una divisione omogenea dei dati.

6 CONCLUSIONE:

In questo progetto, abbiamo voluto esplorare il problema dell'autismo da diverse prospettive, utilizzando le tecniche di intelligenza artificiale per analizzare e classificare i dati relativi a questa condizione.

L'obiettivo di sviluppare un utile strumento ausiliare per diagnosticare la possibilità di spettro autistico attraverso l'uso di un questionario può dirsi raggiunto.

6.1 Sviluppi futuri:

Questo progetto lascia molte porte aperte per sviluppi futuri. Volendo proseguire ancora nell'ambito medico infatti basterebbe un nuovo dataset, quindi con nuovi dati adatti a riconoscere i tratti di altre "malattia" (da cambiare la parola), e con qualche piccola modifica al codice, per poter sfruttare questo progetto per creare altri strumenti ausiliari utili alle diagnosi. Un esempio potrebbe essere l'ambito dell'alzheimer.