Caratteristiche cliniche e di mutazione della classificazione del Glioma

Gruppo di lavoro:

- Giuseppe Grisolia, 736567, g.grisolia1@studenti.uniba.it

Link GitHub:
https://github.com/grisopatateecozze/progetto-Icon

AA 2022/2023

INDICE:

INTRODUZIONE:

1. DATASET:
    1.1. Descrizione dataset
    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
    3.1 "OwlReady2"

4 CLUSTERING:
    4.1 Decisioni progettuali
    4.2 K-means
    4.3 Valutazione Finale del modello

5. APPRENDIMENTO SUPERVISIONATO:
    5.1 Decisioni progettuali
    5.2 Metriche di valutazione
    5.3 Selezione delle Feature
    5.4 Preprocessing dei Dati e Divisione dei Dati
    5.5 K-Nearest Neighbors (KNN)
        5.5.1 Decisioni progettuali
        5.5.2 Ottimizzazione numero di vicini
        5.5.3 Addestramento e predizione
        5.5.4 Valutazione Finale del modello
    5.6 Random Forest
        5.6.1 Decisioni progettuali
        5.6.2 Ottimizzazione scelta numero alberi decisionali
        5.6.3 Valutazione finale del modello
    5.7 Support Vector machines
        5.7.1. Decisioni progettuali
        5.7.2 Ottimizzazione parametro gamma
        5.7.3 Valutazione Finale del modello
    5.8 Neural Network
        5.8.1 Decisioni progettuali
        5.8.2 Creazione modello e addestramento
        5.8.3 Valutazione Finale del modello

5.10 Considerazioni finali e confronto modelli

6 CONCLUSIONE

Introduzione

Questo progetto nasce con l'idea di predire una diagnosi di glioma di tipo LGG (Lower-Grade Glioma) o GBM (Glioblastoma Multiforme) sfruttando un dataset disponibile online.
Per fare ciò sono state implementate metodologie di apprendimento supervisionato e non supervisionato, e un'ontologia di base che consente di utilizzare un modello formale per rappresentare la realtà oggetto di analisi.
I gliomi sono i tumori primari più comuni del cervello. Possono essere classificati come LGG (Lower-Grade Glioma) o GBM (Glioblastoma Multiforme) a seconda dei criteri istologici/di imaging. Anche i fattori clinici e ***molecolari/di mutazione*** sono cruciali per il processo di classificazione.

1. Dataset

Ho scelto un set di dati relativo a pazienti affetti da glioma celebrale reperito dal sito www.archive.ics.uci.edu. Il set di dati è stato costruito sulla base dei progetti di glioma cerebrale TCGA-LGG e TCGA-GBM, e registra caratteristiche individuali, caratteristiche riguardanti mutazioni di geni e diagnosi.

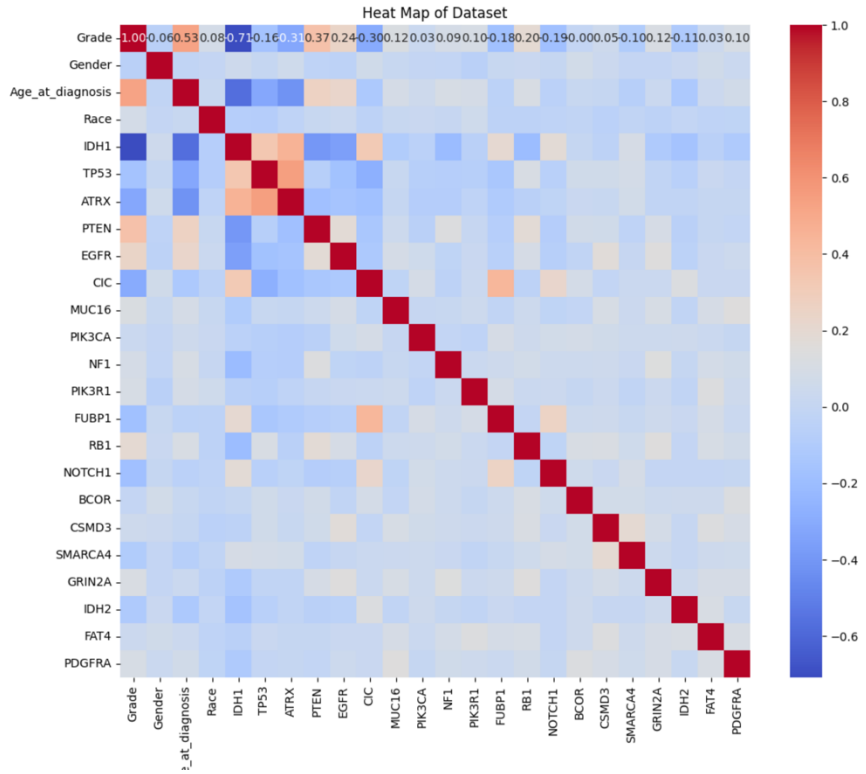
1.1 Descrizione dataset (dominio)

Questo dataset si basa su una raccolta longitudinale di 839 soggetti. Non presenta valori mancanti. Per ogni soggetto sono riportati i 20 geni mutati più frequentemente, le 3 caratteristiche cliniche dei progetti TCGA-LGG e TCGA-GBM e il grado del glioma, per un totale di 24 features. Se il soggetto presenta 1 in corrispondenza del grado del glioma è alta la possibilità che questo sia affetto da glioblastoma (GBM), 0 se è alta la possibilità che sia affetto da glioma di basso grado (LGG).

Feature	Role	Type	Description
Grade	target	Binary (0/1)	*Informazioni sulla classe del grado di glioma* (0 = "LGG"; 1 = "GBM")
Gender	feature	Binary (0/1)	*Genere* (0 = "maschio"; 1 = "femmina")
Age_at_diagnosis	feature	Integer (1-99)	*Età alla diagnosi con il numero di giorni calcolato*
Race	feature	Integer (0-3)	*Razza* (0 = "bianco"; 1 = "nero o afroamericano"; 2 = "asiatico"; 3 = "indiano americano o nativo dell'Alaska")
IDH1	feature	Binary (0/1)	*isocitrate dehydrogenase (NADP(+))1* (0 = "non mutato"; 1= "mutato")
TP53	feature	Binary (0/1)	*tumor protein p53* (0 = "non mutato"; 1= "mutato")
ATRX	feature	Binary (0/1)	*ATRX chromatin remodeler* (0 = "non mutato"; 1= "mutato")
PTEN	feature	Binary (0/1)	*phosphatase and tensin homolog* (0 = "non mutato"; 1= "mutato")
EGFR	feature	Binary (0/1)	*epidermal growth factor receptor* (0 = "non mutato"; 1= "mutato")
CIC	feature	Binary (0/1)	*capicua transcriptional repressor* (0 = "non mutato"; 1= "mutato")
MUC16	feature	Binary (0/1)	*mucin 16, cell surface associated* (0 = "non mutato"; 1= "mutato")
PIK3CA	feature	Binary (0/1)	*phosphatidylinositol-4,5-bisphosphate 3-kinase catalytic subunit alpha* (0 = "non mutato"; 1= "mutato")
NF1	feature	Binary (0/1)	*neurofibromin 1* (0 = "non mutato"; 1= "mutato")
PIK3R1	feature	Binary (0/1)	*phosphoinositide-3-kinase regulatory subunit 1* (0 = "non mutato"; 1= "mutato")
FUBP1	feature	Binary (0/1)	*far upstream element binding protein 1* (0 = "non mutato"; 1= "mutato")
RB1	feature	Binary (0/1)	*RB transcriptional corepressor 1* (0 = "non mutato"; 1= "mutato")
NOTCH1	feature	Binary (0/1)	*notch receptor 1* (0 = "non mutato"; 1= "mutato")
BCOR	feature	Binary (0/1)	*BCL6 corepressor* (0 = "non mutato"; 1= "mutato")
CSMD3	feature	Binary (0/1)	*CUB and Sushi multiple domains 3* (0 = "non mutato"; 1= "mutato")
SMARCA4	feature	Binary (0/1)	*SWI/SNF related, matrix associated, actin dependent regulator of chromatin, subfamily a, member 4* (0 = "non mutato"; 1= "mutato")
GRIN2A	feature	Binary (0/1)	*glutamate ionotropic receptor NMDA type subunit 2A* (0 = "non mutato"; 1= "mutato")
IDH2	feature	Binary (0/1)	*isocitrate dehydrogenase (NADP(+)) 2* (0 = "non mutato"; 1= "mutato")

FAT4	feature	Binary (0/1)	*FAT atypical cadherin 4* (0 = "non mutato"; 1= "mutato")
PDGFRA	feature	Binary (0/1)	*platelet-derived growth factor receptor alpha* (0 = "non mutato"; 1= "mutato")

1.2 Osservazione grafica dei dati



Ho creato una matrice di correlazione, ovvero una rappresentazione tabulare delle correlazioni tra le variabili. Questa rappresentazione grafica fornisce coefficienti di correlazione tra tutte le possibili coppie di variabili nel dataset. Questi coefficienti indicano quanto due variabili siano correlate tra loro. Un valore positivo indica una correlazione positiva (entrambe le variabili aumentano insieme), un valore negativo indica una correlazione negativa (una variabile aumenta mentre l'altra diminuisce), e un valore vicino a zero indica una scarsa correlazione.

2. Ontologia

Un'ontologia è un modello di conoscenza che fornisce una rappresentazione formale delle categorie di dati, delle loro interconnessioni e delle regole che governano il loro utilizzo all'interno di un dominio specifico. Nell'analisi del dominio si identificano i concetti principali, le relazioni tra essi e le loro proprietà e successivamente queste vengono formalizzate mediante un linguaggio di rappresentazione formale come, per esempio, OWL (Ontology Web Language).

2.1 Analisi Dominio

Ho deciso di dividere gli attributi del mio set di dati in:

- "informazioni importanti": ovvero gli attributi cruciali che devono essere rappresentati per immagazzinare le informazioni più importanti del set di dati
- "ciò che caratterizza le informazioni importanti": le proprietà delle entità rappresentate, ovvero le caratteristiche delle informazioni importanti

Non è stato scartato nessun attributo, perché tutti erano utili

2.1.1 Classi

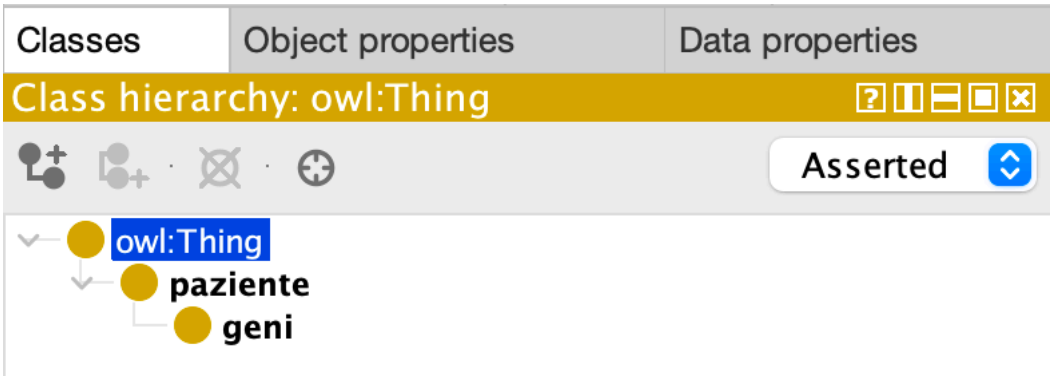
Dopo aver analizzato il dominio ho rappresentato questo con due classi/entity, rispettivamente geni sottoclasse di paziente.

-Paziente: che rappresenta l'insieme dei pazienti presenti nel dataset.
A questa ho associato diverse proprietà:

>*Age_of_diagnosis*: rappresenta età alla diagnosi
>*Gender:* rappresenta il genere del paziente
>*Race:* rappresenta la razza/etnia
>*Grade*: rappresenta il grado di glioma
>*nome dei geni*: stringa che identifica nome del gene

-Geni: che rappresenta i 20 geni mutati più frequentemente; a questa classe ho associato due proprietà:

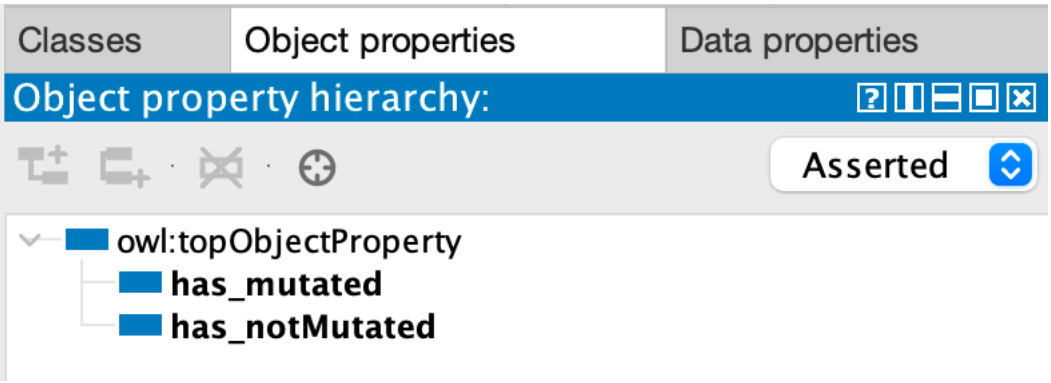
>*ID_gene*: rappresenta il nome del gene (sigla)
>*mutated*: rappresenta se è stato mutato (1) o meno (0)

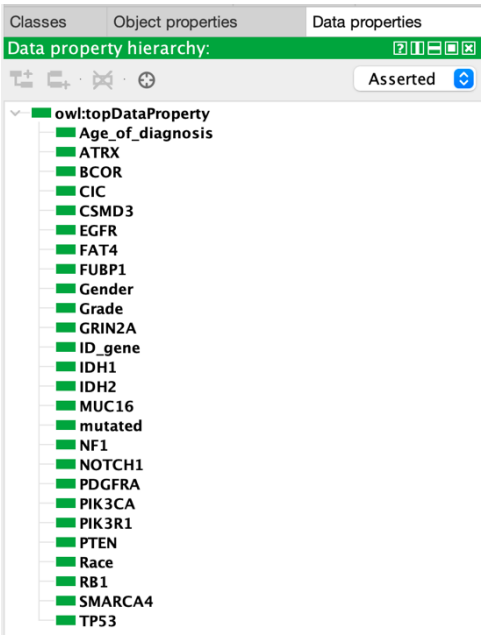


2.1.2 Object propetry

Un object propetry è un tipo di proprietà che specifica una relazione o connessione tra due individui o istanze in un'ontologia, che siano della stessa classe o meno.
Tra le classi ho definito una relazione che rappresenta come queste interagiscono tra loro:

>**has_mutated(paziente, gene) --> bool**: permette di verificare se un gene di un determinato paziente ha subito una mutazione.
>**has_notMutated(paziente, gene) --> bool**: permette di verificare se un gene di un determinato paziente ha subito una mutazione.





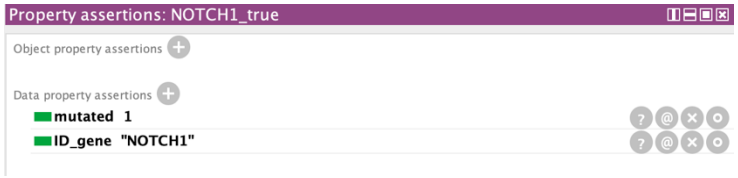
2.1.3 Data propetry

una data property definisce attributi o informazioni specifiche associate a un'istanza, mette in relazione un individuo con un valore primitivo.
Le data propetry dei geni sono: *ID_gene e mutated;*
quelle dei pazienti sono: *tutti i geni, Race, Gender, Grade e Age_of_diagnosis.*

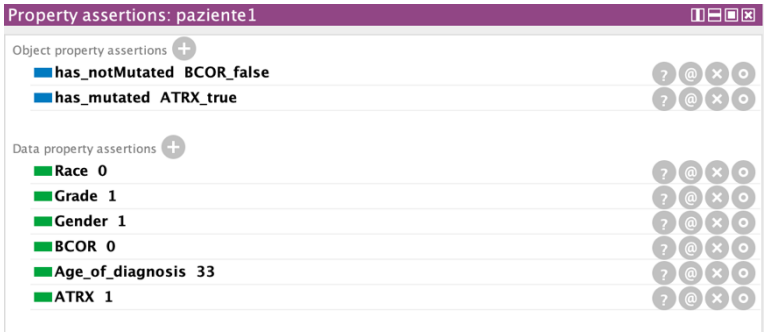
2.1.4 Individuals

Per alcune entità ho inserito delle istanze (individuals), per individuare pazienti a cui attribuire geni mutati e non, ed effettuare prove di query DL. Ho inserito solamente pochi geni tra le varie proprietà di ogni paziente, solamente per poter eseguire delle query funzionanti.

gene NOTCH1_true



paziente1

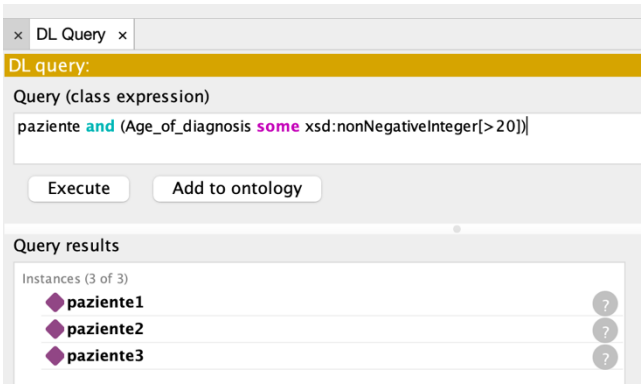


2.2 Software per la realizzazione dell'ontologia

Questa ontologia è stata realizzata con il software Protegè, un software open-source utilizzato per creare, modificare e gestire ontologie con standard OWL.

3. Query

Poi ho formulato delle query in linguaggio DL (Description Logics) per interrogare l'ontologia. DL è un linguaggio formale utilizzato principalmente per la modellazione concettuale delle ontologie nelle discipline dell'intelligenza artificiale.



3.1 "OwlReady2":

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

Dopo aver importato la libreria, questo codice estrae dall’ontologia la lista delle classi, delle object property, dei data property e degli individui che appartengono alle relative classi, e da i seguenti risultati.

```python
from owlready2 import *

print("ONTOLOGIA\n")
onto = get_ontology("Ontologia.owl").load()

# stampa il contenuto principale dell'ontologia
print("Class list in ontology---------------------------------------------------------------------------------------->\n")
print(list(onto.classes()), "\n")

# stampa le proprietà dell'oggetto
print("Object property in ontology---------------------------------------------------------------------------------->\n")
print(list(onto.object_properties()), "\n")

# stampa le proprietà dei dati
print("Object property in ontology---------------------------------------------------------------------------------->\n")
print(list(onto.data_properties()), "\n")

# stampa gli individui della classe paziente
print("paziente list in ontology------------------------------------------------------------------------------------>\n")
paziente = onto.search(is_a=onto.paziente)
print(paziente, "\n")

# stampa gli individui della classe domanda
print("individuals geni list in ontology---------------------------------------------------------------------------->\n")
geni = onto.search(is_a=onto.geni)
print(geni, "\n")
```

```python
# QUERY
print("_____________________________QUERY_____________________________________________________________________________")

# Query per estrarre i pazienti con glioma, di sesso femminile e che presentino il gene ATRX mutato ma non quello BCOR
query_result = list(onto.search(type=onto.paziente, Gender=1, Grade=1, ATRX=1, BCOR=0))
print("\n- Pazienti di sesso FEMMINILE che sono affetti da LGG e presentano ATRX mutato e BCOR non mutato:\n")
for paziente in query_result:
    print(paziente.name)

# Query per estrarre i pazienti con glioma, di sesso maschile e che presentino il gene ATRX mutato ma non quello BCOR
query_result = list(onto.search(type=onto.paziente, Gender=0, Grade=1, ATRX=1, BCOR=0))
print("\n- Pazienti di sesso MASCHILE che sono affetti da LGG e presentano ATRX mutato e BCOR non mutato:\n")
for paziente in query_result:
    print(paziente.name)

# query per estrarre i pazienti con glioma, di sesso femminile e con età alla diagnosi < 30 anni
pazienti_femmine_LGG_30_plus = \
    [p for p in onto.paziente.instances() if p.Age_of_diagnosis and int(p.Age_of_diagnosis[0]) > 30 and 1 in p.Grade and 1 in p.Gender]
print("\n- Pazienti femmine affette da LGG con età > 30:\n")
for paziente in pazienti_femmine_LGG_30_plus:
    print(paziente)

# query per estrarre i pazienti con glioma, di sesso maschile e con età alla diagnosi < 30 anni
pazienti_femmine_LGG_30_plus = \
    [p for p in onto.paziente.instances() if p.Age_of_diagnosis and int(p.Age_of_diagnosis[0]) > 30 and 1 in p.Grade and 0 in p.Gender]
print("\n- Pazienti marchi affetti da LGG con età > 30:\n")
for paziente in pazienti_femmine_LGG_30_plus:
    print(paziente)
```

Dopo mediante questo codice ho interrogato l'ontologia <-----

```
BP1, Ontologia.GRIN2A, Ontologia.Gender, Ontologia.Grade, Ont

aziente6, Ontologia.geni, Ontologia.ATRX_false, Ontologia.AT

_true, Ontologia.CSMD3_false, Ontologia.CSMD3_true, Ontologi
```

Il codice precedente restituisce come risultato:
-la lista dei pazienti di sesso femminile che sono affetti da LGG e presentano un determinato gene mutato e un altro no (che in questo caso sono i geni ATRX e BCOR).
-la lista dei pazienti di sesso maschile che sono affetti da LGG e presentano un determinato gene mutato e un altro no (che in questo caso sono sempre i geni ATRX e BCOR).
-I casi di uomini con LGG ed età superiore a 30 anni e stessa cosa per le donne. Da queste query effettuate come esempio è venuto alla luce il grande potenziale di questa ontologia e delle query eseguite direttamente da codice, che potrebbero sfruttarsi per ottenere rilevanti informazioni sul mio caso di studio.

```
_____________________________QUERY_____________________________________________________________________________

- Pazienti di sesso FEMMINILE che sono affetti da LGG e presentano ATRX mutato e BCOR non mutato:

paziente1
paziente4

- Pazienti di sesso MASCHILE che sono affetti da LGG e presentano ATRX mutato e BCOR non mutato:

paziente5

- Pazienti femmine affette da LGG con età > 30:

Ontologia.paziente1
Ontologia.paziente6

- Pazienti marchi affetti da LGG con età > 30:

Ontologia.paziente5

Process finished with exit code 0
```

4. Clustering

Sono partito svolgendo un'analisi non supervisionata tramite clustering, la quale può fornire una panoramica completa dei dati, consentendomi di identificare eventuali outlier, relazioni tra feature e possibili segmentazioni della popolazione oggetto di studio.
Il clustering puo 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.

4.1 Decisioni progettuali

Questa tecnica è stata usata per raggruppare i pazienti in categorie basate sulle caratteristiche presenti nel file 'TCGA_InfoWithGrade.csv', come le varie mutazioni genetiche che i pazienti possono avere in comune. L'obiettivo è individuare dei cluster, ovvero gruppi di esempi simili a centroidi calcolati in modo automatico. Lo scopo di questa tecnica nel progetto è quello di creare una nuova colonna da aggiungere alle features relative alle mutazioni genetiche, in un nuovo file chiamato 'TCGA-clusters.csv'. Questo potrebbe aprire nuove prospettive sulla varietà di mutazioni e correlazioni tra esse e il grado di glioma, e contribuire a delle diagnosi più accurate.
Nel mio caso ho deciso di utilizzare una tecnica di clustering rigido: il **k-means**, che ho implementato tramite la libreria scikit-learn.

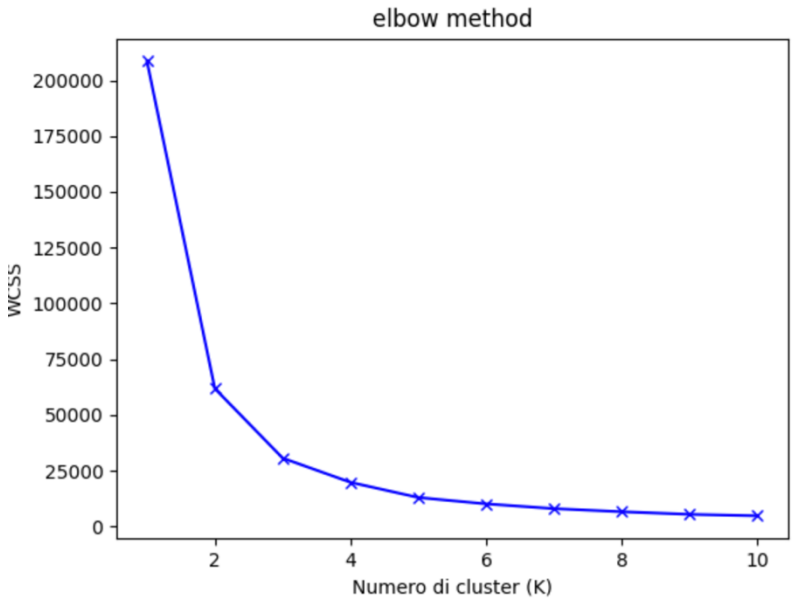
4.2 K-means

K-means è un algoritmo con l'obiettivo principale di suddividere i dati in modo che gli oggetti all'interno dello stesso cluster siano simili tra loro, mentre gli oggetti in cluster diversi siano dissimili. Basicamente funziona così:
Si sceglie un numero predefinito di cluster (k) prima di eseguire l'algoritmo.
Vengono selezionati casualmente k punti dati come centroidi iniziali dei cluster.
Per ciascun punto dati nel dataset, viene calcolata la distanza tra questo e tutti i centroidi e viene assegnato al cluster il cui centroide è più vicino (di solito in base alla distanza euclidea). Successivamente si aggiornano i centroidi di ciascun cluster calcolando la media dei punti dati assegnati a quel cluster. Si ripete il processo fino a convergenza.
L'output dell'algoritmo K-Means sono i centroidi finali e l'assegnazione dei punti dati ai cluster corrispondenti. Creando così clusters di dati simili.

Ho dovuto determinare il numero ottimale di cluster con questo algoritmo:

```
24  wcss = []
25  for k in range(1, 11):
26      kmeans = KMeans(n_clusters=k, n_init=10, random_state=40)
27      kmeans.fit(dataset)
28      wcss.append(kmeans.inertia_)
29
```

A tal scopo ho utilizzato il 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, ho addestrato il modello k-means sui 3 cluster.

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

Dove i parametri scelti per la configurazione del modello indicano:

- **'n_clusters'**: il numero dei cluster in cui dividere i dati del dataset
- **'n_init'**: specifica il numero di volte che l'algoritmo sarà eseguito con diverse inizializzazioni casuali dei centroidi. Un valore maggiore aumenta le probabilità di ottenere una soluzione di migliore qualità a scapito di un maggior tempo di calcolo. Ho scelto un valore non troppo alto, ma sufficiente per massimizzare la qualità dell'output.
- **'random_state'**: questo parametro controlla la riproducibilità dei dati. Fissandolo ad un valore alto, il modello fornirà gli stessi risultati quando viene addestrato con gli stessi dati.

4.3 Valutazioni finali

Ho infine valutato le prestazioni del modello utilizzando due metriche molto comuni:

• **WCSS** (Within-Cluster Sum of Squares) ovvero la somma dei quadrati delle distanze di ogni punto rispetto al proprio centroide di cluster. Più basso è il valore di WCSS maggiore è la coesione all'interno dei cluster.
• **Silhouette Score**: 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: 30608.951016497434
Silhouette Score: 0.5366883251899834

Process finished with exit code 0
```

In generale, con uno Silhouette Score pari a 0.536 è possibile affermare che i risultati presentano una separazione abbastanza buona tra i cluster, con una divisione omogenea dei dati.

5 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. Questo viene prima addestrato con una serie di esempi ideali, costituiti da coppie di input e di output, e dovrebbe diventare in grado di predire output attesi, per dei nuovi dati input forniti nella stessa forma di quelli precedenti sui quali è avvenuto l'addestramento.

5.1 Decisioni progettuali

In questo progetto, l'obiettivo affidato all'apprendimento supervisionato è risolvere un problema di classificazione, in particolare utilizzando la colonna "Grade" come variabile **'target'** da predire, basandosi sulle altre caratteristiche dei pazienti, ovvero le loro mutazioni genetiche.
La prima cosa che ho fatto è stata decidere quali modelli di apprendimento supervisionato utilizzare. Ho scelto gli algoritmi con le migliori prestazioni, in quanto l'elevata sensibilità del tema, quale il cancro al cervello, necessitava di una grande precisione nelle valutazioni. (Ad esempio, per questo ho preferito scegliere il classificatore Random Forest, con maggiore precisione ad un Decision Tree e alla sua minore complessità e conseguente minore precisione.)

Questi sono i 4 algoritmi che ho scelto:

1. **K-NN (k nearest neighbors)**
2. **Random Forest**
3. **SVM (Support Vector Machine)**
4. **Neural Network**

5.2 Metriche di valutazione

Ogni algoritmo implementato è stato valutato mediante 5 metriche e successivamente sono stati prodotti dei grafici per visualizzare appunto le valutazioni fatte in maniera più semplice. Le 5 metriche di valutazione che ho usato sono:

1. **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).
2. **Precision-recall curve:** Questa curva rappresenta il trade-off tra la precisione e il recall di un modello di classificazione binaria.
3. **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.
4. **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).
5. **La cross-validation:** 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.

5.3 Selezione delle features

Non ho ritenuto nessuna delle features del mio dataset irrilevante, in quanto erano, secondo me, tutte utili e impattanti sulla diagnosi. Il mio dataset era composto da:
-**geni** (IDH1, TP53, ATRX, ... PDGFRA) nonché le features principali, che rappresentano il metro di giudizio principale.
-**Età alla diagnosi del cancro** ('Age_of_diagnosis'), **genere** ('Gender') e **razza** ('Race'), sono features che possono essere rilevanti, perché possono influenzare la prevalenza e la manifestazione del cancro. (es: si manifesta tipicamente nelle persone adulte/di genere femminile o di una specifica razza/etnia).
-**Grado del glioma** ('Grade'), ovvero la feature target che specifica il grado del glioma LGG o GBM.

5.4 Preprocessing dei dati e divisione dati

L'addestramento di ogni modello inizia con l'esecuzione del preprocessing dei dati.
Un'operazione fondamentale e molto comune nel preprocessing dei dati è la dummificazione delle features. Operazione che consiste nell'eseguire la codifica one-hot, processo che converte le variabili categoriche in una rappresentazione binaria, in modo che ciascuna categoria venga rappresentata come una colonna binaria separata. Tutto cio nel mio caso non è servito perché nel dataset erano presenti solamente variabili booleane e numeriche.
Dato che il target della nostra predizione è "Grade", la colonna relativa a questo campo viene eliminata dal dataframe con il metodo 'drop()' e viene inserita invece in una nuova variabile.

```
29  # divisione dati in features di input e feature target
30  y = dataset['Grade']
31  X = dataset.drop( labels: 'Grade', axis=1)  # 'Grade' è la colonna target
```

Successivamente si divide il dataset in 80% dati per l'addestramento di ogni modello e 20% di dati per il test di ogni modello. Ho aggiunto anche per questo modello come parametri della funzione di configurazione anche **'shuffle=True'** così da mischiare le righe del mio dataset, in quanto erano ordinate rispetto al grado di glioma (si trovava dall'inizio fino a metà dataset tutti i pazienti con LGG e dalla metà fino al fondo tutti i pazienti con GBM.) e **'stratify=y' per cercare di** mantenere la stessa proporzione di classi o target nei set di addestramento e test, che esiste nel set di dati originale y. Questo è particolarmente utile quando alcune classi sono rappresentate in modo significativamente maggiore o minore rispetto ad altre. L'uso di stratify aiuta a evitare che una delle classi venga sottorappresentata nei set di addestramento o test, garantendo così una valutazione più accurata del modello.

```
31  # divido il dataset in set di addestramento e test
32  X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.20, random_state=42, shuffle=True, stratify=y)
```

Questa suddivisione è il giusto compromesso tra la necessità di avere abbastanza dati sia per l'addestramento e sia per un test accurato del modello.
Scegliendo una proporzione di dati più alta per il set di test, avremmo avuto meno dati disponibili per l'addestramento del modello e questo avrebbe potuto comportare una minore capacità del modello di apprendere le complessità dei dati e avrebbe potuto portare a una generalizzazione meno accurata.
Invece 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 poco in grado di generalizzare su nuovi dati.

5.6 K-Nearest Neighbors (K-NN)

Il k-nn è un algoritmo di classificazione che cerca di determinare la classe di appartenenza di un dato di input, cercando tra tutti gli esempi di addestramento, quello più vicino al dato input in base alla distanza euclidea.

5.6.1 Decisioni progettuali

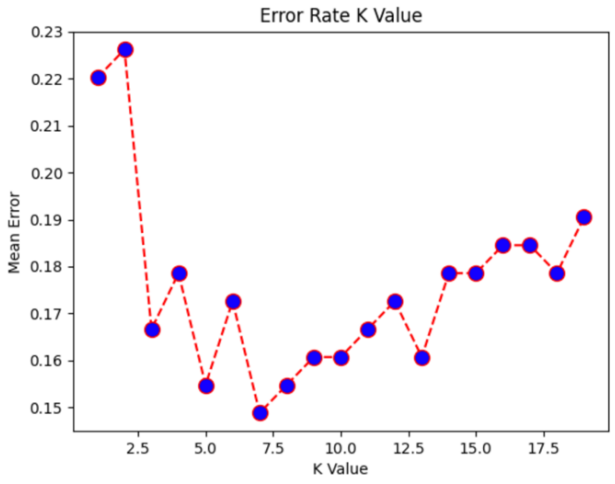
Ho realizzato inizialmente questo modello, perché abbastanza semplice da implementare, versatile, in quanto ha ottime prestazioni con dataset di qualsiasi dimensioni ed è molto robusto in presenza di dati rumorosi e valori anomali. Questo perché il modello si basa sui vicini più prossimi, e quindi un valore anomalo avrà meno impatto sull'output finale.

5.6.2 Ottimizzazione numero di vicini

Prima di implementare il mio knn, ho cercato il numero di vicini ottimale per il suddetto modello. Un valore k che equilibri la complessità del modello con la sua **capacità di generalizzazione**. Ovvero trovare un k per il quale si minimizzi l'errore di previsione del modello. Per fare cio ho utilizzato questo algoritmo :

```python
# calcolo del numero di vicini ottimale da utilizzare per il knn, sulla base di chi da il minor valore di mean error
error = []
for i in range(1, 30):
    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))

# grafico che mostra i cambiamenti dell'mean error, al cambiare del numero di vicini
plt.plot( *args: range(1, 30), error, color='red', linestyle='dashed', marker='o', markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
plt.show()
```

Dalla rappresentazione grafica si capisce benissimo che il valore dell'errore medio è più basso quando k=7. Quindi l'ho scelto come il giusto numero di vicini per ottenere un buon equilibrio tra overfitting e underfitting. Perché un k troppo piccolo potrebbe causare overfitting, mentre uno troppo grande potrebbe causare underfitting.



5.6.3 Addestramento e predizione

Scelto il numero ideale di vicini, addestro il modello coinvolgendo la raccolta di dati di addestramento del mio dataset e il calcolo delle distanze tra i punti dati.

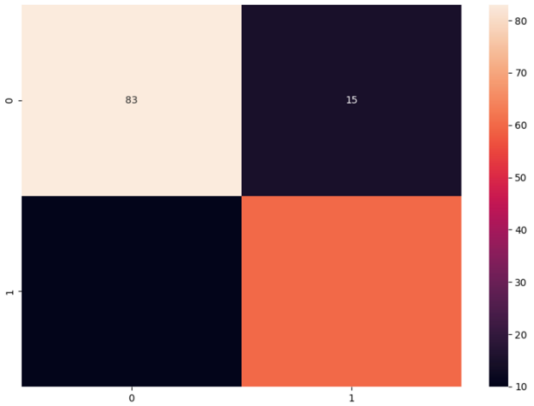
```python
# addestramento del modello con il numero di vicini ottimale trovato (7)
neigh = KNeighborsClassifier(n_neighbors=7)
neigh.fit(X_train, y_train)
```

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.

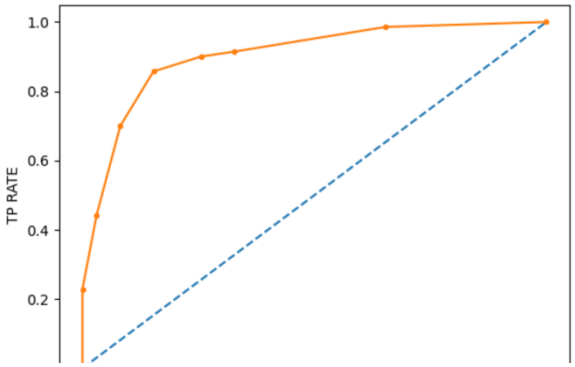
```
55  # effettua previsioni sul test set
56  prediction = neigh.predict(X_test)
57  accuracy = accuracy_score(y_test, prediction)
58  print(f"accuracy_score: {accuracy:.2f}")
```

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** (accuratezza).

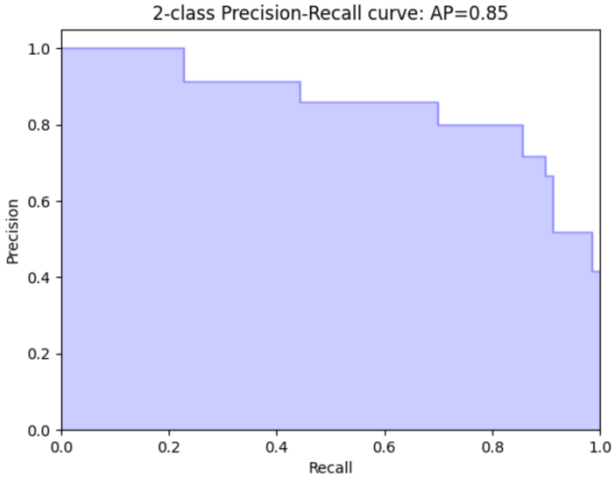
5.6.4 Valutazione finale del modello



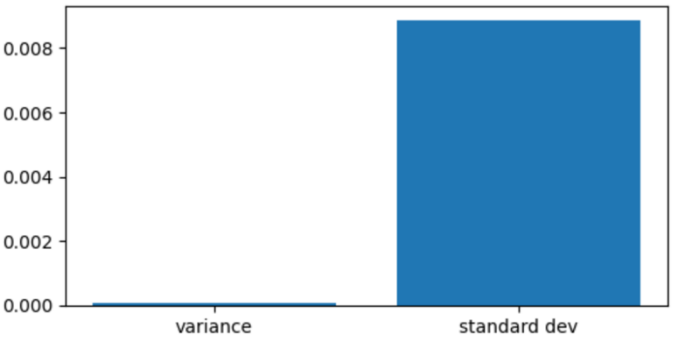
Matrice di confusione: si deduce che il modello ha una buona capacità di fare previsioni corrette per entrambe le classi. Si nota che i falsi positivi (15) sono leggermente più elevati dei falsi negativi (10), ciò potrebbe suggerire che il modello è leggermente incline a effettuare previsioni corrette quando la classe reale è negativa.



ROC Curve: L'Area Under the Curve (AUC) nella ROC Curve è estremamente positiva, con un valore di 0.904. 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 dell'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.

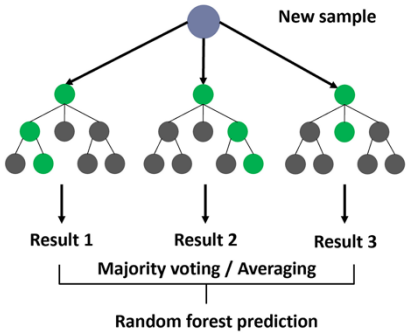
Di seguito anche la stampa del **classification report** e i dei risultati stampati nel terminale, tra cui: esplorazione del dataset, **cv_scores_mean**, **cv_scores_variance**, **cv_score_devstandard**, **AUC** e **f1_score**.

```
RangeIndex: 839 entries, 0 to 838
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Grade             839 non-null    int64
 1   Gender            839 non-null    int64
 2   Age_at_diagnosis  839 non-null    float64
 3   Race              839 non-null    int64
 4   IDH1              839 non-null    int64
 5   TP53              839 non-null    int64
 6   ATRX              839 non-null    int64
 7   PTEN              839 non-null    int64
 8   EGFR              839 non-null    int64
 9   CIC               839 non-null    int64
 10  MUC16             839 non-null    int64
 11  PIK3CA            839 non-null    int64
 12  NF1               839 non-null    int64
 13  PIK3R1            839 non-null    int64
 14  FUBP1             839 non-null    int64
 15  RB1               839 non-null    int64
 16  NOTCH1            839 non-null    int64
 17  BCOR              839 non-null    int64
 18  CSMD3             839 non-null    int64
 19  SMARCA4           839 non-null    int64
 20  GRIN2A            839 non-null    int64
 21  IDH2              839 non-null    int64
 22  FAT4              839 non-null    int64
 23  PDGFRA            839 non-null    int64
dtypes: float64(1), int64(23)
memory usage: 157.4 KB
None
accuracy_score: 0.85
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.85      0.87        98
           1       0.80      0.86      0.83        70

    accuracy                           0.85       168
   macro avg       0.85      0.85      0.85       168
weighted avg       0.85      0.85      0.85       168


Confusion matrix:
 [[83 15]
 [10 60]]

cv_scores mean:0.827195608782435

cv_score variance:7.855200837072869e-05

cv_score dev standard:0.00886295708952315

AUC: 0.904

f1 score:  0.8275862068965518

Process finished with exit code 0
```

5.7 Random Forest

Il Random Forest è un algoritmo che rappresenta un 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 accurato che voglio ottenere in questo progetto.



5.7.1 Decisioni progettuali

Ho deciso di implementare anche una Random Forest perché queste sono notoriamente accurate grazie alla combinazione di molteplici alberi decisionali. Questa aggregazione di predizioni da alberi diversi riduce il rischio di overfitting e migliora la stabilità delle previsioni. Anche perché possono gestire dataset di grandi dimensioni in modo efficiente, grazie alla parallelizzazione e alla suddivisione del lavoro tra gli alberi.
I parametri scelti per la configurazione del modello indicano:

- '**n_estimators'**: ho scelto di impostare a 20 questo parametro, che indica il numero di alberi decisionali che vengono creati, in modo da trovare un equilibrio tra prestazioni e efficienza computazionale, perché un numero maggiore di alberi avrebbe potuto dare prestazioni migliori, ma anche aumentare il tempo di addestramento.
- '**max_depth'**: questo parametro indica la profondità massima degli alberi decisionali all'interno dell'ensemble. Ho scelto 30 in quanto è un valore che permette agli alberi di adattarsi sui 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.

```
69  # Addestramento del classificatore Random Forest prova con parametri casuali
70  # clf1 = RandomForestClassifier(max_depth=30, random_state=0, n_estimators=20)
```

5.7.2 Ottimizzazione scelta n. alberi decisionali

Utilizzando il modello con i parametri casuali descritti sopra ho ottenuto questi risultati:

```
Clasification report:
              precision    recall  f1-score   support

           0       0.79      0.83      0.81        98
           1       0.74      0.70      0.72        70

    accuracy                           0.77       168
   macro avg       0.77      0.76      0.77       168
weighted avg       0.77      0.77      0.77       168


Confussion matrix:
 [[81 17]
 [21 49]]

cv_scores mean:0.8212075848303394

cv_score variance:5.951547554611757e-05

cv_score dev standard:0.00771462737571411


AUC: 0.896

f1 score:  0.7205882352941176

Process finished with exit code 0
```

In seguito, ho capito che avrei potuto ottenere risultati e prestazioni di questo modello, ancor più validi di quelli già riscontrati. Per fare questo ho usato un algoritmo che mi aiutasse a scegliere i migliori parametri in input per la funzione di creazione del modello (max_depth e random_state).

```python
# 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 definisce inizialmente due liste di valori da testare per max_depth e random_state. Poi viene eseguita una valutazione di k-fold cross validation su ogni combinazione di valori di max_depth e random_state e viene calcolata la media delle valutazioni ottenute e si sceglie la combinazione che restituisce i valori migliori.

```
Valore migliore max_depth: 4.0
Valore migliore random_state: 16.0
```

Quindi l'opzione migliore è addestrare un Random Forest, utilizzando come parametri input "max_depth" e "random_state" appena ottenuti, come vediamo dalle immagini seguenti i risultati sono migliorati.

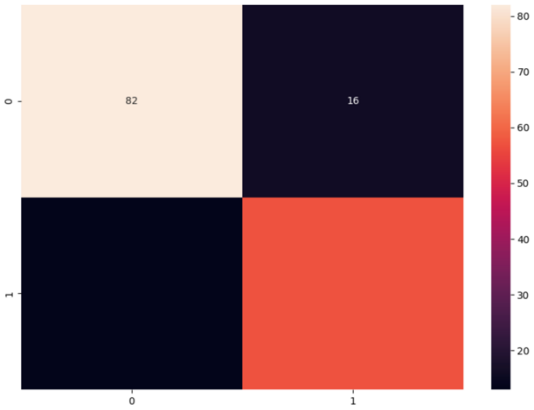
PRIMA:

```
Clasification report:
              precision    recall  f1-score   support

           0       0.79      0.83      0.81        98
           1       0.74      0.70      0.72        70

    accuracy                           0.77       168
   macro avg       0.77      0.76      0.77       168
weighted avg       0.77      0.77      0.77       168


Confussion matrix:
 [[81 17]
 [21 49]]

cv_scores mean:0.8212075848303394

cv_score variance:5.951547554611757e-05

cv_score dev standard:0.00771462737571411


AUC: 0.896

f1 score:  0.7205882352941176

Process finished with exit code 0
```

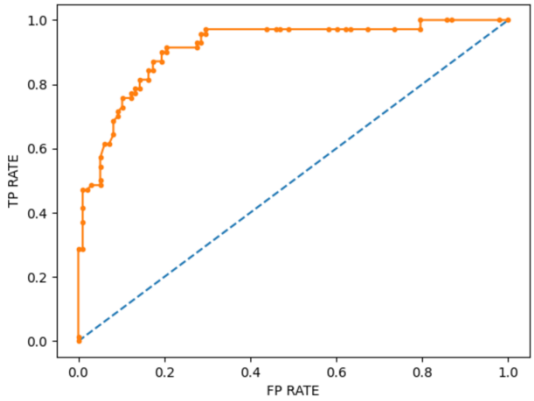
DOPO:

```
Clasification report:
              precision    recall  f1-score   support

           0       0.86      0.84      0.85        98
           1       0.78      0.81      0.80        70

    accuracy                           0.83       168
   macro avg       0.82      0.83      0.82       168
weighted avg       0.83      0.83      0.83       168


Confussion matrix:
 [[82 16]
 [13 57]]

cv_scores mean:0.8534217279726262

cv_score variance:0.00014390065849806754

cv_score dev standard:0.01199586005662235


AUC: 0.914

f1 score:  0.7972027972027972

Process finished with exit code 0
```

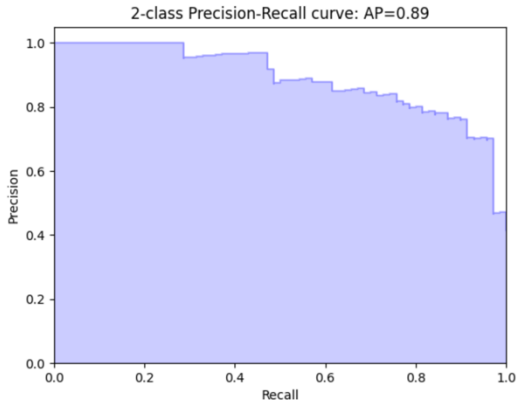
5.7.3 Valutazioni finali del modello



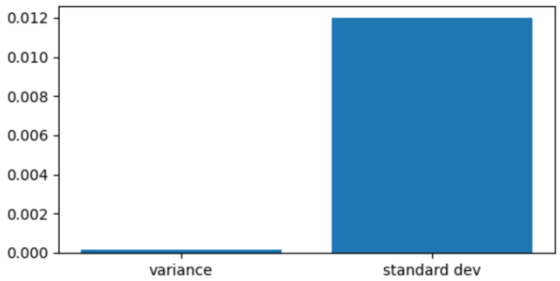
Matrice di confusione: mostra che ci sono pochi falsi positivi, solo 16 a fronte di 82 classificazione corrette, e meno falsi negativi, solo 13 su un totale di 57 previsioni, il che indica una buona capacità del modello nel limitare gli errori di classificazione.



ROC curve: L'Area Under the Curve (AUC) nella ROC Curve è estremamente positiva, con un valore di 0.914. Questo indica che il modello Random Forest ha un'eccellente capacità di discriminare tra le classi positive e negative.



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, ma alle volte etichetta erroneamente alcuni campioni.



Bar Chart di Varianza e Deviazione Standard:
il modello mantiene un livello abbastanza alto 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.

5.8 Support Vector Machine

Le Support Vector Machines (SVM), o macchine a vettori di supporto, sono un tipo di algoritmo di apprendimento automatico utilizzato principalmente per problemi di classificazione e regressione. L'obiettivo principale di una SVM è trovare un iperpiano (una sorta di superficie decisionale) nello spazio dei dati che separi in modo ottimale le diverse classi di punti o predica in modo ottimale i valori target in un problema di regressione.

5.8.1 Decisioni progettuali

Ho scelto questo modello in quanto potente per la classificazione binaria, in particolare quando le classi sono ben separate; può anche gestire relazioni non-lineari attraverso l'uso di kernel.
Ho 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.

5.8.2 Ottimizzazione parametro gamma

L'iperparametro gamma nel kernel RBF influenza la forma e l'ampiezza della regione di influenza di ciascun punto dati sull'allocazione dei vettori di supporto e quindi sull'iperpiano di separazione. Un valore basso di gamma porta a una regione di influenza ampia, mentre un valore alto di gamma porta a una regione di influenza più stretta e focalizzata. In altre parole, un valore basso di gamma rende l'iperpiano di separazione più influenzato da punti distanti, mentre un valore alto di gamma lo rende più influenzato solo da punti vicini. L'obiettivo di questo codice è trovare il valore ottimale di gamma:

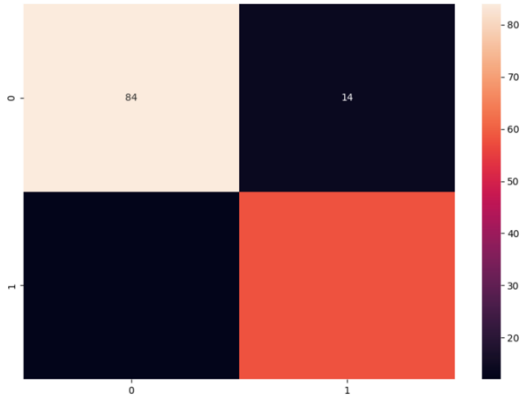
```python
# -------------------scelta valore gamma-----------------------------------
# Definisco i valori di gamma da testare
param_grid = {'gamma': [1e-4, 1e-3, 0.01, 0.1, 1, 10, 100]}

# 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(X_train, y_train)

# Accesso ai risultati della ricerca a griglia
results = grid_search.cv_results_

# Stampa dei valori di gamma testati e delle prestazioni corrispondenti
for gamma, mean_score in zip(results['param_gamma'], results['mean_test_score']):
    print(f"Gamma: {gamma:<10}Mean Score: {mean_score}")

# Stampa il valore di gamma migliore
print("\nMiglior valore di gamma:", grid_search.best_params_['gamma'])
```

Per fare ciò, si utilizza la tecnica della "grid search" con cross-validation. Si inizia col definire una griglia di valori di gamma e, 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 -->

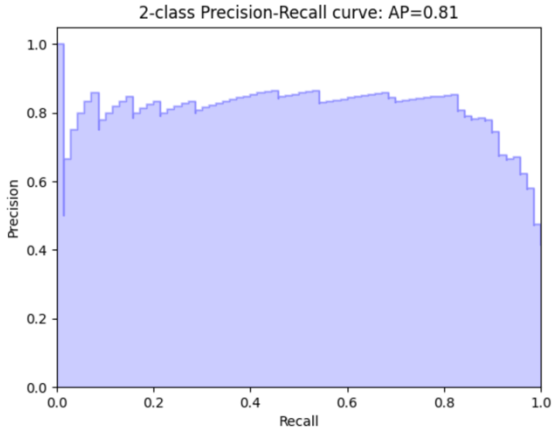
```
Gamma: 0.0001    Mean Score: 0.7331122166943062
Gamma: 0.001     Mean Score: 0.7390713101160863
Gamma: 0.01      Mean Score: 0.80767274737424
Gamma: 0.1       Mean Score: 0.8449530127142069
Gamma: 1         Mean Score: 0.7585185185185186
Gamma: 10        Mean Score: 0.5857048092868988
Gamma: 100       Mean Score: 0.5797346600331674

Miglior valore di gamma: 0.1
```

Questi risultati ci fanno capire che il modello sarà ottimizzato per il valore 0.1 di gamma.

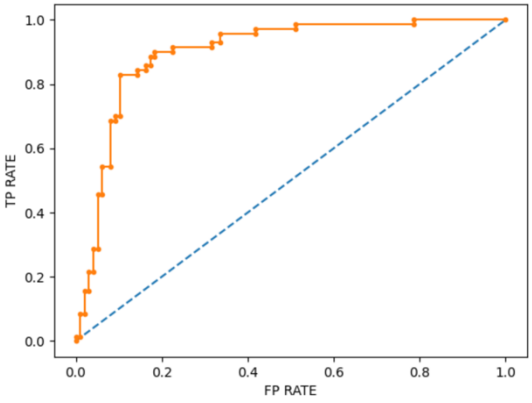
5.8.3 Valutazione finale del modello



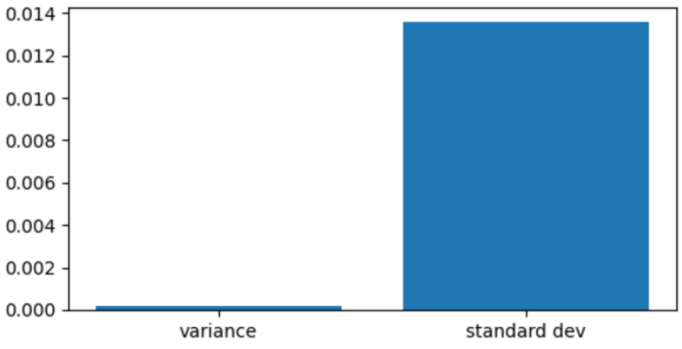
Matrice di Confusione: mostra come il modello abbia effettuato nel complesso delle buone classificazioni, nonostante qualche errore, giustificabile in parte però dalla quantità di previsioni effettuate.



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



ROC curve: L'Area Under the Curve (AUC) nella ROC Curve è molto positiva, con un valore di 0.899. Questo indica che il modello Random Forest ha un'eccellente capacità di discriminare tra le classi positive e negative.



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.

5.9 Neural Network

Le reti neurali sono un modello di machine learning. Il loro nome e la loro struttura sono ispirati al cervello umano, imitando il modo in cui i neuroni biologici si inviano segnali. Sono composte da livelli di nodi che contengono un livello di input, uno o più livelli nascosti e un livello di output. Ciascun nodo, o neurone artificiale, si connette ad un altro e ha un peso e una soglia associati. Se l'output di qualsiasi singolo nodo è al di sopra del valore di soglia specificato, tale nodo viene attivato, inviando i dati al successivo livello della rete. In caso contrario, non viene passato alcun dato al livello successivo della rete.

5.9.1 Decisioni progettuali

Ho scelto infine questo modello per la sua grande capacità di affrontare problemi di apprendimento profondi e complessi e per la capacità di modellare relazioni non lineari e catturare strutture complesse nei dati. Inoltre, una delle caratteristiche distintive delle reti neurali è la loro capacità di apprendere automaticamente le caratteristiche dai dati, eliminando la necessità di progettare manualmente estrattori di caratteristiche. Questo semplifica il processo di sviluppo del modello.

5.9.2 Creazione modello e addestramento

```
2 usages
def create_model():
    model = keras.Sequential([
        keras.layers.Input(shape=(X_train.shape[1],)),
        keras.layers.Dense( units: 64, activation='relu'),
        keras.layers.Dense( units: 32, activation='relu'),
        keras.layers.Dense( units: 1, activation='sigmoid')

    ])

    # Compilazione del modello
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

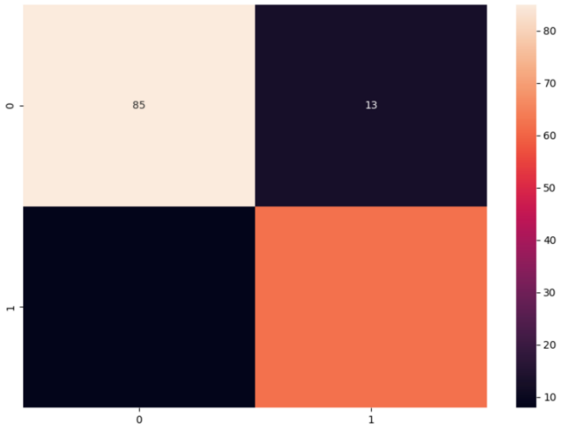
Ho scelto di definire una rete neurale sequenziale con tre livelli con il framework **Keras**. Il primo livello, con 64 neuroni nascosti e utilizza la funzione di attivazione ReLU, il secondo con altri 32 neuroni nascosti e sempre la funzione di attivazione ReLU e il terzo ha un singolo neurone con una funzione di attivazione sigmoide, che produce un valore compreso tra 0 e 1 per la classificazione binaria.

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 64 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. Stessa cosa per il secondo livello ma con un numero più basso di neuroni nascosti = 32. Il terzo 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 ambiti.
Si crea quindi un'istanza del modello e inizia l'addestramento sui dati.

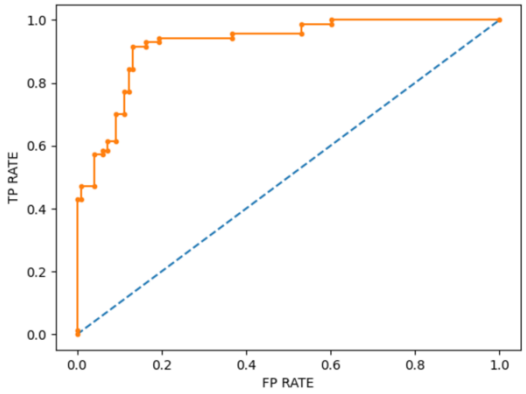
```
# creazione istanza model1 e addestramento
model1 = create_model()
model1.fit(X_train, y_train, epochs=30, batch_size=64, validation_split=0.2)
```

Dove i parametri in input sono:
-"**epochs**" specifica quante volte l'intero dataset di addestramento verrà utilizzato per aggiornare i pesi del modello. go deciso di addestrare il modello per "30 epochs", dato che maggiori epoche potrebbero consentire al modello di adattarsi meglio ai dati di addestramento, ma forse troppo e magari portare all'overfitting.
-"**batch_size**" definisce la dimensione del batch, cioè quante istanze di addestramento vengono utilizzate prima di aggiornare i pesi del modello. Abbiamo scelto di utilizzare un batch di dimensione "64", perché rappresenta un buon compromesso tra l'accuratezza del modello e l'efficienza di addestramento. Fornire abbondante o esagerata quantità di dati per apprendere modelli complessi potrebbe richiedere troppo tempo computazionale.

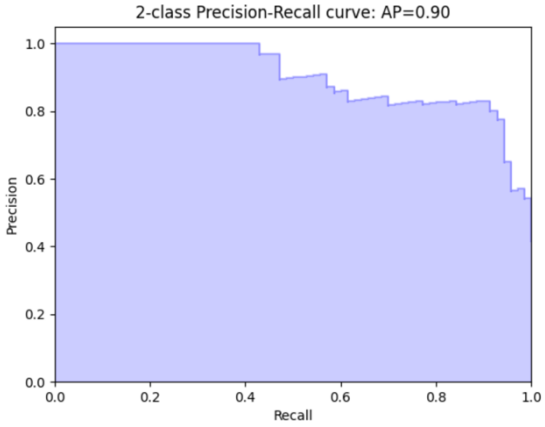
5.9.3 Valutazione finale del modello



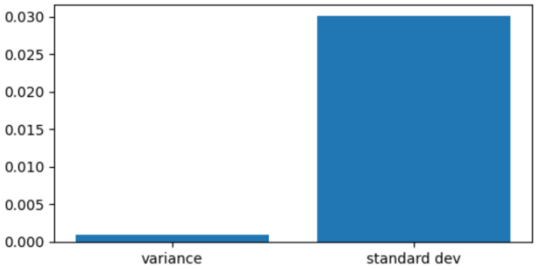
Matrice di Confusione: il grafico mostra che il modello ha previsto correttamente la maggior parte delle istanze sia per la classe "False" che per la classe "True", garantendo affidabilità nonostante alcuni errori.



ROC curve: L'Area Under the Curve (AUC) nella ROC Curve è molto positiva, con un valore di 0.924. Questo indica che il modello neural network ha un'eccellente capacità di discriminare tra le classi positive e negative.



Precision-Recall Curve: il grafico indica un buon bilanciamento tra precision e recall. 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.

```
cv_scores mean:0.849452736318408

cv_score variance:0.0009066706269646797

cv_score dev standard:0.03011097187014527
```

classification report:

```
Clasification report:
              precision    recall  f1-score   support

           0       0.91      0.87      0.89        98
           1       0.83      0.89      0.86        70

    accuracy                           0.88       168
   macro avg       0.87      0.88      0.87       168
weighted avg       0.88      0.88      0.88       168


Confussion matrix:
 [[85 13]
 [ 8 62]]
AUC: 0.924


f1 score:  0.8551724137931035
```

5.10 Considerazioni finali e confronto modelli

Le metriche che ho usato per confrontare i modelli testati sono: accuratezza, varianza e deviazione standard generata attraverso il processo di cross-validation, f1-score, la e l'AUC.

MODELLO	ACCURATEZZA	VARIANZA	DEV. STANDARD	F1-SCORE	AVERAGE-PRECISION	AUC
KNN	0.85	0.00007	0.008	0.82	0.85	0.904
RANDOM FOREST	0.83	0.00014	0.011	0.79	0.89	0.914
SVM	0.85	0.00018	0.013	0.81	0.81	0.899
NEURAL NETWORK	0.88	0.00090	0.030	0.855	0.90	0.924

Da questo confronto ho evinto:
• **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 delle classi del modello. La leggermente alta deviazione standard potrebbe indicare alcune variazioni nelle prestazioni tra i fold.
• **Support vector machines (SVM**) e **KNN** presentano entrambe una buona accuratezza leggermente inferiore alle reti neurali e competenza nel classificare i dati. La bassa varianza sottolinea la capacità di generalizzare su diverse partizioni dei dati.
• **Random Forest** ha dimostrato l'accuratezza minore e anche un basso f1-score, questo suggerisce che il modello sta lottando nel trovare un equilibrio tra la riduzione di falsi positivi e falsi negativi.
In conclusione, i dati evidenziano un chiaro ordine di performance tra i modelli considerati. Il Neural Network è risultato il modello più affidabile e preciso, seguito da SVM e KNN. Random Forest rimane a un livello inferiore di accuratezza e prestazioni complessive.

6 Conclusione

In questo progetto, ho voluto esplorare uno dei piu frequenti cancri al cervello da diverse prospettive, utilizzando le tecniche di intelligenza artificiale per analizzare e classificare i dati relativi a questa condizione. Ho raggiunto l'obiettivo prefissato, di sviluppare un utile strumento ausiliare per diagnosticare la possibilità di avere un glioma celebrale attraverso l'uso dell'AI e di classificatori di vari tipi, e non solo con analisi genetiche/molecolari/di laboratorio.