



Dipartimento di Ingegneria Civile, Informatica e delle
Tecnologie Aeronautiche

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Triennale

**TECNICHE DI MACHINE LEARNING PER LA
PREDIZIONE DEL CANCRO MEDIANTE
ANALISI GENETICA**

Laureando

Diego Scirocco

Matricola 558658

Relatore

Prof. Giuseppe Sansonetti

Anno Accademico 2023/2024

A Mamma, Papà e Lollo.

Indice

Introduzione	vii
1 Fondamenti Teorici	1
1.1 Il Cancro come Malattia Genetica	1
1.1.1 La Proliferazione Cellulare Incontrollata	2
1.1.2 Ruolo delle Mutazioni Genetiche	3
1.2 La Medicina Personalizzata nel Trattamento del Cancro	4
1.2.1 Obiettivo: Trattamenti Specifici per il Paziente	4
1.2.2 Importanza della Classificazione delle Mutazioni	4
1.2.3 Vantaggi della Medicina Personalizzata	4
1.3 Machine Learning e Natural Language Processing	5
1.3.1 Machine Learning e Data Science: Fondamenti teorici	5
1.3.2 Le Principali Tecniche di Machine Learning Supervisionato	6
1.3.3 Tecniche di Natural Language Processing (NLP)	7
1.3.4 Problemi di Classificazione e Sbilanciamento delle Classi	8
2 Stato dell'Arte	11
2.1 Tecniche di Machine Learning per la Classificazione delle Mutazioni Genetiche	11

2.2	Tecniche di Natural Language Processing per l'Analisi di Testi Medici	12
2.3	Limiti dei Metodi Tradizionali di Classificazione	13
3	Analisi dei Dati	15
3.1	Il Dataset MSKCC	15
3.1.1	Provenienza e Struttura	15
3.1.2	Tipologie di Dati: Mutazioni Genetiche e Testi Medici	16
3.1.3	Le Nove Classi di Mutazione	16
3.2	Analisi Esplorativa dei Dati (EDA)	19
3.2.1	Distribuzione delle Classi	19
3.2.2	Analisi delle Feature	20
3.2.3	Verifica e gestione dei valori mancanti	24
3.2.4	Il Problema dello Sbilanciamento	26
4	Metodologia di Ricerca	28
4.1	Ambiente di Sviluppo	28
4.1.1	Linguaggio di Programmazione	28
4.1.2	Ambiente di Programmazione	28
4.1.3	Librerie Utilizzate	29
4.2	Approccio e Struttura del Lavoro	31
4.3	Preprocessing del Testo	32
4.3.1	Pulizia e Normalizzazione del Testo	32
4.3.2	Rimozione di Stopword e Punteggiatura	32
4.3.3	Lemmatizzazione e Tokenizzazione	33
4.4	Feature Engineering	34

4.4.1	Estrazione delle Caratteristiche con TF-IDF	34
4.4.2	One-Hot Encoding delle Caratteristiche Categoriali	35
4.4.3	Riduzione della Dimensionalità con SVD	36
4.4.4	Normalizzazione dei Dati	38
4.5	Divisione e Bilanciamento del Dataset	39
4.5.1	Dataset Originale	39
4.5.2	Dataset Bilanciato	40
4.6	Modellazione, Addestramento e Valutazione dei Modelli	42
4.6.1	Confronto tra i Modelli di Classificazione	43
4.6.2	Ottimizzazione degli Iperparametri	44
4.6.3	Valutazione dei Modelli	48
5	Valutazione Sperimentale	49
5.1	Presentazione dei Risultati	49
5.1.1	Risultati sul Dataset Originale	49
5.1.2	Risultati sul Dataset Bilanciato	51
5.1.3	Confronto tra Dataset Originale e Bilanciato	52
6	Sviluppi Futuri	57
6.1	Possibili Sviluppi Futuri	57
6.1.1	Integrazione di Dati più Recenti	57
6.1.2	Sviluppo di Modelli più Avanzati	57
6.1.3	Applicazione in Altri Ambiti Oncologici	58
6.2	Limitazioni dello Studio	58
6.2.1	Disponibilità dei Dati: Dataset Limitato al 2017	58

6.2.2 Necessità di Ulteriore Validazione Clinica	58
Elenco delle figure	60
Conclusioni	62
Bibliografia	63

Introduzione

Il cancro rappresenta una delle più grandi sfide mediche e scientifiche del nostro tempo, coniugando la complessità biologica della malattia alla necessità di sviluppare trattamenti sempre più efficaci e personalizzati. Negli ultimi decenni, i progressi nelle tecnologie di sequenziamento genetico e nell'elaborazione dei dati hanno aperto nuove opportunità per comprendere meglio le basi genetiche di questa patologia e per identificare strategie terapeutiche più mirate. In questo contesto, il **Machine Learning (ML)** e il **Natural Language Processing (NLP)** si sono affermati come strumenti fondamentali per affrontare la complessità dei dati genetici e clinici. La capacità di questi approcci di analizzare grandi quantità di dati eterogenei, identificare pattern nascosti e costruire modelli predittivi accurati li rende particolarmente adatti a supportare la ricerca in ambito oncologico.

La presente tesi si inserisce in questo ambito di ricerca, con l'obiettivo di esplorare l'applicazione di tecniche avanzate di ML e NLP per la classificazione delle mutazioni genetiche legate al cancro. Questo progetto si propone non solo di sviluppare modelli in grado di affrontare un problema così complesso, ma anche di contribuire, attraverso l'analisi e la modellazione, alla comprensione delle basi genetiche della malattia. L'approccio utilizzato unisce un metodo rigoroso con un'attenzione particolare alle applicazioni pratiche, combinando fra loro diverse tecniche per analizzare ed elaborare i dati.

Struttura del documento: Il presente elaborato di tesi è organizzato in capitoli che documentano i fondamenti teorici, lo Stato dell'Arte, la metodologia adottata, i risultati ottenuti e le prospettive future. Questo percorso riflette l'approccio seguito nel progetto, dalla formulazione del problema alla valutazione dei risultati, con l'obiettivo di fornire una visione completa e coerente del lavoro svolto.

Capitolo 1

Fondamenti Teorici

1.1 Il Cancro come Malattia Genetica

Il cancro è una patologia che si sviluppa principalmente a causa di alterazioni genetiche, ovvero mutazioni nel DNA che interferiscono con i normali processi di crescita e divisione cellulare. Queste mutazioni possono essere ereditarie o acquisite in seguito a esposizione a fattori ambientali come sostanze chimiche, radiazioni o infezioni virali. La loro presenza può portare alla trasformazione di cellule sane in cellule tumorali che proliferano in modo incontrollato. Comprendere il cancro come malattia genetica è fondamentale per sviluppare approcci terapeutici specifici basati sul profilo genetico dei tumori.

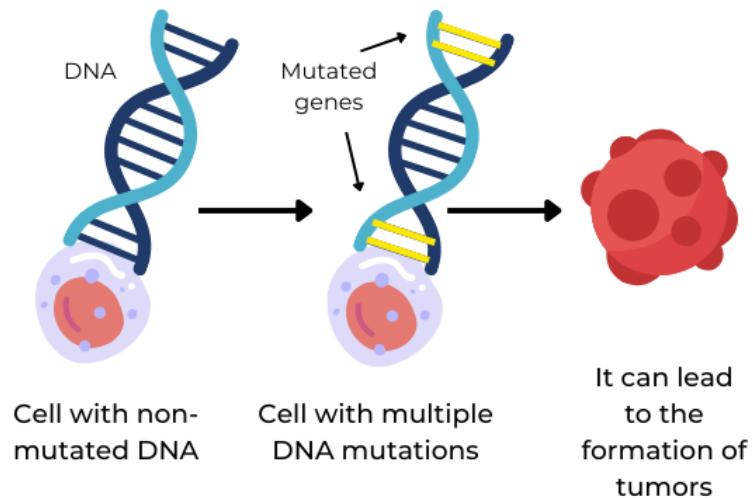


Figura 1.1: Mutazioni del DNA[1]

1.1.1 La Proliferazione Cellulare Incontrollata

La proliferazione cellulare incontrollata è una delle caratteristiche distintive del cancro. Le cellule normali seguono un ciclo di crescita regolato da segnali molecolari che promuovono o inibiscono la divisione e la morte cellulare programmata (apoptosi). Nelle cellule tumorali, invece, le mutazioni genetiche portano alla disfunzione di questi meccanismi, provocando una crescita autonoma e inarrestabile. Questo processo permette alle cellule cancerose di formare masse tumorali e, in alcuni casi, di invadere i tessuti circostanti e metastatizzare in altre parti del corpo.

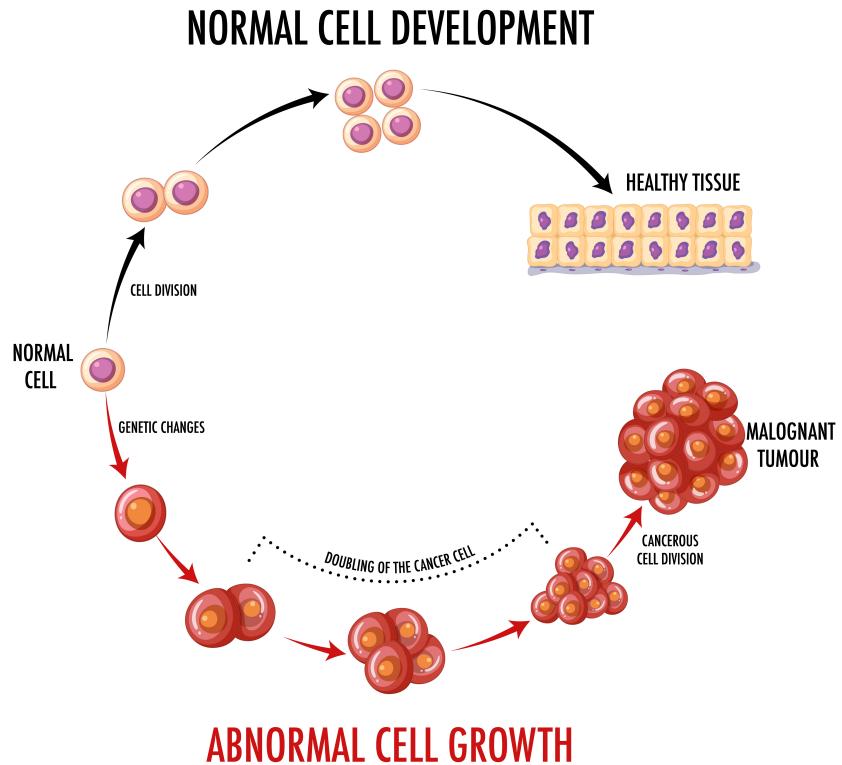


Figura 1.2: Proliferazione Cellulare Incontrollata [2]

1.1.2 Ruolo delle Mutazioni Genetiche

Le mutazioni genetiche nel cancro sono comunemente distinte in due categorie: mutazioni **driver** e mutazioni **passenger**. Le mutazioni driver promuovono attivamente la crescita e la sopravvivenza delle cellule tumorali, mentre le mutazioni passenger, pur essendo presenti, non hanno effetti diretti sullo sviluppo del tumore. Le mutazioni driver spesso riguardano geni che controllano la divisione cellulare, la riparazione del DNA e l'apoptosi. La comprensione di quali mutazioni siano rilevanti aiuta a sviluppare farmaci mirati che possano inibire le proteine prodotte da geni mutati. [3]

1.2 La Medicina Personalizzata nel Trattamento del Cancro

La medicina personalizzata in ambito oncologico si basa sull'analisi delle caratteristiche genetiche specifiche di un tumore per adattare i trattamenti alle diverse casistiche di ogni paziente. Invece di trattare il cancro con un approccio standardizzato, si utilizzano terapie mirate che agiscono sulle specifiche mutazioni. Questo approccio migliora notevolmente l'efficacia dei trattamenti, riducendo inoltre gli effetti collaterali.

1.2.1 Obiettivo: Trattamenti Specifici per il Paziente

Attraverso tecnologie di sequenziamento, è diventato possibile identificare le caratteristiche molecolari che differenziano un tipo di tumore da un altro. Questo permette di selezionare farmaci che agiscono direttamente sui bersagli molecolari specifici del tumore, aumentando la probabilità di successo terapeutico.

1.2.2 Importanza della Classificazione delle Mutazioni

La classificazione delle mutazioni genetiche è fondamentale per il successo della medicina personalizzata. Quest'ultima permette di individuare mutazioni specifiche che influenzano la risposta al trattamento, distinguendo mutazioni ad alto impatto clinico da quelle di minore rilevanza.

1.2.3 Vantaggi della Medicina Personalizzata

La medicina personalizzata porta numerosi vantaggi, tra cui una maggiore efficacia ed una minore tossicità rispetto alle terapie tradizionali. Questo approccio riduce gli effetti collaterali, poiché i trattamenti sono più mirati e meno invasivi per le cellule sane. Inoltre,

permette di risparmiare risorse attraverso l'uso trattamenti più efficienti e riduce così il rischio di sviluppare resistenza ai farmaci. [4]

1.3 Machine Learning e Natural Language Processing

1.3.1 Machine Learning e Data Science: Fondamenti teorici

Il **Machine Learning (ML)** è una branca dell'intelligenza artificiale che si dedica allo sviluppo di algoritmi capaci di apprendere dai dati e migliorare le loro prestazioni autonomamente. Questa disciplina è strettamente collegata con la **Data Science**, che sfrutta strumenti matematici e statistici per estrarre informazioni e valore dai dati.

Tipologie di apprendimento:

- **Supervisionato:** L'algoritmo apprende da un dataset etichettato, dove ogni esempio è associato a una risposta corretta. Esempi includono la classificazione (es. riconoscimento di immagini) e la regressione (es. previsione di valori numerici).
- **Non supervisionato:** L'obiettivo è identificare strutture o classi nei dati non etichettati, come il *clustering* (es. K-means) e la riduzione dimensionale (es. PCA).
- **Apprendimento per rinforzo:** L'algoritmo interagisce con un ambiente, apprendendo attraverso test ed errori per massimizzare una ricompensa (es. giochi o robotica).

Concetti chiave:

- **Overfitting e Underfitting:** L'*overfitting* avviene quando un modello è troppo complesso e si 'abita' ai dati di training, generalizzando male su nuovi dati.

L'*underfitting*, è causato da modelli troppo semplici per catturare le tendenze nei dati e pattern nascosti.

- **Pipeline del Machine Learning:** Comprende i diversi passaggi tra cui la raccolta dati, la pre-elaborazione, la selezione del modello, la valutazione e il deployment.

Il ML è ormai fondamentale nella Data Science, trovando applicazioni in campi come l'analisi predittiva, la medicina e il riconoscimento vocale.

1.3.2 Le Principali Tecniche di Machine Learning Supervisionato

Nell'ambito del **Machine Learning** supervisionato, la classificazione è uno degli obiettivi principali, e per affrontarla esistono diverse tecniche, ciascuna con i propri punti cardine e le proprie limitazioni. Le tecniche di classificazione supervisionata possono essere suddivise in approcci basati su modelli lineari, alberi decisionali, metodi ensemble e tecniche più complesse come reti neurali. I modelli lineari, come ad esempio la **regressione logistica** e le **Support Vector Machines (SVM)**, sono tra i più utilizzati per la loro semplicità. La regressione logistica è ideale per problemi di classificazione binaria e si basa sull'assunzione che esista una relazione lineare tra le caratteristiche dei dati e le probabilità di appartenenza a una classe. Le SVM sono più adatte a problemi con dati ad alta dimensionalità e complessi, poiché cercano di separare le classi lavorando sul margine tra di esse, e possono anche gestire problemi non lineari. Nonostante la loro efficienza, queste due tecniche soffrono di scarse prestazioni su dataset molto complessi, e richiedono un'attenta scelta dei parametri. Gli **alberi di decisione** sono un altro ramo di modelli molto utilizzati, in grado di rappresentare decisioni in modo chiaro e facilmente interpretabile. Un albero di decisione divide i dati in base a regole decisionali. Purtroppo però, gli alberi di decisione tendono a soffrire di **overfitting** se non configurati correttamente. Per questo motivo, sono

spesso combinati con metodi ensemble, come il **Random Forest** o il **Gradient Boosting**, che uniscono più alberi per migliorare le prestazioni e ridurre il rischio di overfitting. Tecniche come il **Random Forest** costruiscono una 'foresta di alberi' in cui ciascun albero è addestrato su un sottoinsieme casuale dei dati, migliorando iterativamente la robustezza del modello finale. I metodi ensemble, come **Random Forest** e **Gradient Boosting**, risultano quindi migliori rispetto ai singoli modelli. Invece di basarsi su un singolo modello, queste tecniche costruiscono un'insieme di modelli deboli, correggendo gli errori dei modelli precedenti. **Gradient Boosting**, con varianti come **XGBoost** e **LightGBM**, è particolarmente popolare per le sue ottime prestazioni, specialmente su dataset complessi. In generale, la scelta della tecnica di classificazione dipende dalle caratteristiche specifiche del problema e può variare a seconda dei casi. [5] [6]

1.3.3 Tecniche di Natural Language Processing (NLP)

Il **Natural Language Processing (NLP)** è una sottocategoria dell'intelligenza artificiale che si dedica alla comprensione e all'uso del linguaggio umano da parte delle macchine. Il **NLP** unisce metodi di linguistica computazionale e apprendimento automatico per elaborare i dati testuali.

Fasi fondamentali del NLP:

1. Pre-elaborazione del testo:

- *Tokenizzazione*: Suddivisione del testo in parole o frasi.
- *Rimozione di stop-word*: Eliminazione delle parole comuni (es. "il", "e", "di") che non aggiungono valore semantico.
- *Stemming e Lemmatizzazione*: Riduzione delle parole alle loro radici.

2. Rappresentazione dei dati testuali:

- *Bag of Words (BoW)*: Rappresentazione del testo come un vettore di frequenze delle parole.
- *TF-IDF (Term Frequency-Inverse Document Frequency)*: Misurazione dell'importanza di una parola in un documento.

Le diverse applicazioni del **Natural Language Processing** includono: **Classificazione del testo**, per esempio il rilevamento dello spam nelle e-mail; **Riconoscimento di entità nominate (NER)**, che consente l'identificazione di nomi, date e organizzazioni; **Analisi del sentiment**, utile per determinare l'emozione o il tono di un testo; **Traduzione automatica**, ad esempio nei sistemi di traduzione come Google Translate o Deepl. Il NLP è attualmente una delle aree più rilevanti dell'IA a causa della sua vasta gamma di applicazioni. [7]

1.3.4 Problemi di Classificazione e Sbilanciamento delle Classi

La **classificazione** rappresenta uno dei problemi principali nel **Machine Learning** e ha l'obiettivo di assegnare una classe a un dato input basandosi su esempi forniti durante la fase di addestramento, tuttavia una delle sfide più complesse in questo contesto è lo sbilanciamento delle classi, cioè una situazione in cui una delle categorie risulta significativamente meno rappresentata rispetto alle altre. Questo problema si verifica molto spesso in molte applicazioni reali, come nel rilevamento di frodi finanziarie o nella diagnosi medica, dove le istanze delle classi minoritarie possono risultare numericamente trascurabili. Lo sbilanciamento influenza negativamente l'addestramento dei modelli, questo perché gli algoritmi tendono a favorire la classe dominante rispetto alle istanze della classe meno rappresentata. Questo può portare a problemi significativi, specialmente in contesti critici

dove un errore potrebbe comportare conseguenze rilevanti, come ad esempio la mancata diagnosi di una malattia rara. Per risolvere il problema dello sbilanciamento delle classi, una delle strategie più utilizzate è il ribilanciamento dei dati, che può essere implementato attraverso diverse tecniche come:

- **Oversampling:** Creazione di copie sintetiche o aumento del numero di esempi delle classi minoritarie, ad esempio utilizzando metodi come **SMOTE** (*Synthetic Minority Oversampling Technique*).
- **Undersampling:** Riduzione del numero di esempi delle classi dominanti per equilibrare la distribuzione delle classi.

Un altro aspetto fondamentale per gestire lo sbilanciamento è l'utilizzo di metriche corrette per valutare le prestazioni dei modelli, questo perchè le metriche tradizionali come l'**accuracy** possono essere fuorvianti in ambienti sbilanciati. Le metriche principali includono:

- **Precision:** Indica la percentuale di predizioni corrette per la classe positiva rispetto a tutte le predizioni effettuate per quella classe. Si calcola come $\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$, dove TP rappresenta i veri positivi e FP i falsi positivi.
- **Recall:** Misura la capacità del modello di individuare correttamente le istanze positive. Si calcola come $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$, dove FN sono i falsi negativi.
- **F1-score:** È la media armonica tra *precision* e *recall*, utile per bilanciare i due aspetti. Si calcola come $\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$.
- **Accuracy:** Indica la percentuale di predizioni corrette rispetto al totale, ma risulta poco informativa in contesti con classi sbilanciate.

Le applicazioni delle tecniche per gestire lo sbilanciamento delle classi sono molteplici. Ad esempio, nel rilevamento delle frodi, le transazioni fraudolente sono solo una piccola parte del totale, quindi è fondamentale utilizzare metodi appropriati per identificare tali eventi. Nella diagnosi medica, le malattie rare sono un caso comune di sbilanciamento in cui ogni errore di classificazione può avere conseguenze gravi. Infine anche nel campo della sicurezza informatica la rilevazione di attacchi all'interno di grandi volumi di dati presenta problematiche simili. Per ottenere modelli predittivi affidabili e utili, è necessario affrontare efficacemente queste situazioni. [8]

Capitolo 2

Stato dell'Arte

Questo capitolo mira a fornire una panoramica delle principali tecniche e metodi attualmente utilizzati per la classificazione delle mutazioni genetiche e l'analisi dei testi medici, sottolineando i progressi ottenuti e i limiti dei metodi tradizionali.

2.1 Tecniche di Machine Learning per la Classificazione delle Mutazioni Genetiche

La classificazione delle mutazioni genetiche è una delle applicazioni più complesse nel *Machine Learning*, in particolare nell'ambito della ricerca sul cancro e della medicina di precisione. L'obiettivo è assegnare ogni mutazione a una determinata classe, basandosi su informazioni biologiche, molecolari e cliniche. Ultimamente i modelli di *Machine Learning* tradizionali e avanzati hanno trovato ampia applicazione in questo campo. Tecniche come le **Support Vector Machines (SVM)**, **Random Forest (RF)** e **Gradient Boosting Machines (GBM)** sono state molto utilizzate, grazie alla loro capacità di gestire dataset complessi con tante feature. Anche se il rapido sviluppo del **Deep Learning** ha portato a

un cambio di rotta, con l'introduzione di modelli come le *reti neurali profonde* (*Deep Neural Networks*, DNN), che sono risultate particolarmente efficaci nell'elaborazione di grandi quantità di dati genomici. Un ruolo essenziale però è anche svolto anche dalla selezione e dall'ingegnerizzazione delle feature (*feature engineering*). Tecniche avanzate come la riduzione della dimensionalità tramite *Principal Component Analysis* (PCA) o algoritmi basati su autoencoder permettono di ottenere informazioni rilevanti, migliorando le prestazioni dei modelli.

2.2 Tecniche di Natural Language Processing per l'Analisi di Testi Medici

L'analisi dei testi medici è fondamentale per estrarre conoscenze utili da fonti non strutturate come articoli scientifici, report clinici e database di annotazioni genetiche. Il *Natural Language Processing* (**NLP**) ha cambiato questo campo introducendo nuovi metodi per comprendere il linguaggio naturale e ottenere informazioni utili. Tra le tecniche più utilizzate vi sono i modelli di rappresentazione del testo come il **BoW**(*Bag of Words*), il **TF-IDF** (*Term Frequency-Inverse Document Frequency*) e i modelli distribuzionali basati su **word embeddings**, come **Word2Vec** e **GloVe**. Ultimamente modelli basati su *Transformer*, come ad esempio *BERT* (*Bidirectional Encoder Representations from Transformers*), hanno migliorato la capacità di comprendere il contesto dei termini medici. Sistemi che supportano le decisioni cliniche possono utilizzare queste tecniche, come database di annotazioni genomiche (es. ClinVar) o nella categorizzazione automatica di articoli per la ricerca medica. [9] [10]

2.3 Limiti dei Metodi Tradizionali di Classificazione

Prima dell'avvento del **Machine Learning** e del **Natural Language Processing**, i metodi tradizionali di classificazione utilizzavano principalmente regole definite manualmente e modelli statistici convenzionali. Sebbene queste tecniche abbiano rappresentato un passo importante nella storia dell'analisi dei dati, presentano diversi limiti che ne riducono l'efficacia in contesti complessi come quello delle mutazioni genetiche e dell'analisi di testi medici. Uno dei limiti più significativi è rappresentato dalla **rigidità delle regole definite manualmente**. Questi sistemi si affidano ad una conoscenza esperta predefinita per creare regole o soglie decisionali. Ad esempio un metodo tradizionale potrebbe cercare mutazioni specifiche sulla base di liste precompilate di varianti note. Questo approccio risulta inefficace nel rilevare nuovi pattern o anomalie, limitando la capacità del sistema di adattarsi a nuove conoscenze. Inoltre soffrono di una **limitata capacità di gestione di dati ad alta dimensionalità e complessità**. Nel caso delle mutazioni genetiche, l'enorme quantità di dati genomici richiede una capacità di elaborazione che i modelli statistici tradizionali difficilmente riescono a supportare senza un'eccessiva semplificazione. Nello stesso modo nell'analisi di testi medici i metodi tradizionali faticano a estrarre informazioni rilevanti da dati non strutturati come articoli o report clinici. Un altro importante limite è la **suscettibilità al rumore nei dati**. I metodi comuni non sono progettati per gestire dati rumorosi, incompleti o ambigui, che sono invece molto comuni nei contesti medici e biologici. Per esempio nei database di mutazioni genetiche, informazioni errate o non standardizzate possono portare a errori significativi nei risultati. Infine, i metodi tradizionali non offrono soluzioni efficaci per affrontare il problema delle **classi sbilanciate**. Poiché si basano su assunzioni rigide sulle distribuzioni dei dati, tendono a ignorare le classi meno rappresentate, che invece spesso rivestono un'importanza critica. Queste

limitazioni evidenziano la necessità di metodi più automatizzati e flessibili, come quelli offerti dal *Machine Learning* e dal *Natural Language Processing*, in grado di adattarsi ai dati complessi della biomedicina. [11]

Capitolo 3

Analisi dei Dati

Questo capitolo descrive il dataset MSKCC utilizzato in questa ricerca, analizzandone la struttura, le tipologie di dati e le nove classi di mutazione. Viene inoltre presentata l'Analisi Esplorativa dei Dati (EDA) e le tecniche di preprocessing utilizzate.

3.1 Il Dataset MSKCC

Il dataset del **Memorial Sloan Kettering Cancer Center (MSKCC)** è una risorsa essenziale per lo studio del cancro e delle mutazioni genetiche associate. Questo dataset fornisce informazioni dettagliate su **campioni di tumori**, inclusi dati genetici, clinici e testuali, che descrivono le caratteristiche molecolari e le mutazioni rilevate nei pazienti oncologici.

3.1.1 Provenienza e Struttura

Il dataset MSKCC è stato raccolto attraverso studi clinici e genetici condotti dal Memorial Sloan Kettering Cancer Center, uno dei più importanti centri di ricerca oncologica al mondo. La struttura del dataset è organizzata in tabelle che includono sia dati numerici

che testuali, come le informazioni sulle varianti genetiche (mutazioni) e i dettagli clinici sui pazienti. Ogni riga rappresenta un caso clinico con le sue specifiche genetiche e cliniche, facilitando così l'analisi statistica e l'addestramento di modelli predittivi. [11]

3.1.2 Tipologie di Dati: Mutazioni Genetiche e Testi Medici

Il dataset include diversi tipi di dati fondamentali per la ricerca. Le **mutazioni genetiche** sono annotate in termini di cambiamenti specifici nel DNA (es. sostituzioni di basi, delezioni e inserzioni), fornendo informazioni dettagliate sulle alterazioni a livello molecolare. Il dataset contiene anche **testi medici** come rapporti clinici, annotazioni sui trattamenti e risposte ai farmaci, che permettono di interpretare i risultati genetici in relazione alla diagnosi e al decorso clinico del paziente.

3.1.3 Le Nove Classi di Mutazione

Le mutazioni genetiche presenti nel dataset sono classificate in **nove categorie principali**, rappresentano le diverse tipologie di alterazioni genomiche che possono influenzare il comportamento del tumore. Queste categorie includono mutazioni **driver** e **passenger** e ciascuna classe ha un diverso impatto clinico. La classificazione è fondamentale per capire quali mutazioni sono direttamente correlate alla crescita e alla diffusione del cancro, nello specifico sono:

- **Classe 1: Guadagno di funzione**

Si tratta di mutazioni che attivano in modo anomalo una proteina, portando ad una proliferazione delle cellule tumorali. Sono direttamente responsabili dell'insorgenza o della sviluppo del cancro.

- **Classe 2: Probabile Guadagno di funzione**

Si tratta di mutazioni che potrebbero conferire un guadagno di funzione, potrebbero agire come driver ma non vi è certezza.

- **Classe 3: Perdita di funzione**

Si tratta di mutazioni che inibiscono l'attività di geni soppressori tumorali, eliminando meccanismi di controllo della proliferazione cellulare, di conseguenza agiscono come driver.

- **Classe 4: Probabile perdita di funzione**

Si tratta di mutazioni che probabilmente causano una perdita di funzione, ma il loro impatto come driver rimane meno evidente.

- **Classe 5: Neutro**

Si tratta di mutazioni che non influenzano direttamente la crescita o la sopravvivenza cellulare, di conseguenza non contribuiscono alla progressione del cancro. Per questo possono essere classificate come Passenger.

- **Classe 6: Probabilmente neutro**

Si tratta di mutazioni considerate non rilevanti per la funzione cellulare, ma la cui natura potrebbe essere oggetto di ulteriori studi. Solitamente vengono classificate come Passenger.

- **Classe 7: Cambio di funzione**

Si tratta di mutazioni che trasformano la funzione originale del gene in una nuova, contribuendo alla progressione tumorale, di conseguenza sono di tipo Driver.

- **Classe 8: Probabile cambio di funzione**

Si tratta di mutazioni che potrebbero alterare la funzione originale del gene, ma il loro ruolo è ancora da definire. Probabilmente di tipo Driver.

- **Classe 9: Inconcludente**

Si tratta di mutazioni il cui impatto funzionale o ruolo è incerto. Attualmente non vengono classificate in una categoria specifica.

CLASSE	DESCRIZIONE	TIPOLOGIA
1	GUADAGNO DI FUNZIONE	DRIVER
2	PROBABILE GUADAGNO DI FUNZIONE	SOLITAMENTE DRIVER
3	PERDITA DI FUNZIONE	DRIVER
4	PROBABILE PERDITA DI FUNZIONE	SOLITAMENTE DRIVER
5	NEUTRO	PASSENGER
6	PROBABILMENTE NEUTRO	PASSENGER
7	CAMBIO DI FUNZIONE	DRIVER
8	PROBABILE CAMBIO DI FUNZIONE	SOLITAMENTE DRIVER
9	INCONCLUDENTE	NON DEFINITO

Figura 3.1: Le 9 Classi

3.2 Analisi Esplorativa dei Dati (EDA)

L’Analisi Esplorativa dei Dati (EDA) è una fase fondamentale in ogni progetto di data science. Grazie all’EDA è possibile ottenere una comprensione dettagliata del dataset, individuando pattern e tendenze che influiscono sulle scelte e sulle performance dei modelli di Machine Learning. [12] [13]

3.2.1 Distribuzione delle Classi

L’EDA ha rivelato una **distribuzione sbilanciata delle classi** nel set di dati MSKCC.

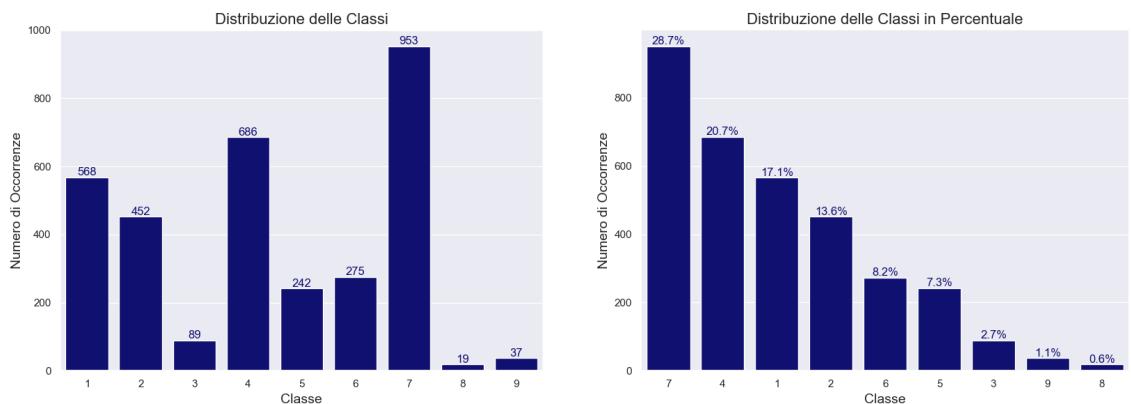


Figura 3.2: Distribuzione delle Classi

La **Figura 3.2** evidenzia in modo chiaro lo sbilanciamento tra le classi, con alcune che presentano una percentuale di campioni notevolmente più alta rispetto ad altre. Ad esempio, la classe **7** rappresenta il **28,7%** dei campioni, mentre la classe **8** solo lo **0,6%**. Tale differenza rende indispensabile l’adozione di tecniche di bilanciamento o di algoritmi in grado di gestire lo sbilanciamento, al fine di ottimizzare le performance del modello.

3.2.2 Analisi delle Feature

L'analisi delle feature è uno step fondamentale in un progetto di Machine Learning perché consente di identificare le **variabili più rilevanti** per la previsione del target(Classe). In questo progetto, l'analisi delle feature si concentra su tre aspetti principali: **geni**, **varianti** e **testo**.

3.2.2.1 Geni

L'analisi delle feature genetiche si concentra sull'identificazione dei **geni unici** e dei **più frequenti** per ciascuna classe. Questo tipo di analisi è utile per esplorare eventuali correlazioni tra specifici geni e le classi di varianti. Il dataset analizzato contiene **262 geni unici**. Tra i geni più frequenti, si trovano BRCA1, TP53, EGFR, PTEN e BRCA2.

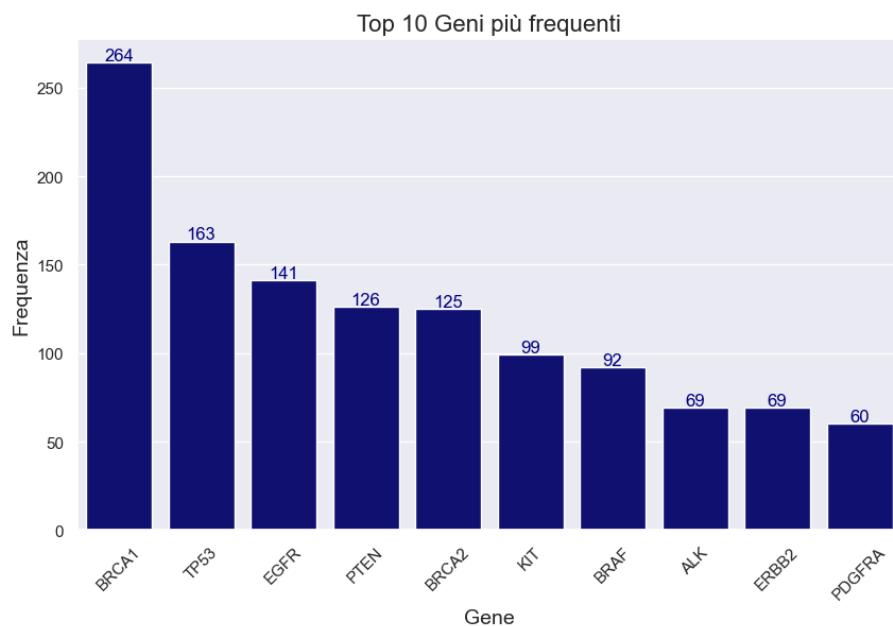


Figura 3.3: 10 Geni più frequenti nel Dataset

È stato altresì utile analizzare la distribuzione percentuale dei geni per **ciascuna classe** target del dataset, al fine di comprendere meglio le relazioni esistenti tra di essi.

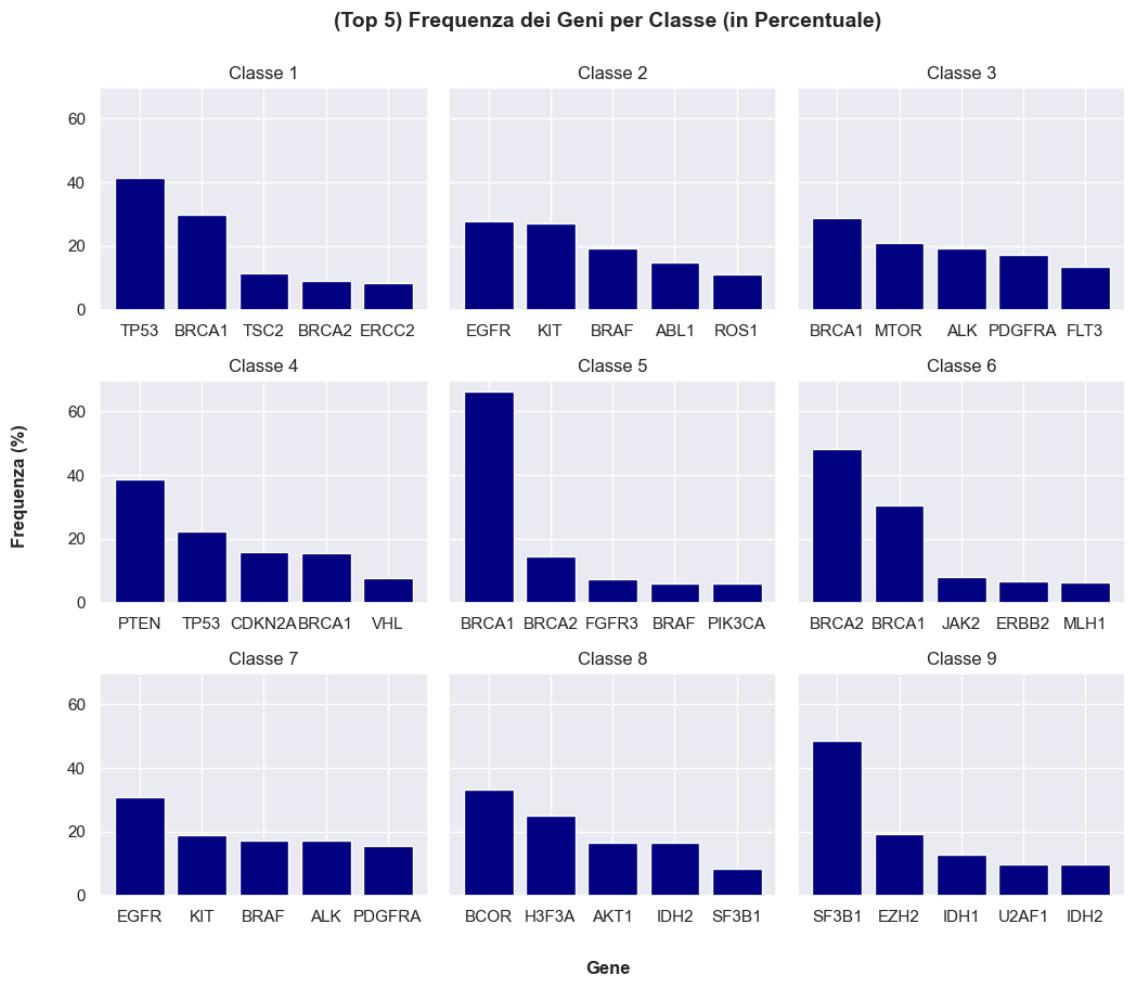


Figura 3.4: Distribuzione Percentuale dei Geni per Classe

3.2.2.2 Varianti

L'analisi delle varianti segue un approccio simile a quello utilizzato per i geni, con l'obiettivo di identificare le **varianti uniche** e quelle **più comuni** per ciascuna classe. Questa analisi è necessaria per individuare eventuali pattern caratteristici che possano essere utili per la

costruzione di feature discriminanti. L'analisi ha identificato un totale di **2993 varianti uniche**. Tra queste, le dieci varianti più frequenti sono: 'Truncating Mutations', 'Deletion', 'Amplification', 'Fusions', 'Overexpression', 'G12V', 'Q61L', 'E17K', 'T58I' e 'Q61R'.

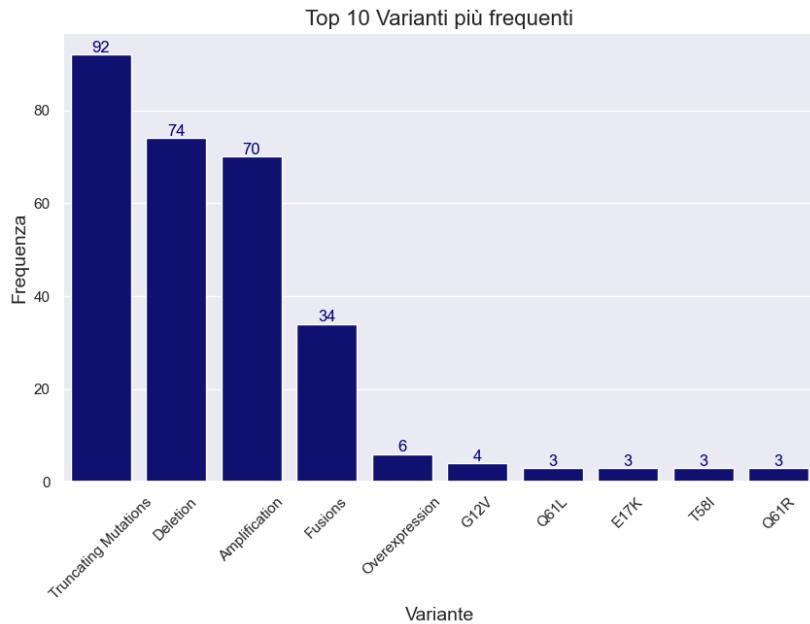


Figura 3.5: 10 Varianti più frequenti nel Dataset

3.2.2.3 Testo

Un'analisi analoga è stata effettuata anche per la feature **testo**, fondamentale per la classificazione in questo contesto. Per rendere il testo compatibile con i modelli di Machine Learning è stato necessario applicare una serie di tecniche di pre-elaborazione. Questi passaggi hanno trasformato il testo grezzo in un formato strutturato, in modo che potesse essere utilizzato dagli algoritmi di classificazione. Questa analisi ha permesso di ottenere informazioni sulla **distribuzione della lunghezza** del testo per le diverse mutazioni genetiche.

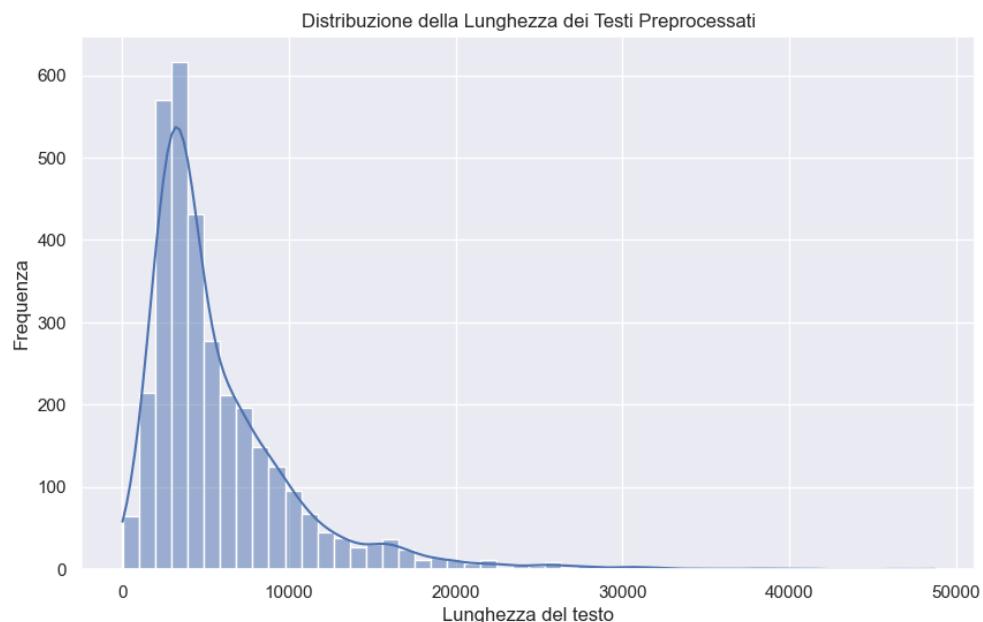


Figura 3.6: Distribuzione della Lunghezza dei Testi Preprocessati

Come mostrato nella **Figura 3.6**, i testi associati alle mutazioni genetiche risultano essere generalmente molto complessi e lunghi. La media delle parole per riga è infatti di **5836**, con alcune mutazioni che raggiungono anche le **48707** parole nella loro descrizione. La lunghezza e la complessità dei testi pongono problemi significativi per i modelli di Machine Learning, i quali necessitano di un'**elaborazione accurata** per gestire efficacemente queste grandi quantità di informazioni.



Figura 3.7: WordCloud dei Termini più Frequenti

La **Figura 3.7** mostra una rappresentazione visiva delle parole più frequenti nei testi attraverso una **WordCloud**. In questa rappresentazione la dimensione delle parole è proporzionale alla loro frequenza ovvero le parole più utilizzate appaiono più grandi e in evidenza, mentre quelle meno frequenti sono di dimensioni più ridotte. Questo tipo di analisi mostra in modo immediato le parole chiave e i termini più ricorrenti nei dati testuali. La **WordCloud** è un utile strumento esplorativo utilizzato per evidenziare la presenza di termini tecnici che potrebbero essere particolarmente significativi per il problema in esame.

3.2.3 Verifica e gestione dei valori mancanti

La presenza di **valori mancanti** nei dataset rappresenta un problema molto comune nell'ambito dell'analisi dei dati e del **Machine Learning**. Se i valori mancanti non vengono gestiti adeguatamente possono influire negativamente sull'accuratezza e l'affidabilità dei

modelli predittivi, questo perchè introducono incertezza e possono alterare le relazioni tra le variabili. La corretta gestione dei valori mancanti è quindi uno step cruciale nell'elaborazione preliminare dei dati. Nel nostro caso specifico, l'analisi esplorativa ha evidenziato che solo **5 record** presentano valori mancanti, tutti relativi alla **feature testuale**. Questo dato dimostra che il dataset nel complesso è completo e di buona qualità. E' importante notare però che anche un numero limitato di valori mancanti deve essere trattato con attenzione per evitare di introdurre distorsioni nei modelli o complicazioni nelle fasi successive.

[14]

Le strategie principali per affrontare i valori mancanti sono:

- **Imputazione dei valori mancanti:** Sostituzione dei valori mancanti con stime come la media, la mediana o il valore più frequente della variabile.
- **Eliminazione dei record incompleti:** Rimozione delle righe (o colonne) contenenti i valori mancanti. È consigliata solo quando la quantità di dati eliminati è marginale rispetto alla dimensione complessiva del dataset, per evitare una perdita eccessiva di informazioni.

Resoconto dei Valori Mancanti per Feature		Righe con Valori Mancanti:			
Feature	Valori Mancanti	ID	Gene	Variation	Class
ID	0	1109	FANCA	S1088F	1
Gene	0	1277	ARID5B	Truncating Mutations	1
Variation	0	1407	FGFR3	K508M	6
Class	0	1639	FLT1	Amplification	6
text	5	2755	BRAF	G596C	7

Figura 3.8: Valori Mancanti

Nel nostro caso, la presenza di soli 5 record incompleti rispetto alla dimensione totale del dataset ha giustificato l'adozione della seconda strategia ovvero l'**eliminazione** dei record

contenenti i valori mancanti. Come mostrato nella **Figura 3.8**, le righe che presentano valori nulli appartengono a classi ampiamente rappresentate, come le classi **1, 6 e 7**. Ciò suggerisce che la rimozione di questi record **non compromette** in modo grave la distribuzione delle classi nel dataset. La decisione di eliminare questi record è stata presa dalla semplicità e dal potenziale impatto trascurabile che questa rimozione avrebbe avuto sull'integrità e la rappresentatività del dataset. Dopo l'eliminazione dei record il dataset ha subito un leggero cambiamento come mostrato nella **Figura 3.9**.

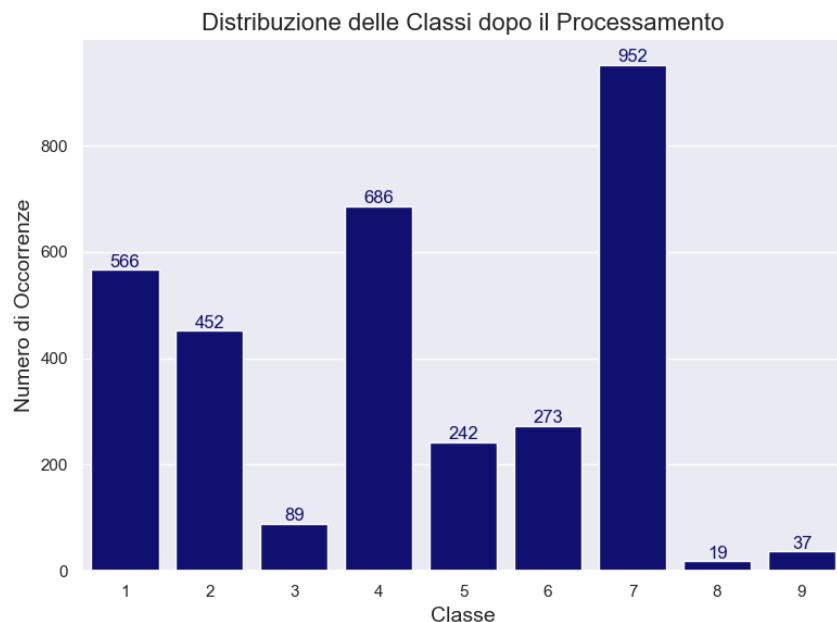


Figura 3.9: Distribuzione delle Classi Aggiornata

3.2.4 Il Problema dello Sbilanciamento

Dall'analisi esplorativa dei dati è emerso un notevole sbilanciamento tra le classi del dataset. In particolare, classi come la **3, 8 e 9** presentano un numero **significativamente inferiore** di campioni rispetto a classi come la **1, 4 e 7**, che sono **ampiamente rappresentate**. Questo sbilanciamento, se non adeguatamente gestito, può influenzare negativamente le

prestazioni dei modelli di classificazione, portando a una scarsa capacità di generalizzazione per le classi meno rappresentate. Per affrontare questo problema, è stata presa in considerazione l'adozione di diverse strategie di bilanciamento del dataset. In particolare, si è deciso di addestrare i modelli sia sul **dataset originale** che su un **dataset bilanciato**, applicando tecniche di bilanciamento dei dati. Tra queste tecniche, si è scelto di utilizzare **SMOTE** (Synthetic Minority Over-sampling Technique), che genera nuovi campioni sintetici per le classi minoritarie. L'obiettivo è stato quello di confrontare i risultati ottenuti sui due dataset, al fine di determinare se l'uso di un dataset bilanciato potesse migliorare le prestazioni del modello, in particolare per quanto riguarda la capacità di classificare correttamente le classi meno rappresentate. [15]

Capitolo 4

Metodologia di Ricerca

4.1 Ambiente di Sviluppo

4.1.1 Linguaggio di Programmazione

Il linguaggio di programmazione utilizzato per questo progetto è **Python**, uno dei linguaggi più popolari nell'ambito della scienza dei dati. Python è famoso per la sua sintassi semplice ed intuitiva, che rende facile l'approccio alla programmazione anche per i non esperti. La vasta disponibilità di librerie e framework specifici per l'elaborazione dei dati, l'apprendimento automatico e l'analisi statistica fa sì che Python sia un'ottima scelta per progetti di ml.

4.1.2 Ambiente di Programmazione

Per lo sviluppo e l'esecuzione del codice è stato scelto l'ambiente **Jupyter Notebook**. Questo ambiente offre una piattaforma interattiva che consente di scrivere ed eseguire il codice in modo semplice. Jupyter Notebook è particolarmente adatto per lavori di data analysis e Machine Learning perché permette di visualizzare i risultati in tempo reale accanto

al codice facilitando così l'esplorazione dei dati e il debug. Supporta l'esecuzione di codice Python e offre una semplice interfaccia per integrare grafici, tabelle e altre visualizzazioni direttamente nel documento. Jupyter è anche molto utilizzato nella comunità scientifica grazie alla compatibilità con diversi linguaggi di programmazione. Permette di condividere facilmente i notebook con altri ricercatori o colleghi consentendo una collaborazione efficace.

4.1.3 Librerie Utilizzate

Per realizzare l'intero flusso di lavoro, sono state utilizzate **diverse librerie Python** ognuna delle quali ha svolto un ruolo fondamentale in diverse fasi del progetto. Le librerie impiegate sono state suddivise in diverse categorie a seconda delle loro funzionalità:

- **Calcolo e Manipolazione Dati:**

- NumPy e Pandas per la gestione e manipolazione dei dati.
- Counter, Re e String per il conteggio degli elementi e la manipolazione di stringhe.

- **NLP e Preprocessing:**

- nltk per la tokenizzazione, la rimozione delle stopwords e lo stemming.
- TfidfVectorizer per la vettorizzazione del testo.

- **Gestione degli Squilibri di Classe:**

- SMOTE per bilanciare il dataset, trattando il problema dello squilibrio tra le classi.

- **Machine Learning e Modelli:**

- `Scikit-learn` per i modelli di classificazione, tra cui `Random Forest`, `XGBoost` e altre tecniche di valutazione come `ConfusionMatrix` e `classification_report`.

- **Visualizzazione:**

- `Matplotlib`, `Seaborn`, e `Plotly` per la creazione di grafici e la visualizzazione interattiva dei dati.

Queste librerie sono state essenziali per il preprocessing dei dati, la costruzione e la valutazione dei modelli, nonché per la visualizzazione dei risultati. [16] [17]

4.2 Approccio e Struttura del Lavoro

Questo capitolo descrive la struttura del lavoro svolto fornendo una visione chiara delle diverse fasi e del flusso delle attività. L'obiettivo è mostrare i passaggi fondamentali che hanno permesso lo sviluppo del progetto. Di seguito è riportato uno schema che illustra il **workflow** del progetto, il quale evidenzia le principali fasi e i collegamenti tra di esse.

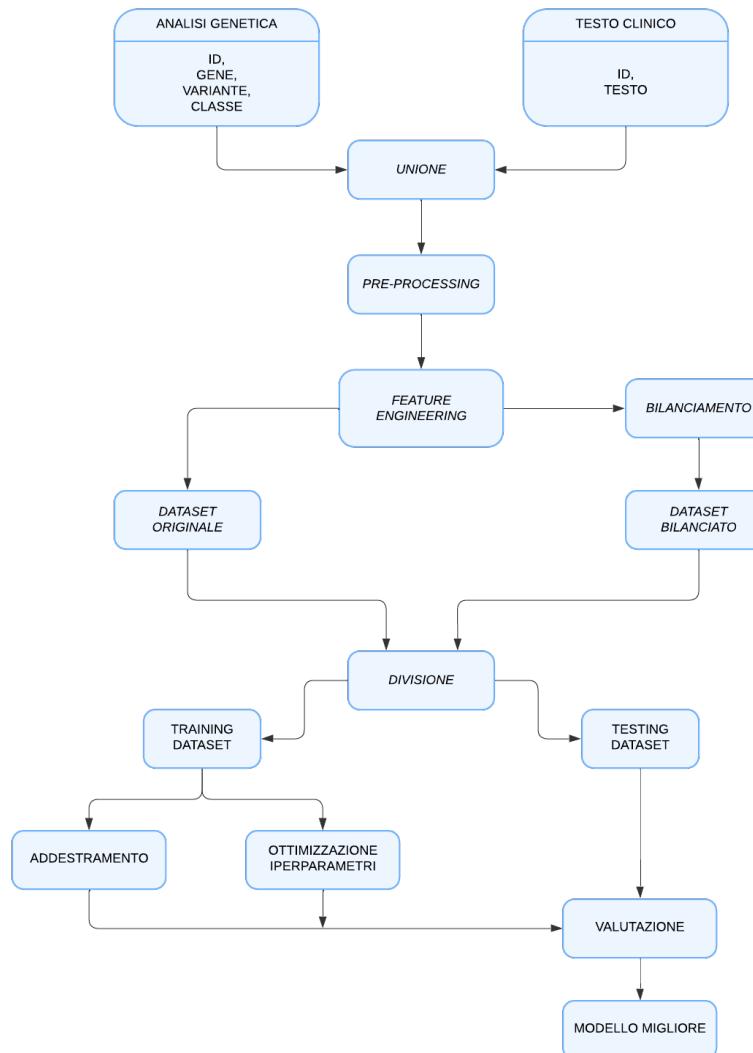


Figura 4.1: Modello del Workflow: Approccio e Metodologia

4.3 Preprocessing del Testo

4.3.1 Pulizia e Normalizzazione del Testo

La pulizia del testo comprende la rimozione di **elementi superflui** o **incoerenti** che possono interferire con l'analisi. Nel progetto, la funzione `pre_process` esegue i seguenti passaggi:

- **Conversione in minuscolo:** Tutto il testo viene convertito in minuscolo per garantire uniformità e evitare che parole identiche ma con diversa capitalizzazione vengano trattate come distinte.
- **Rimozione di punteggiatura, HTML e caratteri speciali:** Vengono eliminati tutti i caratteri di punteggiatura, i tag HTML e i caratteri speciali, lasciando solo testo alfanumerico e spazi.
- **Rimozione di spazi multipli:** Eventuali sequenze di spazi multipli vengono ridotte a un singolo spazio, normalizzando la spaziatura tra le parole.

Queste operazioni garantiscono che il testo sia **omogeneo** e privo di elementi che non contribuiscono al significato semantico.

4.3.2 Rimozione di Stopword e Punteggiatura

Le **stopword** sono parole comuni che hanno **scarso valore** informativo, come articoli, preposizioni e congiunzioni. La loro rimozione aiuta a **ridurre il rumore** e a concentrare l'analisi sulle parole più significative.

- **Definizione delle Stopword:** Il progetto utilizza le stopword predefinite della libreria NLTK per la lingua inglese, a cui vengono aggiunte manualmente parole specifiche del contesto come "fig", "figure", "et", "al", "also", "data", etc.
- **Rimozione delle Stopword:** Durante la tokenizzazione, le parole vengono confrontate con l'insieme delle stop words e rimosse se presenti.

4.3.3 Lemmatizzazione e Tokenizzazione

Lo **Lemmatizzazione** riduce le parole alla loro radice. La **tokenizzazione** divide il testo in singole parole o unità significative chiamate **token**.

- **Lemmatizzazione:** Il progetto utilizza il *WordNetLemmatizer*, un algoritmo che rimuove i suffissi comuni dalle parole inglesi.
- **Tokenizzazione:** La funzione *word_tokenize* di NLTK divide il testo in token, separando le parole in base agli spazi e alla punteggiatura.

Lemmatizzazione e tokenizzazione vengono applicati in sequenza: il testo viene prima tokenizzato, poi ogni token viene lemmatizzato. La lemmatizzazione riduce la dimensionalità del vocabolario e raggruppa parole con significato simile, mentre la tokenizzazione rende il testo processabile dagli algoritmi di Machine Learning. [7]

```

# Inserimento di altre stop words manualmente adatte al contesto
custom_words = ["fig", "figure", "et", "al", "al.", "also", "data", "analyze", "study",
                "table", "using", "method", "result", "conclusion", "author", "find", "found", "show"]

# Unione delle stop words di default + la punteggiatura + quelle aggiunte manualmente
stop_words = set(stopwords.words('english')) + list(string.punctuation) + custom_words

wordnet_lemmatizer = WordNetLemmatizer()

def pre_process(text):
    if isinstance(text, float):
        return '' # Gestione di valori float

    text = str(text).lower().strip()

    # Rimozione di punteggiatura, HTML e caratteri speciali
    text = re.sub(f'[{re.escape(string.punctuation)}]', ' ', text)
    text = re.sub(r'<.*?>+', ' ', text)
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)
    text = re.sub(r'\s+', ' ', text)

    # Tokenizzazione, rimozione stopwords e lemmatizzazione
    tokens = word_tokenize(text)
    filtered_tokens = [
        wordnet_lemmatizer.lemmatize(word) for word in tokens
        if word not in stop_words and not word.isdigit() and len(word) > 1 # ignora parole di lunghezza 1
    ]
    return ' '.join(filtered_tokens)

```

Figura 4.2: Preprocessing del Testo

4.4 Feature Engineering

La **Feature Engineering** è una fase cruciale nel processo di costruzione di un modello di Machine Learning, in quanto determina come le informazioni presenti nel dataset vengono trasformate in input utili per l'algoritmo. Nel nostro caso, abbiamo utilizzato una combinazione di tecniche per **estrarre**, **ridurre** e **normalizzare** le caratteristiche del dataset al fine di ottenere un modello di classificazione robusto e performante.

4.4.1 Estrazione delle Caratteristiche con TF-IDF

L'**estrazione delle caratteristiche** da un dataset che contiene informazioni testuali è una fase fondamentale, poiché i modelli di Machine Learning non possono lavorare direttamente con testo non strutturato. In questo caso, abbiamo utilizzato la tecnica **TF-IDF (Term Frequency - Inverse Document Frequency)** per trasformare il testo medico nella

colonna **text** in una rappresentazione numerica. TF-IDF è una tecnica di **vettorizzazione** che combina due concetti:

- **Term Frequency (TF)**: misura la frequenza di una parola all'interno di un documento. Più alta è la frequenza, maggiore sarà l'importanza della parola nel documento.
- **Inverse Document Frequency (IDF)**: misura l'importanza di una parola nell'intero corpus. Le parole che appaiono frequentemente in molti documenti avranno un punteggio IDF basso, mentre le parole che compaiono solo in pochi documenti avranno un punteggio IDF più alto.

L'algoritmo calcola un punteggio per ciascuna parola in base a queste due misure e crea una matrice in cui ogni colonna rappresenta una parola e ogni riga rappresenta un documento (nel nostro caso, una mutazione genetica descritta nel testo). Questo consente al modello di classificazione di trattare il testo come un **input numerico** e utilizzarlo per fare previsioni.

[18] [19]

4.4.2 One-Hot Encoding delle Caratteristiche Categoriali

Nel nostro dataset, le variabili **Gene** e **Variation** sono variabili categoriali, cioè contengono etichette che non hanno un ordine naturale. Per permettere al nostro modello di Machine Learning di trattarle come input numerici, abbiamo applicato la tecnica di **One-Hot Encoding**. One-Hot Encoding trasforma ogni categoria in una nuova variabile binaria, che assume il valore di 1 se la categoria è presente e 0 altrimenti. Questo approccio è utile per rappresentare variabili categoriali senza introdurre un ordine artificiale tra le categorie. **Passaggi per l'One-Hot Encoding**:

1. Per ogni variabile categoriale (in questo caso, **Gene** e **Variation**), è stato creato un nuovo set di colonne, una per ogni categoria presente.
2. Le colonne ottenute sono state aggiunte al dataset e le variabili originali sono state rimosse.
3. Il risultato è stato una rappresentazione numerica che permette al modello di lavorare con variabili categoriali.

Vantaggi di One-Hot Encoding:

- Non introduce alcun ordine tra le categorie.
- Ogni categoria è rappresentata come una variabile indipendente, evitando che l'algoritmo attribuisca significato a un ordine inesistente.

[16]

```
# One Hot Encoding per Gene e Variation
one_hot_encoder = OneHotEncoder()

# Applica One Hot Encoding su Gene e Variation
X_gene_variation = one_hot_encoder.fit_transform(X[['Gene', 'Variation']]).toarray()
```

Figura 4.3: One Hot Encoding per le Feature Categoriali

4.4.3 Riduzione della Dimensionalità con SVD

Dopo aver applicato TF-IDF, il numero di caratteristiche (parole) nel nostro dataset può diventare molto elevato, portando a una **alta dimensionalità**. Questo può causare due problemi principali:

- **Overfitting:** un numero elevato di caratteristiche può rendere il modello più complesso e incline a "memorizzare" il dataset di addestramento invece di generalizzare bene.
- **Computational Cost:** un numero elevato di caratteristiche può rallentare notevolmente l'addestramento e l'inferenza del modello.

Per ridurre la dimensionalità, abbiamo utilizzato **SVD (Singular Value Decomposition)**, una tecnica di **decomposizione delle matrici** che permette di ridurre il numero di variabili, mantenendo la maggior parte dell'informazione contenuta nei dati. SVD riduce il numero di dimensioni della matrice TF-IDF in modo che vengano conservate solo le componenti principali, eliminando il rumore e le dimensioni meno rilevanti. Il numero di componenti da mantenere è scelto in base a una valutazione empirica che bilancia la perdita di informazione con l'efficienza computazionale. **Vantaggi della riduzione della dimensionalità con SVD:**

- **Miglioramento delle prestazioni:** ridurre il numero di caratteristiche aiuta a evitare l'overfitting e aumenta la capacità di generalizzazione del modello.
- **Miglioramento della velocità di addestramento:** riducendo la dimensionalità, il tempo necessario per addestrare il modello è significativamente ridotto.

Nel nostro caso, abbiamo ridotto la matrice TF-IDF da **5000 caratteristiche** (parole) a un numero più contenuto di **300 caratteristiche**. Questo ci ha permesso di mantenere le informazioni più rilevanti senza compromettere troppo le prestazioni del modello.

4.4.4 Normalizzazione dei Dati

La **normalizzazione** dei dati è un passo importante quando si combinano caratteristiche che possono avere scale molto diverse. Nel nostro dataset, abbiamo utilizzato diverse tipologie di dati: alcune sono numeriche (come i codici genetici) e altre sono vettori di parole generati da TF-IDF, che sono molto più grandi e hanno una scala diversa. Per evitare che alcune caratteristiche dominino sull'addestramento del modello, abbiamo applicato una **normalizzazione** a tutte le caratteristiche numeriche. Abbiamo utilizzato il **StandardScaler** di scikit-learn per ridurre la varianza delle caratteristiche numeriche, portando tutte le variabili sulla stessa scala con una **media zero** e una **deviazione standard unitaria**.

In particolare:

- Le colonne contenenti i codici per i **geni** e le **variazioni** (che sono variabili categoriali) sono state trasformate in numeriche tramite **One-Hot Encoding**.
- Le caratteristiche derivate dal **TF-IDF** (testo) sono state normalizzate separatamente.

La normalizzazione permette al modello di imparare più velocemente e in modo più stabile, poiché tutte le caratteristiche sono trattate in modo uniforme.

```
# TF-IDF
tfidf = TfidfVectorizer(max_features=5000) # Limita il numero di feature
X_text_tfidf = tfidf.fit_transform(X['text'])

# SVD
svd = TruncatedSVD(n_components=300, random_state=42) # Riduci a 300 dimensioni
X_text_svd = svd.fit_transform(X_text_tfidf)

#SCALER
scaler = StandardScaler()
X_text_scaled = scaler.fit_transform(X_text_svd)
```

Figura 4.4: TF-IDF, SVD e Normalizzazione

4.5 Divisione e Bilanciamento del Dataset

Un altro passo fondamentale nel processo di addestramento di un modello di Machine Learning è la **suddivisione** del dataset in set di **training** e **test**. Questa divisione consente di addestrare il modello, ottimizzare gli iperparametri e valutarne le prestazioni. Nel nostro caso, è stato necessario confrontare le prestazioni del modello sia sul dataset **originale** che su una versione **bilanciata**. Questo confronto è stato essenziale per trarre conclusioni corrette, poiché il bilanciamento delle classi può influire significativamente sui risultati.

4.5.1 Dataset Originale

Per quanto riguarda il dataset **originale**, la suddivisione è stata eseguita utilizzando la funzione **train_test_split**, destinando l'**80%** dei dati al **training set** e il **20%** al **test set**. È stato importante mantenere la distribuzione delle classi nel **test set** tramite lo **stratified split** per garantire che la distribuzione delle classi fosse rappresentata in modo simile tra i due set. Successivamente, l'addestramento del modello e la valutazione delle sue prestazioni sono stati eseguiti su questa struttura, senza alcun intervento sul bilanciamento delle classi.

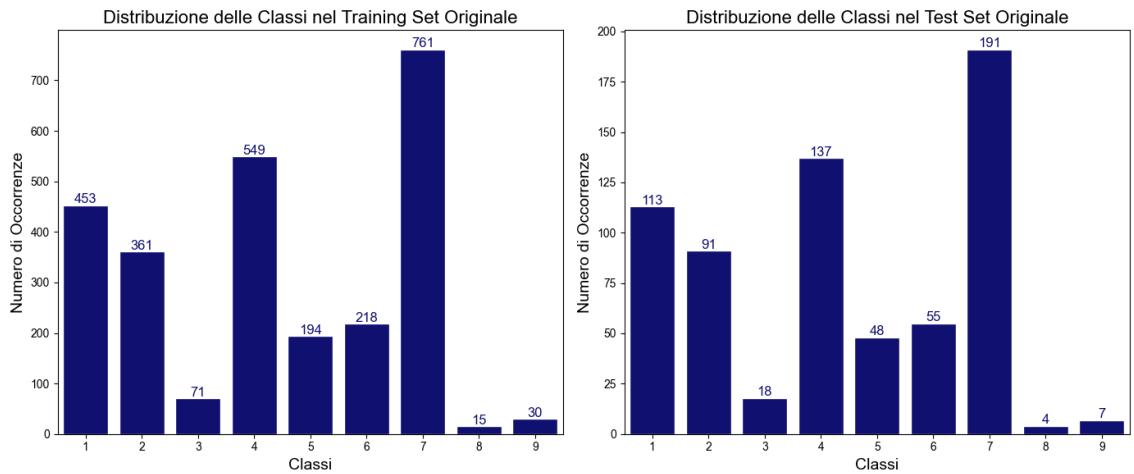


Figura 4.5: Divisione Dataset Originale

4.5.2 Dataset Bilanciato

Nel caso del **dataset bilanciato**, l’obiettivo principale è stato quello di affrontare lo **sbilanciamento** delle classi, in particolare per le classi **3, 8, e 9**, che avevano un numero di campioni significativamente inferiore rispetto alle altre. Per evitare che le classi maggioritarie dominassero la fase di addestramento e influenzassero negativamente la valutazione del modello, è stato applicato un metodo di bilanciamento tramite **SMOTE (Synthetic Minority Over-sampling Technique)**. Successivamente, la divisione del dataset in **train** e **test** è stata effettuata mantenendo il rapporto **80:20**, come nel caso del dataset originale. E’ importante sottolineare che l’uso di **SMOTE** prima della divisione in set di **training** e **test** non è una pratica generalmente raccomandata. Questo approccio può portare a un **data leakage**, poiché la creazione di campioni sintetici potrebbe influenzare i dati di test, compromettendo la valutazione del modello. Nonostante ciò, si è scelto di applicare questa tecnica in modo controllato, considerando che il dataset originale era particolarmente sbilanciato e conteneva un numero molto ridotto di campioni per alcune classi. Pertanto, questa decisione è stata presa per cercare di migliorare le prestazioni complessive del

modello, pur mantenendo la validità delle conclusioni.

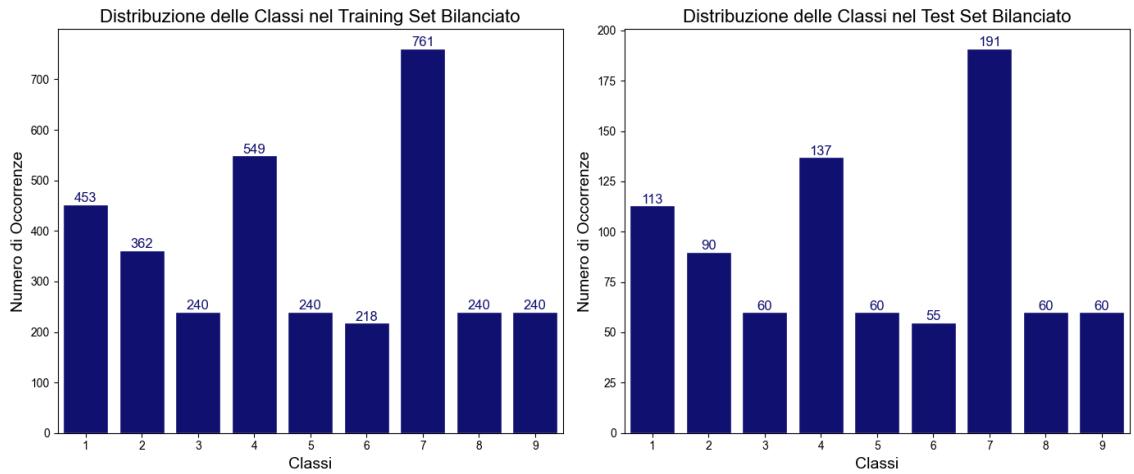


Figura 4.6: Divisione Dataset Bilanciato (SMOTE)

4.5.2.1 SMOTE

La tecnica **SMOTE** (**Synthetic Minority Over-sampling Technique**) è un metodo di **oversampling** che genera nuovi campioni sintetici per le classi minoritarie, basandosi sulle informazioni già esistenti. In particolare, SMOTE crea nuovi esempi combinando le caratteristiche di campioni esistenti appartenenti alla stessa classe, con l'obiettivo di migliorare la rappresentanza delle classi sbilanciate nel dataset. Il funzionamento di SMOTE è il seguente:

1. Per ogni campione della classe minoritaria, SMOTE seleziona uno o più dei suoi vicini più prossimi.
2. Vengono quindi creati nuovi campioni sintetici, interpolando tra il campione originale e i suoi vicini.

3. Questo processo aumenta il numero di campioni nelle classi minoritarie, bilanciando così la distribuzione delle classi.

Nel nostro caso, SMOTE è stato utilizzato per le classi **3**, **8**, e **9**, che presentavano un numero significativamente inferiore di campioni rispetto alle classi maggioritarie. Il bilanciamento ottenuto ha permesso al modello di considerare tutte le classi in modo **più equo** durante l'addestramento, riducendo il rischio di bias verso le classi più rappresentate. [8] [20]

Synthetic Minority Oversampling Technique

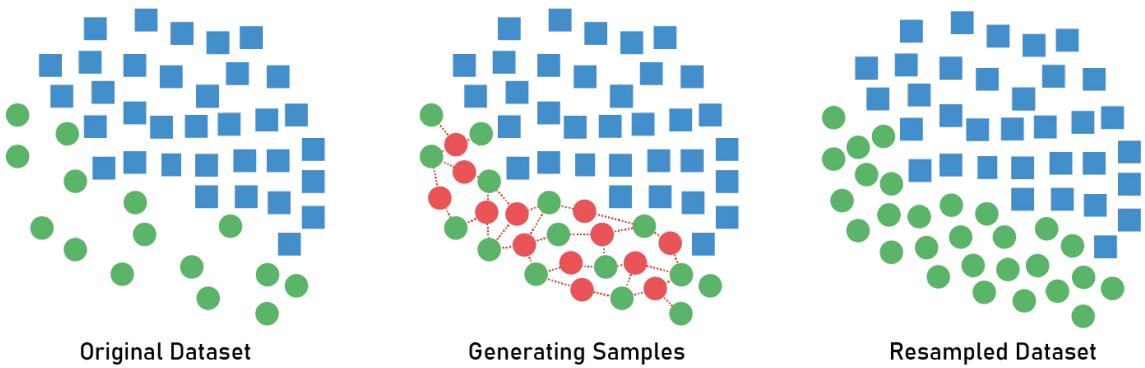


Figura 4.7: Esempio dell'OverSampling con SMOTE [21]

4.6 Modellazione, Addestramento e Valutazione dei Modelli

La fase di **modellazione**, **addestramento** e **valutazione** dei modelli è una delle più critiche nel processo di sviluppo di un sistema di Machine Learning. In questa sezione, vengono esplorati diversi modelli di classificazione, insieme alle tecniche di ottimizzazione degli iperparametri e alle metriche utilizzate per la valutazione delle prestazioni. Il con-

fronto tra modelli ci consente di scegliere quello che meglio si adatta al nostro problema, mentre l'ottimizzazione degli iperparametri mira a migliorare ulteriormente le prestazioni.

4.6.1 Confronto tra i Modelli di Classificazione

In questa sezione, vengono analizzati tre modelli di classificazione comunemente utilizzati: **K-Neighbors**, **Random Forest** e **XGBoost**. Ogni modello ha caratteristiche uniche che lo rendono adatto a specifiche situazioni, e il confronto tra di essi ci permette di scegliere il modello che offre le migliori prestazioni nel nostro caso.

4.6.1.1 K-Neighbors

Il **K-Neighbors** (KNN) è un algoritmo di classificazione basato sul principio che le istanze simili **appartengono alla stessa classe**. Per ogni punto da classificare KNN calcola la distanza tra il punto e i suoi K vicini più prossimi nel dataset di addestramento e assegna la classe in base alla classe della maggioranza dei suoi punti vicini. I principali vantaggi di K-Neighbors sono la sua **semplicità** e la capacità di gestire **relazioni non lineari**. KNN può essere **costoso** computazionalmente parlando per dataset di grandi dimensioni, poiché richiede il calcolo della distanza tra tutti i punti. E' necessario prestare attenzione nella configurazione di questo algoritmo, poichè KNN è sensibile alla scelta del parametro K e alla **normalizzazione delle caratteristiche**, infatti le variabili con scale molto diverse o un numero di vicini non adatto al problema possono influenzare negativamente le performance del modello. [22]

4.6.1.2 Random Forest

Il **Random Forest** è un ensemble learning method (combina le previsioni di più modelli) che costruisce un gran numero di alberi decisionali che vengono valutati tramite l'**indice**

di Gini, ovvero una metrica che valuta la qualità dei dati all'interno dei nodi e ne fa una media per ottenere la previsione finale. Ogni albero è addestrato su un gruppo di dati casuale e per ogni divisione viene presa, in modo casuale, una selezione di caratteristiche. Questo rende il Random Forest molto robusto contro l'overfitting e in grado di gestire variabili sia di tipo numerico che categorico. Il vantaggio principale del Random Forest è la sua capacità di **gestire dati complessi e ad alta dimensione**, come nel nostro caso, dove i dati derivano sia da variabili testuali (TF-IDF) che da variabili numeriche (Gene, Variation). Il Random Forest è anche molto utile nel determinare l' **importanza delle caratteristiche** identificando quali variabili influenzano maggiormente la classificazione.

[23] [24] [25]

4.6.1.3 XGBoost

XGBoost (Extreme Gradient Boosting) è un algoritmo di boosting che migliora le performance di modelli di tipo **decision tree** riducendo l'errore di previsione. Il tutto avviene costruendo un albero alla volta, dove ogni nuovo albero cerca di correggere iterativamente gli errori di quello precedente. XGBoost è molto valido per la sua capacità di gestire dati sbilanciati e per la velocità di addestramento, rendendolo uno dei modelli più utilizzati. XGBoost è particolarmente vantaggioso quando si lavora con grandi quantità di dati o quando la qualità dei dati è incerta. [26]

4.6.2 Ottimizzazione degli Iperparametri

L'ottimizzazione degli **iperparametri** è un passo fondamentale nel miglioramento delle prestazioni dei modelli di Machine Learning. Gli iperparametri sono parametri che non vengono appresi dal modello durante il training, ma devono essere impostati prima

dell’addestramento. Ottimizzare questi parametri permette di migliorare la capacità di generalizzazione del modello e prevenire fenomeni di overfitting o underfitting.

4.6.2.1 Scelta dei Parametri per i Modelli

Ogni modello ha una serie di iperparametri che influenzano il suo comportamento e la sua capacità di apprendimento.

```
# Definizione delle griglie di parametri per i modelli
def get_param_grids():
    param_grids = {
        "RandomForest": {
            'n_estimators': [100, 200, 300, 400],
            'max_depth': [30, 20, None],
            'min_samples_split': [2, 5],
            'class_weight': [None, 'balanced']
        },
        "K-Neighbors": {
            'n_neighbors': [3, 5, 7],
            'weights': ['uniform', 'distance'],
            'metric': ['euclidean', 'manhattan']
        },
        "XGBoost": {
            'n_estimators': [100, 200, 300],
            'max_depth': [3, 5, 7, 9],
            'learning_rate': [0.01, 0.1],
        }
    }
    return param_grids
```

Figura 4.8: Definizione Parametri per Modello

La **Figura 4.8** rappresenta la definizione delle griglie di parametri utilizzate per l’ottimizzazione dei tre modelli: **Random Forest**, **K-Nearest Neighbors (KNN)** e **XGBoost**.

I parametri inclusi sono descritti di seguito:

- **Random Forest:**

- **n_estimators**: Numero di alberi nella foresta (es. 100, 200, 300, 400).
- **max_depth**: Profondità massima di ogni albero (es. 30, 20, o senza limite specifico con **None**).

- **min_samples_split**: Numero minimo di campioni richiesti per dividere un nodo (es. 2 o 5).
- **class_weight**: Ponderazione delle classi per gestire dataset sbilanciati (es. `None` o `balanced`).

- **K-Nearest Neighbors (KNN)**:

- **n_neighbors**: Numero di vicini considerati nella stessa classe (es. 3, 5 o 7).
- **weights**: Metodo di ponderazione dei vicini può essere uniforme o in base alla distanza (`uniform` o `distance`).
- **metric**: Distanza utilizzata per il calcolo della vicinanza (es. `euclidean` o `manhattan`).

- **XGBoost**:

- **n_estimators**: Numero di alberi potenziati (es. 100, 200 o 300).
- **max_depth**: Profondità massima di ogni albero (es. 3, 5, 7 o 9).
- **learning_rate**: Tasso di apprendimento che controlla il contributo di ogni albero nel modello (es. 0.01 o 0.1).

Queste griglie vengono utilizzate per la ricerca dei parametri ottimali durante la fase di tuning tramite tecniche come `GridSearch` o `RandomizedSearch`, al fine di migliorare le performance dei modelli.

4.6.2.2 Grid Search e Cross-Validation

Una delle tecniche più comuni per ottimizzare gli iperparametri è il **Grid Search**, che esplora un insieme predefinito di combinazioni di iperparametri. La tecnica funziona creando una griglia di valori da testare e valutando la performance di ciascuna combinazione

tramite una tecnica di **cross-validation**. Questo processo garantisce che il modello venga testato su diverse combinazioni di parametri per trovare quella che offre le migliori prestazioni. La **cross-validation** è una tecnica che consiste nel suddividere il dataset in più sottoinsiemi (*folds*), addestrando il modello su alcuni di questi sottoinsiemi e testandolo sugli altri. Questo processo viene ripetuto più volte per garantire che il modello venga testato su tutti i dati disponibili, migliorando la robustezza della valutazione. Un'alternativa al **Grid Search** è il **Random Search**, che esplora lo spazio degli iperparametri selezionando casualmente le combinazioni da testare. Sebbene il **Random Search** possa risultare più efficiente in alcuni casi, poiché consente di coprire uno spazio di ricerca più ampio con meno combinazioni testate, in questo lavoro si è scelto di utilizzare il **Grid Search**. Questa scelta è motivata dalla necessità di eseguire una ricerca sistematica e completa, garantendo così di identificare la combinazione di iperparametri ottimale per il modello in questione. [16]

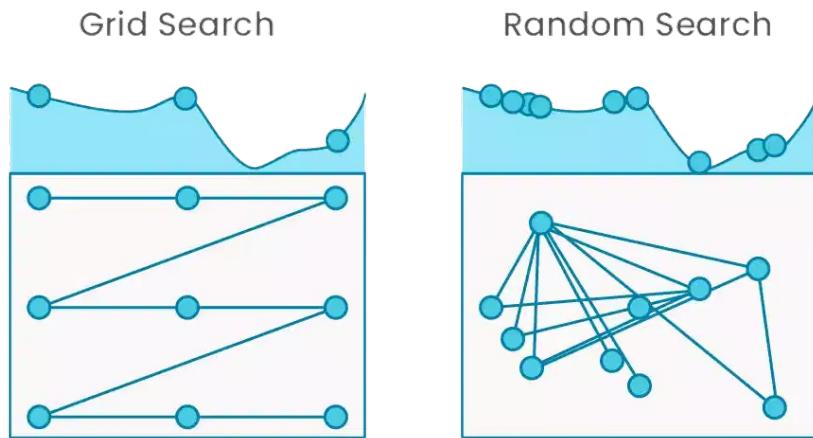


Figura 4.9: Confronto GridSearch e RandomSearch [27]

4.6.3 Valutazione dei Modelli

La **valutazione dei modelli** è una fase cruciale nel processo di Machine Learning, poiché consente di misurare l'efficacia e la capacità di generalizzazione del modello sui dati non visti, ovvero il **set di test**. Dopo aver addestrato il modello sui dati di training, è essenziale verificarne le prestazioni sul set di test, che non è stato utilizzato durante l'addestramento. Questo passaggio aiuta a prevenire il **overfitting**, ovvero quando il modello si adatta troppo strettamente ai dati di addestramento e perde la capacità di generalizzare. Il processo di valutazione include l'analisi delle metriche di performance, come **precisione**, **recall**, **F1-score**, **accuracy**, e **support**. Queste metriche forniscono una visione complessiva della qualità delle previsioni fatte dal modello, considerando sia le classi maggioritarie che minoritarie. La precisione e il recall vengono utilizzati per bilanciare il trade-off (la differenza tra due metriche in conflitto) tra i falsi positivi e i falsi negativi, mentre l'F1-score rappresenta una sintesi di questi due valori. L'accuratezza seppur molte volte utile deve essere interpretata con cautela in presenza di dataset sbilanciati come nel nostro caso. [16]

Capitolo 5

Valutazione Sperimentale

5.1 Presentazione dei Risultati

5.1.1 Risultati sul Dataset Originale

5.1.1.1 Performance dei Modelli

In questa sezione, vengono presentati i risultati ottenuti dai diversi modelli di Machine Learning addestrati sul dataset originale, caratterizzato da uno squilibrio di classe significativo. Sono stati impiegati tre modelli: **Random Forest**, **K-Nearest Neighbors (KNN)** e **XGBoost**. Ciascun modello è stato sottoposto a una ricerca a griglia (Grid Search) per ottimizzare gli iperparametri, utilizzando l'**F1-score pesato** come metrica di valutazione durante la validazione incrociata.

Random Forest		K-Nearest Neighbors	
Metrica	Valore	Metrica	Valore
F1-Score Totale	0.6358	F1-Score Totale	0.611
Accuratezza Totale	0.6461	Accuratezza Totale	0.6175

XGBoost	
Metrica	Valore
F1-Score Totale	0.6529
Accuratezza Totale	0.6596

Figura 5.1: Performance dei modelli sul dataset Originale

5.1.1.2 Confronto delle Metriche

Per confrontare le prestazioni dei modelli, sono state utilizzate diverse metriche di valutazione, tra cui **precisione**, **recall**, **F1-score** e **support** per ogni classe, oltre all'**accuratezza** e all'**F1-score totale pesato**. I risultati dettagliati per ciascun modello sono stati presentati in tabelle separate, consentendo un confronto diretto delle prestazioni su ciascuna classe e a livello globale.

Random Forest		K-Nearest Neighbors							
Classe	Precisione	Recall	F1-Score	Support	Classe	Precisione	Recall	F1-Score	Support
1	0.57	0.62	0.6	113	1	0.6	0.58	0.59	113
2	0.7	0.41	0.51	91	2	0.57	0.42	0.48	91
3	0.38	0.33	0.35	18	3	0.45	0.28	0.34	18
4	0.75	0.66	0.7	137	4	0.58	0.72	0.64	137
5	0.41	0.4	0.4	48	5	0.37	0.35	0.36	48
6	0.86	0.58	0.7	55	6	0.83	0.55	0.66	55
7	0.65	0.88	0.75	191	7	0.68	0.79	0.73	191
8	0.0	0.0	0.0	4	8	1.0	0.5	0.67	4
9	0.75	0.86	0.8	7	9	1.0	0.71	0.83	7

XGBoost				
Classe	Precisione	Recall	F1-Score	Support
1	0.63	0.6	0.62	113
2	0.71	0.52	0.6	91
3	0.43	0.33	0.38	18
4	0.69	0.7	0.7	137
5	0.33	0.33	0.33	48
6	0.86	0.56	0.68	55
7	0.68	0.88	0.77	191
8	1.0	0.25	0.4	4
9	1.0	0.71	0.83	7

Figura 5.2: Confronto delle Metriche sul dataset Originale

Come mostrato dalle **Figure 5.1** e **5.2**, i modelli che ottengono le migliori performance sono **Random Forest (63%)** e **XGBoost (65%)**. In particolare, **XGBoost** si distingue

per la capacità di classificare, anche se con difficoltà, le classi meno rappresentate.

5.1.2 Risultati sul Dataset Bilanciato

5.1.2.1 Performance dei Modelli

Per risolvere il problema dello squilibrio tra le classi presente nel dataset originale, è stata applicata la tecnica di **oversampling SMOTE**. I modelli **Random Forest**, **KNN** e **XGBoost** sono stati addestrati sul dataset bilanciato e sottoposti nuovamente alla ricerca dei migliori iperparametri, sfruttando le stesse griglie usate in precedenza.

Random Forest		K-Nearest Neighbors	
Metrica	Valore	Metrica	Valore
F1-Score Totale	0.7711	F1-Score Totale	0.7333
Accuratezza Totale	0.753	Accuratezza Totale	0.7143

XGBoost	
Metrica	Valore
F1-Score Totale	0.77
Accuratezza Totale	0.7554

Figura 5.3: Performance dei modelli sul dataset Bilanciato

5.1.2.2 Confronto delle Metriche

Analogamente al dataset originale, le prestazioni dei modelli sul dataset bilanciato sono state valutate utilizzando **precisione**, **recall**, **F1-score** e **supporto** per ogni classe, insieme all'**accuratezza totale** e all'**F1-score totale pesato**. Le tabelle comparative mostrano i risultati ottenuti per ciascun modello, permettendo di analizzare l'impatto del bilanciamento delle classi sulle performance.

Random Forest					K-Nearest Neighbors				
Classe	Precisione	Recall	F1-Score	Support	Classe	Precisione	Recall	F1-Score	Support
1	0.66	0.65	0.65	113	1	0.56	0.54	0.55	113
2	0.66	0.48	0.55	90	2	0.64	0.53	0.58	90
3	0.83	0.87	0.85	60	3	0.72	0.85	0.78	60
4	0.73	0.81	0.77	137	4	0.68	0.77	0.72	137
5	0.67	0.63	0.65	60	5	0.55	0.67	0.6	60
6	0.95	0.64	0.76	55	6	0.87	0.62	0.72	55
7	0.68	0.79	0.73	191	7	0.73	0.7	0.71	191
8	0.97	1.0	0.98	60	8	0.95	0.98	0.97	60
9	1.0	1.0	1.0	60	9	0.97	0.97	0.97	60

XGBoost				
Classe	Precisione	Recall	F1-Score	Support
1	0.7	0.62	0.66	113
2	0.68	0.51	0.58	90
3	0.87	0.78	0.82	60
4	0.71	0.82	0.76	137
5	0.58	0.68	0.63	60
6	0.92	0.64	0.75	55
7	0.71	0.81	0.76	191
8	1.0	0.98	0.99	60
9	1.0	0.97	0.98	60

Figura 5.4: Confronto delle Metriche sul dataset Bilanciato

Come visibile nelle **Figure 5.3** e **5.4**, anche in questo caso i modelli più performanti sono stati **Random Forest (77,1%)** e **XGBoost (77%)**. Oltre a un effettivo miglioramento generale, è interessante notare che, grazie al bilanciamento delle classi minoritarie, **Random Forest** è riuscito a ottenere, anche se di poco, prestazioni superiori rispetto a **XGBoost**, nonostante quest'ultimo sia considerato un modello più avanzato.

5.1.3 Confronto tra Dataset Originale e Bilanciato

5.1.3.1 Miglioramento nelle Performance

Il confronto diretto dei risultati ottenuti sui due dataset evidenzia chiaramente l'impatto del bilanciamento delle classi sulle prestazioni dei modelli. In particolare è possibile osservare un miglioramento significativo in termini di **F1-score** per le classi minoritarie, questo conferma l'efficacia della tecnica **SMOTE** nel gestire lo squilibrio delle classi. Sebbene l'**accuratezza totale** possa mostrare variazioni, l'**F1-score totale pesato** che considera il bilanciamento delle classi tende ad aumentare quando i modelli vengono addestrati sul

dataset bilanciato. In entrambi gli esperimenti, i modelli che si sono dimostrati più adatti a questa tipologia di classificazione sono **Random Forest** e **XGBoost**, raggiungendo complessivamente ottime prestazioni di apprendimento. Inoltre, è evidente un miglioramento nella capacità di classificare le mutazioni genetiche su tutte e **9 classi** presenti, a conferma della validità dell'approccio adottato.

5.1.3.2 Confronto delle Matrici di Confusione

Per analizzare visivamente le prestazioni dei modelli, sono riportate le **matrici di confusione** relative al **dataset originale** e al **dataset bilanciato**. Le matrici di confusione forniscono un quadro chiaro della distribuzione delle predizioni corrette e degli errori di classificazione per ciascun modello e consentono di evidenziare eventuali miglioramenti nella capacità di classificare le classi meno rappresentate.

Random Forest: Le matrici di confusione per il modello Random Forest sono mostrate nella **Figura 5.5**. Nel dataset originale, il modello presenta difficoltà nel classificare le classi minoritarie, con un numero significativo di predizioni errate o mancanti. Nel dataset bilanciato, si osserva un miglioramento nella classificazione delle classi meno rappresentate, con una riduzione dei falsi negativi.

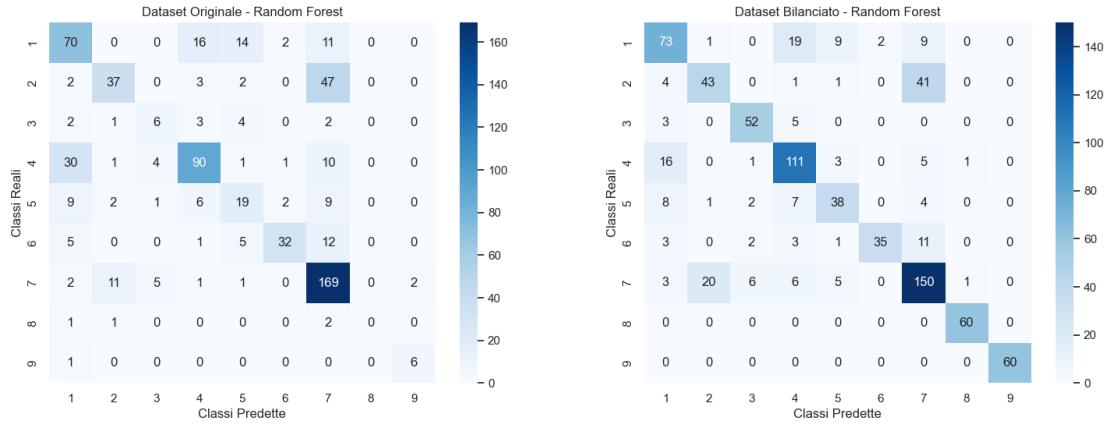


Figura 5.5: Confronto Matrici di Confusione per Random Forest

K-Nearest Neighbors (KNN): Le matrici di confusione per il modello KNN sono riportate nella **Figura 5.6**. Nel dataset originale, il modello fatica a gestire le classi poco rappresentate. Tuttavia, con il dataset bilanciato, le predizioni corrette aumentano per tutte le classi, anche se il modello risulta comunque meno efficace rispetto agli altri.

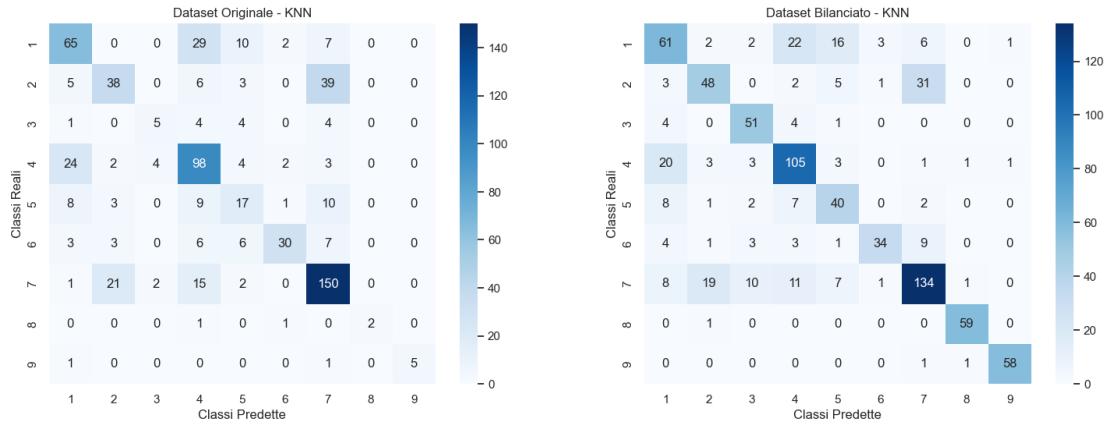


Figura 5.6: Confronto Matrici di Confusione per K-Neighbors

XGBoost: Le matrici di confusione per il modello XGBoost sono presentate nella **Figura 5.7**. Sul dataset originale, XGBoost mostra buone prestazioni nel classificare le

classi principali. Con il dataset bilanciato, il modello migliora ulteriormente, riducendo significativamente gli errori per tutte le classi.

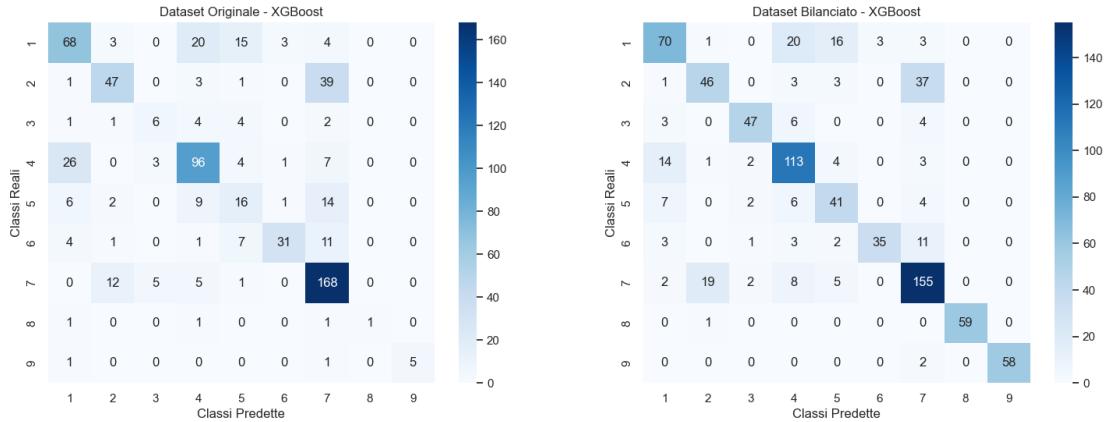


Figura 5.7: Confronto Matrici di Confusione per XGBoost

5.1.3.3 Limitazioni e osservazioni

Sebbene il bilanciamento delle classi abbia portato a un miglioramento generale delle prestazioni, è fondamentale considerare alcune limitazioni e potenziali problematiche legate all'uso della tecnica **SMOTE**. In primo luogo, la creazione di istanze sintetiche potrebbe introdurre **rumore** o **bias** nel dataset, specialmente se le caratteristiche delle nuove istanze non rispecchiano in modo fedele la distribuzione reale dei dati. Questo può influenzare negativamente la capacità del modello di generalizzare su dati non visti. La performance dei modelli è strettamente legata alla scelta degli **iperparametri**, che potrebbero non essere ottimali in alcuni casi, soprattutto se la ricerca degli iperparametri non è stata sufficientemente approfondita, questo a causa anche della non trascurabile potenza computazionale richiesta. A ciò si aggiunge la **complessità intrinseca del problema** che rende difficile per i modelli distinguere tra classi con confini poco definiti come nel nostro caso. Rimane comunque necessario denotare che l'utilizzo di SMOTE aumenta il volume di dati da elaborare con un conseguente **incremento dei tempi di calcolo**, questo aspetto può diventare

particolarmente critico per dataset molto grandi o per modelli complessi come **XGBoost**. Nonostante queste limitazioni, il bilanciamento delle classi rimane una strategia efficace per migliorare le prestazioni dei modelli, soprattutto per le classi minoritarie. E' essenziale però valutare attentamente i compromessi e implementare tecniche complementari come la rimozione del rumore o un ulteriore ottimizzazione degli iperparametri, per mitigare gli effetti indesiderati e ottenere risultati più affidabili.

Capitolo 6

Sviluppi Futuri

6.1 Possibili Sviluppi Futuri

6.1.1 Integrazione di Dati più Recenti

L'utilizzo di **dati più aggiornati e completi** potrebbe rappresentare un fattore determinante per migliorare l'efficacia dei modelli di classificazione. L'integrazione con informazioni aggiornate o provenienti da studi clinici recenti potrebbe migliorare la qualità del dataset, in modo tale da consentire una **migliore generalizzazione** e un **incremento dell'accuratezza** nei processi decisionali.

6.1.2 Sviluppo di Modelli più Avanzati

L'adozione di modelli di Machine Learning più avanzati come le **reti neurali profonde (Deep Learning)** potrebbe offrire una capacità di generalizzazione migliore rispetto ai modelli tradizionali. Inoltre, tecniche come il reinforcement learning potrebbero essere esplorate per migliorare l'adattamento del modello ai cambiamenti nei dati. [16]

6.1.3 Applicazione in Altri Ambiti Oncologici

I modelli sviluppati potrebbero essere applicati in altri ambiti oncologici, come la **predizione della risposta ai trattamenti** o la **diagnosi precoce** in tipologie di tumori specifici. L'adattamento dei modelli a **nuove patologie** potrebbe aprire nuove strade per l'utilizzo della tecnologia in **medicina personalizzata**.

6.2 Limitazioni dello Studio

6.2.1 Disponibilità dei Dati: Dataset Limitato al 2017

Una delle principali limitazioni di questo studio risiede nella **disponibilità limitata dei dati**, che si arrestano al 2017. Questa restrizione compromette non solo la rappresentatività dei modelli, ma anche la qualità delle previsioni, poiché trattamenti e approcci diagnostici potrebbero essere cambiati nel tempo, alterando la distribuzione delle classi nel dataset. Inoltre, il dataset evidenzia uno **sbilanciamento delle classi**, che limita la capacità del modello di identificare correttamente le classi meno rappresentate. Come dimostrano i risultati tale squilibrio riduce l'accuratezza del modello soprattutto nelle previsioni relative alle **classi minoritarie**. E' importante considerare che molte mutazioni genetiche incluse nel dataset presentavano descrizioni **deboli e incomplete**, negli ultimi anni i progressi scientifici potrebbero aver apportato conoscenze importanti che potrebbero modificare radicalmente le valutazioni e le prestazioni dei modelli analizzati. [11]

6.2.2 Necessità di Ulteriore Validazione Clinica

Nonostante i risultati promettenti ottenuti dai modelli risulta necessaria una **validazione clinica** più approfondita per confermare la loro **efficacia** in contesti reali. La validazione

potrebbe includere test su un campione di pazienti più ampio e diversificato, con dati provenienti da diverse **strutture ospedaliere** e **contesti geografici** in modo tale da poter verificarne l'accuratezza e la generalizzabilità dei modelli.

Elenco delle figure

1.1	Mutazioni del DNA[1]	2
1.2	Proliferazione Cellulare Incontrollata [2]	3
3.1	Le 9 Classi	18
3.2	Distribuzione delle Classi	19
3.3	10 Geni più frequenti nel Dataset	20
3.4	Distribuzione Percentuale dei Geni per Classe	21
3.5	10 Varianti più frequenti nel Dataset	22
3.6	Distribuzione della Lunghezza dei Testi Preprocessati	23
3.7	WordCloud dei Termini più Frequenti	24
3.8	Valori Mancanti	25
3.9	Distribuzione delle Classi Aggiornata	26
4.1	Modello del Workflow: Approccio e Metodologia	31
4.2	Preprocessing del Testo	34
4.3	One Hot Encoding per le Feature Categoriali	36
4.4	TF-IDF, SVD e Normalizzazione	38
4.5	Divisione Dataset Originale	40
4.6	Divisione Dataset Bilanciato (SMOTE)	41

4.7	Esempio dell'OverSampling con SMOTE [21]	42
4.8	Definizione Parametri per Modello	45
4.9	Confronto GridSearch e RandomSearch [27]	47
5.1	Performance dei modelli sul dataset Originale	50
5.2	Confronto delle Metriche sul dataset Originale	50
5.3	Performance dei modelli sul dataset Bilanciato	51
5.4	Confronto delle Metriche sul dataset Bilanciato	52
5.5	Confronto Matrici di Confusione per Random Forest	54
5.6	Confronto Matrici di Confusione per K-Neighbors	54
5.7	Confronto Matrici di Confusione per XGBoost	55

Conclusioni

Il lavoro di tesi presentato in questo elaborato ha rappresentato un'esplorazione approfondita delle potenzialità offerte dal **Machine Learning** e dal **Natural Language Processing** applicati a un campo complesso come quello della genetica oncologica. L'obiettivo di esplorare e proporre metodi innovativi per la classificazione delle mutazioni genetiche è stato perseguito con un approccio rigoroso, combinando fra loro conoscenze teoriche e competenze pratiche. Il lavoro svolto dimostra come l'interazione tra tecnologia e scienza possa aprire nuove strade verso una migliore comprensione delle patologie e verso lo sviluppo di soluzioni più mirate e personalizzate. Attraverso l'analisi dei dati genetici e testuali, è stato possibile non solo sviluppare modelli capaci di fornire previsioni affidabili, ma anche contribuire alla crescente integrazione delle tecniche di **Intelligenza Artificiale** nella ricerca biomedica. Dal punto di vista personale e accademico, questo progetto ha rappresentato una sfida stimolante e arricchente. Il percorso intrapreso mi ha permesso di approfondire temi complessi, di mettere in pratica strumenti avanzati e di comprendere più a fondo l'impatto che tecnologie come il Machine Learning possono avere sulla società e sulla medicina moderna. Mi auguro che il lavoro da me svolto possa contribuire, anche se in minima parte, al progresso scientifico e alla lotta contro il cancro, dimostrando ancora una volta come l'innovazione tecnologica possa diventare un alleato fondamentale nella medicina moderna.

Bibliografia

- [1] Treating Mutations in Cancer Research | LIDE Biotech.
- [2] Vecteezy.
- [3] M. Lauricella. Biochimica delle cellule tumorali. In *Biochimica delle cellule tumorali*. IT, 2018. Accepted: 2019-11-18T11:20:29Z.
- [4] P Cappelletti. La Medicina Personalizzata fra ricerca e pratica clinica: il ruolo della Medicina di Laboratorio. *La Medicina Personalizzata fra ricerca e pratica clinica: il ruolo della Medicina di Laboratorio*, 2009.
- [5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [6] Zhi-Hua Zhou. *Machine Learning*. Springer Nature, August 2021.
- [7] M Emms and S Luz. Machine Learning for Natural Language Processing. *Machine Learning for Natural Language Processing*, 2007.

- [8] Alberto Fernandez, Salvador Garcia, Francisco Herrera, and Nitesh V. Chawla. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *jair*, 61:863–905, April 2018.
- [9] Maxwell W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, June 2015.
- [10] Amit Bhola and Arvind Kumar Tiwari. Machine learning based approaches for cancer classification using gene expression data. *Machine Learning and Applications: An International Journal*, 2(3/4):01–12, 2015.
- [11] Iker Huerga and Wendy Kan. Personalized medicine: Redefining cancer treatment. <https://kaggle.com/competitions/msk-redefining-cancer-treatment>, 2017. Kaggle.
- [12] Suresh Kumar Mukhiya and Usman Ahmed. *Hands-On Exploratory Data Analysis with Python: Perform EDA techniques to understand, summarize, and investigate your data*. Packt Publishing Ltd, March 2020. Google-Books-ID: QcHZDwAAQBAJ.
- [13] Kabita Sahoo, Abhaya Kumar Samal, Jitendra Pramanik, and Subhendu Kumar Pan. Exploratory data analysis using python. *International Journal of Innovative Technology and Exploring Engineering*, 8(12):4727–4735, 2019.
- [14] Federico Cismondi, André S Fialho, Susana M Vieira, Shane R Reti, João MC Sousa, and Stan N Finkelstein. Missing data in medical databases: Impute, delete or classify? *Artificial intelligence in medicine*, 58(1):63–72, 2013.

- [15] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de Las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.
- [16] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* "O'Reilly Media, Inc.", October 2022. Google-Books-ID: X5ySEAAAQBAJ.
- [17] Jason Brownlee. Machine Learning Mastery With Python. *Machine Learning Mastery With Python*, 2016.
- [18] Robert Dzisevič and Dmitrij Šešok. Text Classification using Different Feature Extraction Approaches. In *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–4, April 2019.
- [19] Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. *Using TF-IDF to Determine Word Relevance in Document Queries*, 2003.
- [20] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [21] Zaki Jefferson. Bank Data: SMOTE, August 2020.
- [22] Oliver Kramer and Oliver Kramer. K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23, 2013.
- [23] Yanjun Qi. Random forest for bioinformatics. *Ensemble machine learning: Methods and applications*, pages 307–323, 2012.

- [24] Luis Eduardo Boiko Ferreira, Heitor Murilo Gomes, Albert Bifet, and Luiz S Oliveira. Adaptive random forests with resampling for imbalanced data streams. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.
- [25] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13–20, 2012. Proceedings 8*, pages 154–168. Springer, 2012.
- [26] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [27] 360DigiTMG Group. Difference between GridSearchCV and RandomizedSearchCV - 360DigiTMG.