UNIVERSITÀ DEGLI STUDI
DI SALERNO

Corso di Laurea in Informatica

Coronavirus Dyagnos-AI

Autore:
Emmanuel De Luca

Docenti:
Prof. Polese Giuseppe
Prof.ssa Caruccio Loredana

Contents

1 Introduzione al problema

1.1 Il Machine Learning e la medicina di precisione

Il tema della medicina di precisione con l'ausilio dell'intelligenza artificiale è un tema ampiamente trattato in letteratura scientifica. Lo scopo principale di molti studi è l'applicazione di metodi di Machine Learning per definire un modello computazionale di classificazione che riesca a discriminare tra una malattia piuttosto che un'altra.

1.2 Il Covid-19 e la medicina di precisione

Durante il triennio 2020/23 la malattia da pandemia Coronavirus 19 è stata ampiamente oggetto di studio in letteratura. L'applicazione di Modelli di IA e Machine Learning per una diagnosi più accurata della malattia è una delle svolte più interessanti in questo ambito. In particolare, oggetto di questo progetto e in parte dello studio pubblicato preso come base di partenza, le varie analisi dei sintomi; delle routine giornaliere e dei dati clinici dei pazienti sono state utilizzate in diversi studi per cercare di creare modelli molto più precisi di altri già esistenti.

1.3 Il problema degli approcci IA in ambito medico

Come qualsiasi approccio di Machine Learning e IA, il problema principale individuato è la reperibilità dei dati. La discrepanza tra i dati "accademici" e quelli reali rende molti dei modelli inutilizzabili nell'effettivo, soprattutto perchè c'è la necessità di poter utilizzare dati reali e non solo dati di sintesi. Inoltre un'altro dei problemi principali di modelli del genere è l'explainability, ossia il processo in cui si spiega all'utente finale il processo decisionale che ha portato il modello a dare una risposta piuttosto che un'altra.

1.4 Il problema affrontato

Partendo da uno studio pubblicato in letteratura l'intenzione di questo progetto è quello di definire un modello che analizzi i dati classificando correttamente malati di influenza H1N1 e malati di Covid-19, cercando di porre maggiore attenzione sui dati presenti nel dataset, prestando maggiore attenzione sull'explainability, cercando di trovare un approccio migliore per affrontare il problema utilizzando approcci studiati.

Nota: Le fonti principali per questa prima analisi del problema sono reperibili ai seguenti link: ***Using machine learning of clinical data to diagnose COVID-19: a systematic review and meta-analysis*** e ***All models are wrong and yours are useless: making clinical prediction models impactful for patients***

1.4.1 Il problema dell'explainability nei modelli di Machine Learning

Uno dei problemi principali del Machine Learning, che rende un modello di machine learning ottimo in letteratura ma non utilizzabile nella realtà, è l'explainability stessa del modello, ossia quanto il processo decisionale su cui si basa il modello sia comprensibile a chi lo utilizza. Oltre a migliorare la fiducia verso un approccio di IA, migliorare l'explanability di un modello porta benefici in ambito legale e di miglioramento del modello stesso. Un esempio potrebbe essere un modello di Machine Learning che decide se assumere o meno una persona, migliorando l'explainability del modello si può capire il processo decisionale che sta dietro una decisione, rendendo il processo stesso trasparente e affidabile, permettendo di modificare decisioni che portano in errore il modello stesso.

1.4.2 E in ambito clinico?

Migliorare l'explainability di un modello in ambito clinico permette di rendere il processo decisionale dietro una predizione più affidabile e trasparente: un medico che non sa perchè un modello ha diagnosticato una malattia ad un paziente non si fiderà mai del modello, un paziente a cui viene diagnosticata una malattia rara da un modello di machine learning, ma di cui il medico non riesce a spiegare il processo decisionale, non si fiderà mai della diagnosi. È chiaro quindi che anche in ambito clinico l'explainability del modello è un fattore molto delicato di cui preoccuparsi, in quanto potrebbe anche rendere un modello eccellente in letteratura non usabile nel concreto.

1.4.3 Il problema dei dati mancanti

Un'altro dei principali problemi in ambito Machine Learning, medico o meno, è sicuramente la mancanza e la scarsità di dati, che in certi casi porta il modello a fallire. Nello studio utilizzato per raccogliere i dati è stata posta poca attenzione sulla gestione dei dati mancanti nel dataset, dato che l'algoritmo utilizzato (*XGBoost*) può gestire dati sparsi automaticamente, il che implica la capacità di trattare i valori nulli in modo appropriato senza che prima venga fatto un pre-processing accurato, aspetto che in questo progetto verrà trattato in modo più approfondito.

2 Data Understanding and Data Preparation

2.1 Data Gathering

Il dataset utilizzato e le risorse necessarie per proseguire con il completamento del progetto sono stati raccolti dal seguente studio condotto in letteratura: ***Using machine learning of clinical data to diagnose COVID-19: a systematic review and meta-analysis***. Ulteriori spunti sono ricavanti invece dal seguente studio: ***All models are wrong and yours are useless: making clinical prediction models impactful for patients***. Il paper utilizzato per la raccolta del materiale è corredato dalla seguente repository: *link*.

2.2 Struttura della repository originale

All'interno della repository GitHub da cui sono stati raccolti i dati possiamo trovare il Jupyter Notebook dello studio, in cui sono stati riportati i vari passaggi della fase di testing e addestramento del modello, oltre che la confusion matrix, il decision tree utilizzato per definire uno dei modelli implementati e migliorare l'explainability delle predizioni, il grafico che mostra il rate di falsi negativi e falsi positivi dei vari approcci utilizzati nel loro studio. Oltre ciò è anche possibile trovare il dataset grezzo, sotto il nome di "*COVIDandFLUdata.csv*", che il dataset in cui sono stati eliminati tutti i dati discriminatori per la predizione, sotto il nome di "*UsedCombined.txt*".

2.2.1 Dati Discriminatori: un problema per le predizioni?

Come detto nel paragrafo precedente il dataset originale presenta delle feature che risultano essere discriminatorie per la predizione, in particolare per i pazienti positivi al Covid-19 è indicata anche la regione di appartenenza e se ha contratto la malattia a Wuhan o meno. Nella repository da cui è stato reperito il dataset tuttavia è presente anche la versione dello stesso senza queste feature discriminatorie. É importante notare che i dati presenti all'interno del dataset sono stati raccolti da una serie di pubblicazioni e unite in un solo dataset dagli autori dello studio.

Nota: All'interno della repository originale (*Link*) è presente il file in cui viene riportato, paziente per paziente, il paper da cui sono stati estrapolati i dati, sotto il nome di "*LiteratureSearchDataindividualpatients.tsv*"

2.3 Data Examination

Il primo passo svolto è stato convertire il dataset da ".txt" con valori formattati da spazi, ad un file formattato con virgole ".csv". Il dataset reperito è composto da 1486 righe e 22 colonne, contenente informazioni generali del paziente e i sintomi riscontrati. Le varibili inerenti le informazioni del paziente sono divise tra categoriche (il sesso, il community trasmission e la regione) e continue (l'età). Le variabili inerenti i sintomi sono per la maggio parte variabili categoriche, tranne per la temperatura; i linfociti; il livello di siero e i neutrofili che sono rappresentati come variabili continue. Il dataset oltre a presentare molte feature presenta anche molti valori nulli, come sarà meglio evidenziato nelle sezioni successive. In questa pagina verrano elencate e approfondite tutte le feature presenti.

Feature presenti:

1. **Diagnosis**: feature utilizzata per dare il risultato della predizione del modello, i valori che può assumere sono [*H1N1, COVID19*].
2. **D**: feature utilizzata per dare il risultato della predizione del modello, i valori che può assumere sono [*0, 1*].
3. **Age**: feature utilizzata per indicare l'età del paziente, i valori che può assumere sono continui, in questo contesto i valori assunti oscillano nel range [*1, 81*].
4. **Sex**: feature utilizzata per indicare il sesso del paziente, i valori che può assumere sono [*M, F*].
5. **Neutrophil**: feature utilizzata per indicare il livello di globuli bianchi nel sangue, i valori che assumono oscillano sono nel range [*0.2, 91*].
6. **NeutrophilCategorical**: feature utilizzata per indicare il livello medio di neutrofili in cui identificare l'individuo, i valori vengono identificati in tre classi principali [*low, normal, high*].
7. **SerumLevelsOfWhiteBloodCell**: feature utilizzata per indicare il livello di siero nei globuli bianchi, i valori di siero oscillano nel range [*0.5, 36.07*].
8. **SerumLevelsOfWhiteBloodCellCategorical**: feature utilizzata per indicare il livello medio di siero nei globuli bianchi, i valori vengono identificati in tre classi principali [*low, normal, high*].
9. **Lymphocytes**: feature utilizzata per misurare il livello di linfociti nel sangue, i valori assunti dalla feature oscillano tra [*0.01, 30*]

10. **LymphocytesCategorical**: feature utilizzata per indicare il livello medio di linfociti nel sangue, i valori vengono identificati in tre classi principali [*low, normal, high*].
11. **CTscanResults**: feature usata per indicare i risultati del test CT per verificare la presenza o meno di polmonite, i valori che può assumere sono [*pos, neg*].
12. **XrayResults**: feature utilizzata per indicare i risultati del test radiografico effettuato sul torace dei pazienti, i valori che può assumenre sono [*pos, neg*].
13. **Diarrhea**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
14. **Fever**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
15. **Coughing**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
16. **SoreThroat**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
17. **NauseaVomitting**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
18. **Temperature**: feature utilizzata per riportare la temperatura corporea del paziente, i valori che assume oscillano nel range [*36.5, 40*].
19. **Fatigue**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
20. **RenalDisease**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].
21. **Diabetes**: feature che indica la presenza o meno del sintomo per il paziente, i valori che assume sono [*yes, no*].

2.4 Data Cleaning

Una fase importante del Data preparation risulta essere quella di data cleaning, ossia, tutta quella serie di operazioni che portano al miglioramento qualitativo dei dati presenti nel dataset.

2.4.1 Verifica dei valori nulli

Un'operazione importante di data cleaning è la verifica della presenza o meno di valori nulli all'interno del dataset, risolvibile tramite imputazione o cancellazione delle colonne/righe ma non senza effetti collaterali sul dataset, dato che un imputazione o una cancellazione potrebbero portare a dati poco rappresentativi o addirittura alla cancellazione di dati importanti.

Di seguito verrà riportata l'analisi del dataset e dei valori nulli.

Column	Non-Null Count	Null Count	Dtype
Diagnosis	1485	0	object
D	1485	0	int64
Age	1457	28	float64
Sex	1409	76	object
neutrophil	103	1382	float64
neutrophilCategorical	148	1337	object
serumLevelsOfWhiteBloodCell	151	1334	float64
serumLevelsOfWhiteBloodCellCategorical	191	1294	object
lymphocytes	156	1329	float64
lymphocytesCategorical	197	1288	object
CTscanResults	161	1324	object
XrayResults	47	1438	object
Diarrhea	450	1035	object
Fever	926	559	object
Coughing	862	623	object
SoreThroat	670	815	object
NauseaVomitting	422	1063	object
Temperature	629	856	float64
Fatigue	531	954	object
RenalDisease	226	1259	object
diabetes	226	1259	object

Table 1: Analisi del dataset con valori nulli

Come possiamo osservare tranne per la variabile dipendente *"Diagnosis"* e per la feature *"D"*, utilizzate entrambe per codificare la predizione del modello il dataset presenta molti valori nulli.

2.4.2 Verifica dei duplicati

Un'operazione di data cleaning, importante quanto l'operazione di verifica dei valori nulli, è sicuramente la verifica dei duplicati. I duplicati sono record in un dataset che si ripetono più volte, portando quindi potenziali bias. Di seguito saranno riportate le informazioni inerenti ai duplicati.

Numero di duplicati nel dataset: **307**.

Nel dataset è presente un numero alto di record duplicati, è stato quindi ritenuto opportuno eliminarli.

Grandezza dataset senza duplicati: **1178 records**.

2.4.3 Verifica dei valori nulli senza duplicati

A questo punto possiamo verificare che nel nuovo dataset i valori nulli sono distribuiti come segue.

Column	Non-Null Count	Null Count	Dtype
Diagnosis	1178	0	object
D	1178	0	int64
Age	1160	18	float64
Sex	1119	59	object
neutrophil	103	1075	float64
neutrophilCategorical	148	1030	object
serumLevelsOfWhiteBloodCell	151	1027	float64
serumLevelsOfWhiteBloodCellCategorical	190	988	object
lymphocytes	156	1022	float64
lymphocytesCategorical	197	981	object
CTscanResults	159	1019	object
XrayResults	47	1131	object
Diarrhea	431	747	object
Fever	894	284	object
Coughing	829	349	object
SoreThroat	640	538	object
NauseaVomitting	403	775	object
Temperature	616	562	float64
Fatigue	521	657	object
RenalDisease	220	958	object
diabetes	220	958	object

Table 2: Dataset senza duplicati

2.4.4 Imputazione e gestione dei valori nulli

Dalle analisi svolte sul dataset è facilmente intuibile che i valori nulli sono molto presenti nel dataset, tuttavia alcuni di essi sono facilmente gestibili. La colonna ***"neutrophil"*** risulta avere il *91%* dei valori nulli, e dato che le stesse informazioni sono riportate nella colonna ***neutrophilCategorical***, che risulta avere una percentuale minore di dati mancanti, è stato ritenuto opportuno eliminare la colonna ***"neutrophil"***. Stesso discorso vale per ***"serumLevelsOfWhiteBloodCell"*** e per ***"lymphocytes"***, le cui rispettive colonne con variabili categoriche hanno percentuale minore di dati mancanti. La colonna ***"Age"*** ha subito l'imputazione per media, la colonna ***"Sex"***, la colonna ***"Coughing"*** e la colonna ***"SoreThroat"*** hanno invece subito l'imputazione per moda, dato che i valori mancanti erano in percentuale piccolissima. Per quanto riguarda la colonna ***"Fever"***, dato che presenta solo il *24%* di dati mancanti, è stata fatta imputazione per moda, portando successivamente a eseguire un imputazione sulla colonna ***"Temperatura"*** dato che si può facilmente dedurre che se il valore di ***"Fever"*** è *"Yes"* allora la temperatura è più alta di *36.5*, in caso contrario la temperatura è uguale a *36.5*. Partendo quindi dalle imputazioni fatte sulla colonna ***"Fever"*** e sulla colonna ***"Temperature"*** è stato eseguito un aggiustamento sui valori della colonna ***"Fever"***, dato che in molti record risultava discordante con quanto riportato nella colonna ***"Temperature"***. Per le colonne ***"Coughing"*** e ***"SoreThroat"*** si è scelto di utilizzare il metodo di imputazione per moda, dato che i valori mancanti risultavano essere in percentuale bassa. Per quanto riguarda la colonna ***"XrayResults"*** e la colonna ***"CTscanResults"***, dato che le informazioni riportate nelle stesse risultano essere di fondamentale importanza per la riuscita delle previsioni, è stata effettuata l'imputazione categorica dei valori nulli con la creazione della categoria *"Non_Presente"*.

2.4.5 Creazione delle variabili "Dummy"

Dopo le opportune modifiche al dataset i valori mancanti risultavano essere tutti di colonne categoriche la cui mancanza non dipendendeva nè dai dati osservati nè da quelli non osservati, le informazioni presenti tuttavia risultavano essere essenziali per le predizioni e la loro cancellazione avrebbe portato a perdita di informazioni importanti. È stato quindi deciso che in fase di codifica delle variabili categoriche in variabili numeriche la tecnica utilizzata sarebbe stata quella della codifica "one-hot" tramite utilizzo di variabili dummy, ossia variabili binarie che permettono di convertire in "1" e "0" le variabili categoriche. Questa tecnica ha permesso di "aggirare" la presenza di variabili nulle in colonne necessarie per le predizioni e non imputabili.

2.4.6 Risultato finale

Dopo aver eseguito una ulteriore cancellazione dei duplicati, dopo aver eseguito il processo di imputazione, dopo aver cancellato la colonna *"Dyagnosis"*, in quanto riportante informazioni ridondanti, e dopo aver convertito tutte le variabili categoriche in variabili binarie, il dataset risultante risulta avere la seguente percentuale di valori nulli:

Colonna	**Valori Nulli**
D	0.0
Age	0.0
Temperature	0.0
Sex_F	0.0
Sex_M	0.0
neutrophilCategorical_high	0.0
neutrophilCategorical_low	0.0
neutrophilCategorical_normal	0.0
serumLevelsOfWhiteBloodCellCategorical_Low	0.0
serumLevelsOfWhiteBloodCellCategorical_Normal	0.0
serumLevelsOfWhiteBloodCellCategorical_high	0.0
serumLevelsOfWhiteBloodCellCategorical_low	0.0
serumLevelsOfWhiteBloodCellCategorical_normal	0.0
lymphocytesCategorical_High	0.0
lymphocytesCategorical_Low	0.0
lymphocytesCategorical_Normal	0.0
CTscanResults_Inconclusive	0.0
CTscanResults_Neg	0.0
CTscanResults_Non_Presente	0.0
CTscanResults_Pos	0.0

Table 3: Percentuale dati nulli

Colonna	Valori Nulli
CTscanResults_neg	0.0
XrayResults_Neg	0.0
XrayResults_Non_Presente	0.0
XrayResults_Pos	0.0
Diarrhea_No	0.0
Diarrhea_Yes	0.0
Fever_No	0.0
Fever_Yes	0.0
Coughing_No	0.0
Coughing_Yes	0.0
SoreThroat_No	0.0
SoreThroat_Yes	0.0
NauseaVomitting_No	0.0
NauseaVomitting_Yes	0.0
Fatigue_No	0.0
Fatigue_Yes	0.0
RenalDisease_No	0.0
RenalDisease_Yes	0.0
diabetes_No	0.0
diabetes_Yes	0.0

Table 4: Percentuale dati nulli

Come possiamo osservare dai risultati delle operazioni eseguite sul dataset nessuna colonna risulta avere dati con valore nullo.

2.5 Data Exploration

Una volta terminata l'analisi dei dati e la pulizia del dataset, è stato ritenuto opportuno rimuovere la colonna "unamed", in quanto non utilizzata se non per indicare il numero di colonna nel dataset originale. È opportuno ora quindi capire la struttura assunta dal dataset. Di seguito verranno quindi mostrate le informazioni inerenti al nuovo dataset creato.

La nuova dimensionalità del dataset: ***(1147 records, 51 colonne/features)***.

Elenco delle colonne:

Column	Non-Null Count	Null Count	Dtype
D	1147	0	int64
Age	1147	0	float64
Temperature	1147	0	float64
Sex_F	1147	0	int64
Sex_M	1147	0	int64
neutrophilCategorical_high	1147	0	int64
neutrophilCategorical_low	1147	0	int64
neutrophilCategorical_normal	1147	0	int64
serumLevelsOfWhiteBloodCellCategorical_Low	1147	0	int64
serumLevelsOfWhiteBloodCellCategorical_Normal	1147	0	int64
serumLevelsOfWhiteBloodCellCategorical_high	1147	0	int64
serumLevelsOfWhiteBloodCellCategorical_low	1147	0	int64
serumLevelsOfWhiteBloodCellCategorical_normal	1147	0	int64
lymphocytesCategorical_High	1147	0	int64
lymphocytesCategorical_Low	1147	0	int64
lymphocytesCategorical_Normal	1147	0	int64
CTscanResults_Inconclusive	1147	0	int64
CTscanResults_Neg	1147	0	int64
CTscanResults_Non_Presente	1147	0	int64

Table 5: Analisi Dataset finale

Column	Non-Null Count	Null Count	Dtype
CTscanResults_Pos	1147	0	int64
CTscanResults_neg	1147	0	int64
XrayResults_Neg	1147	0	int64
XrayResults_Non_Presente	1147	0	int64
XrayResults_Pos	1147	0	int64
Diarrhea_No	1147	0	int64
Diarrhea_Yes	1147	0	int64
Fever_No	1147	0	int64
Fever_Yes	1147	0	int64
Coughing_No	1147	0	int64
Coughing_Yes	1147	0	int64
SoreThroat_No	1147	0	int64
SoreThroat_Yes	1147	0	int64
NauseaVomitting_No	1147	0	int64
NauseaVomitting_Yes	1147	0	int64
Fatigue_No	1147	0	int64
Fatigue_Yes	1147	0	int64
RenalDisease_No	1147	0	int64
RenalDisease_Yes	1147	0	int64
diabetes_No	1147	0	int64
diabetes_Yes	1147	0	int64

Table 6: Analisi Dataset finale

Come possiamo osservare la maggior parte delle colonne rappresenta valori ***"int64"***, mentre per due colonne del dataset i dati rappresentati assume valore ***"float64"***.

2.6 Analisi della variabile dipendente

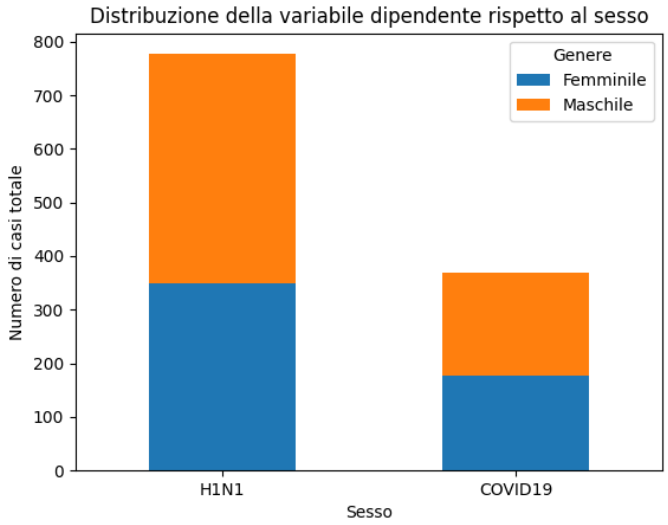
Durante questa fase verranno mostrate e analizzate le varie caratteristiche e correlazioni tra la variabile dipendente e le altre feature presenti all'interno del dataset.

2.6.1 Variabile dipendente

La variabile dipendente del dataset, ossia la variabile da predire, è riportata all'interno del dataset stesso con la label "D". Il valore assunto dalla variabile è *"0"* nel caso pazienti malati di *"H1N1"*, *"1"* nel caso di pazienti malati di *"COVID-19"*.

2.6.2 Distribuzione della variabile dipendente rispetto alcune feature

Di seguito verranno riportati alcuni grafici rappresentati della distribuzione della variabile dipendente rispetto alcune delle feature presenti nel dataset, in modo tale da meglio comprendere la struttura e la distribuzione dei dati nelle varie classi.



Da questo grafico possiamo osservare come i pazienti nel dataset siano maggiormente di sesso maschile (*620 records*) che di sesso femminile (*527 records*), ed essi si distribuiscono maggiormente sulla classe *"H1N1"* (*777 records totali*) che sulla classe *"COVID19"* (*370 records totali*).

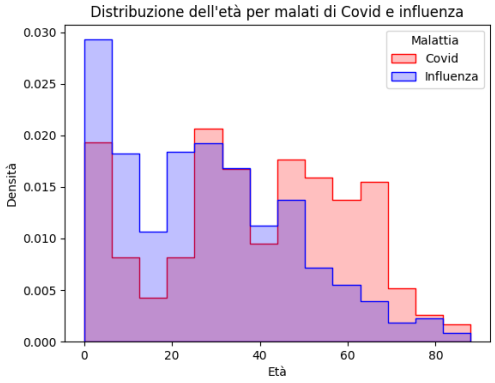


Figure 1: Distribuzione età

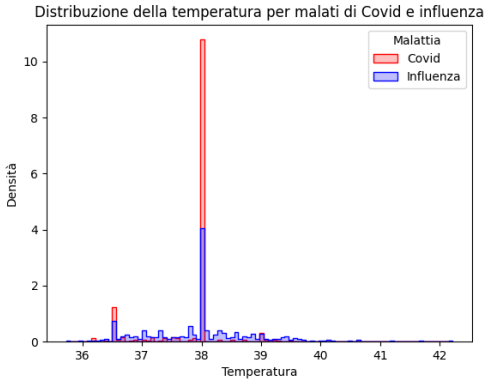


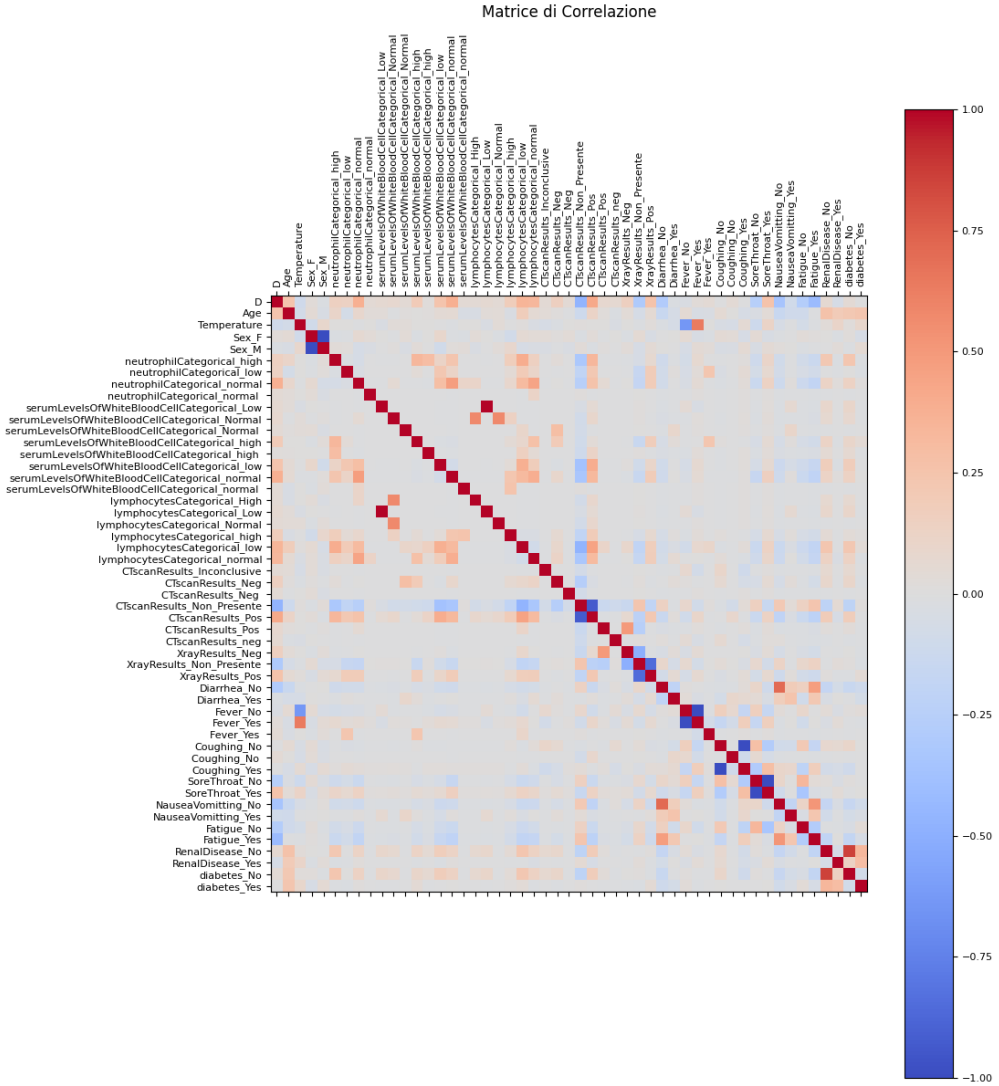
Figure 2: Distribuzione temperatura

Figure 3: Grafici delle distribuzioni rispetto la variabile dipendente

Come possiamo osservare dai grafici riportati la maggior parte dei pazienti che hanno riscontrato l'influenza risultano avere un età compresa tra gli *"0-20 anni"*, mentre la maggior parte delle persone che hanno riscontrato il Covid-19 risultano avere un età compresa tra i *"30-65 anni"*, e la maggior parte dei pazienti per entrambe le classi risulta avere una temperatura di *"38.0 gradi"*.

2.7 Matrice di correlazione delle variabili

Di seguito possiamo osservare la matrice di correlazione delle variabili, uno strumento statistico utilizzato per misurare le correlazioni tra due variabili in un dataset.



La matrice ci permette di capire e osservare i pattern e le relazioni tra le varie feature, in modo tale da capire e scegliere in maniera efficace le feature migliori per la classificazioni e quali invece danno informazioni ridondanti. La heatmap è stata creata utilizzando la libreria *"matplotlib.pyplot"*.

2.8 Bilanciamento delle classi

Dopo aver analizzato in generale il dataset possiamo ora verificare la distribuzione dei valori nelle due classi presenti nel dataset ***"COVID19"*** e ***"H1N1"***, ove *"H1N1"* indica l'influenza, in modo tale da evitare eventuali differenza in termini di numero di records tra una classe e l'altra che potrebbe portare il modello a potenziali bias nelle predizioni.

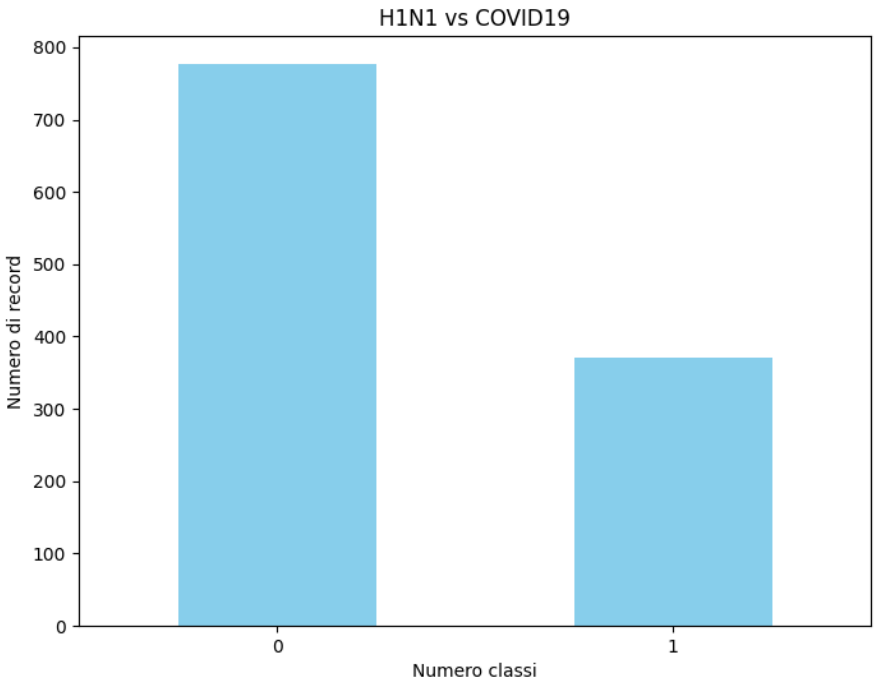


Figure 4: Grafico bilanciamento classi.

Come mostrato dal grafico il numero di records rappresentativi della classe *"H1N1"* (*777*, mostrati sull'asse x all'indice *"0"*) risultano essere in quantità superiore rispetto ai records rappresentativi della classe *"COVID19"* (*370*, mostrati sull'asse x all'indice *"1"*).

2.8.1 Oversampling o Undersampling?

Il problema principale di avere un dataset con classi sbilanciate è che il modello potrebbe presentare bias rispetto la classe minoritaria, portando a favorire la classe maggioritaria durante le predizioni. Ci sono varie strategie per gestire questa problematica, le due principali sono ***"l'oversampling"*** e ***"l'undersampling"***.
L'oversampling consiste nella creazione di nuove righe della classe minoritaria partendo dalle righe appartenenti alla medesima classe già presenti nel dataset, tuttavia, essendo i dati generati dall'oversampling dati *"artificiali"*, ciò che si rischia applicando *"oversampling"* in maniera eccessiva è la poca rappresentatività dei dati, come affrontato anche nella sezione *1.3.* L'undersampling invece permette di ridurre la dimensionalità di una classe in modo tale da ribilanciare la distribuzione stessa delle classi, portando a scegliere una sotto-popolazione da una popolazione iniziale e mantenendo la rappresentatività dei dati rispetto la realtà, tuttavia il problema principale che un approccio di undersampling porta è, in base all'approccio scelto, la creazione di dati viziati condizionati dalle carrateristiche scelte per selezionare i records sottogrruppo della classe maggioritaria oppure dal fattore randomico che potrebbe selezionare dati poco rappresentativi o casi limite. Il dataset utilizzato per questo report presenta uno sbilanciamento di **407 records**, ragionevolmente è stato ritenuto quindi opportuno applicare entrambi i metodi per mantenere quanto più possibile i dati fedeli rispetto ai dati reali. I metodi verranno applicati solo al set di training, dato che il validation test deve avvicinarsi ai dati del mondo reale e non avrebbe quindi senso utilizzare anche dati sintetici per la validazione.

2.8.2 L'importanza della rappresentatività dei dati

Le metodologie applicate sono la tecnica **"SMOTE"** per effettuare l'oversampling, e l'undersampling **"randomico"** per effettuare l'undersampling. La scelta nasce dal volontà di tenere quanto più possibile i dati puri, senza dover utilizzare eccessivamente dati artificiali o perdere informazioni importanti. Utilizzando le tecniche per ribilanciare il dataset porta inevitabilmente alla creazione di bias che possono portare un modello a fare predizioni errate, è giusto quindi capire e progettare bene le varie fasi, in modo tale da non generare problemi di overfitting e underfitting del modello. Per le ragioni sopra esposte è stato quindi ritenuto opportuno utilizzare entrambi gli approcci, per cercare di contenere la creazione di dati viziati o discostanti dalla realtà. I records di differenza tra classe maggioritaria e minoritaria nel training set verranno per il *60%* gestiti tramite undersampling e per il *40%* tramite oversampling.

2.8.3 Applicazione delle metodologie: Undersampling randomico

Utilizzando la tecnica dell'*"undersampling randomico"* sono stati selezionati campioni iniziali appartenenti alla classe maggioritaria, in modo tale da avere una differenza del *60%* inferiore rispetto a prima. Di seguito verrà mostrato il grafico del bilanciamento delle classi dopo aver eseguito la tecnica dell'*undersampling randomico*.

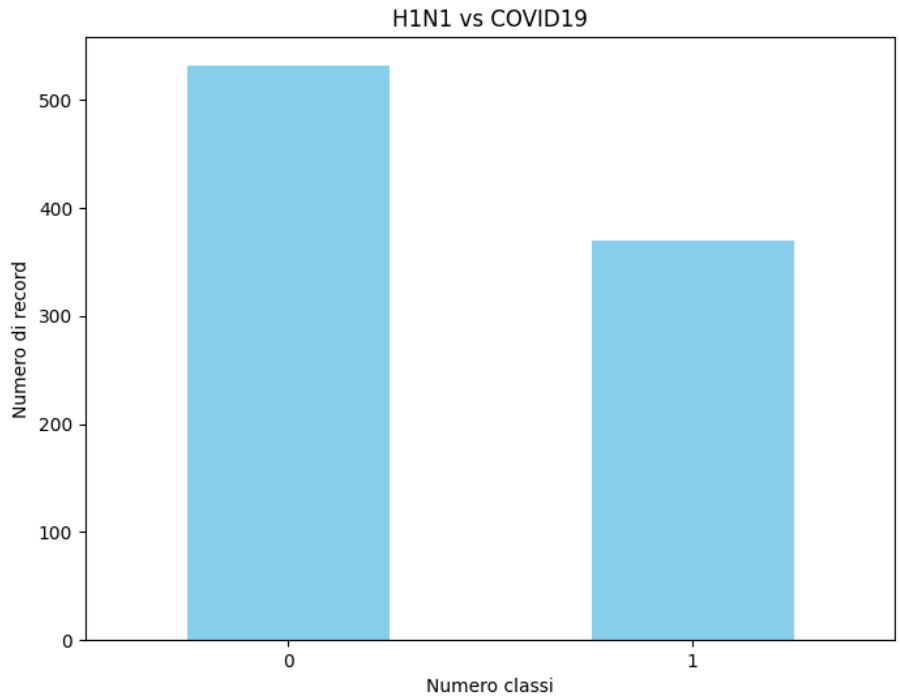


Figure 5: Grafico bilanciamento classi (UNDERSAMPLING).

La differenza tra la classe maggioritaria e la classe minoritaria ora risulta essere del solo 40% dei records.

2.8.4 Applicazione delle metodologie: SMOTE

Per bilanciare in modo efficace il dataset, senza perdere informazioni e senza perdere la rappresentatività stessa dei dati, applichiamo ora la tecnica ***"SMOTE"*** che utilizza la distanza euclidea per valutare e creare nuovi dati artificiali che nascono dalla perturbazione dei dati originali. Di seguito verra mostrato il risultato dell'applicazione dell'Oversampling SMOTE, realizzato utilizzando la libreria *"imblearn.over_sampling"*.

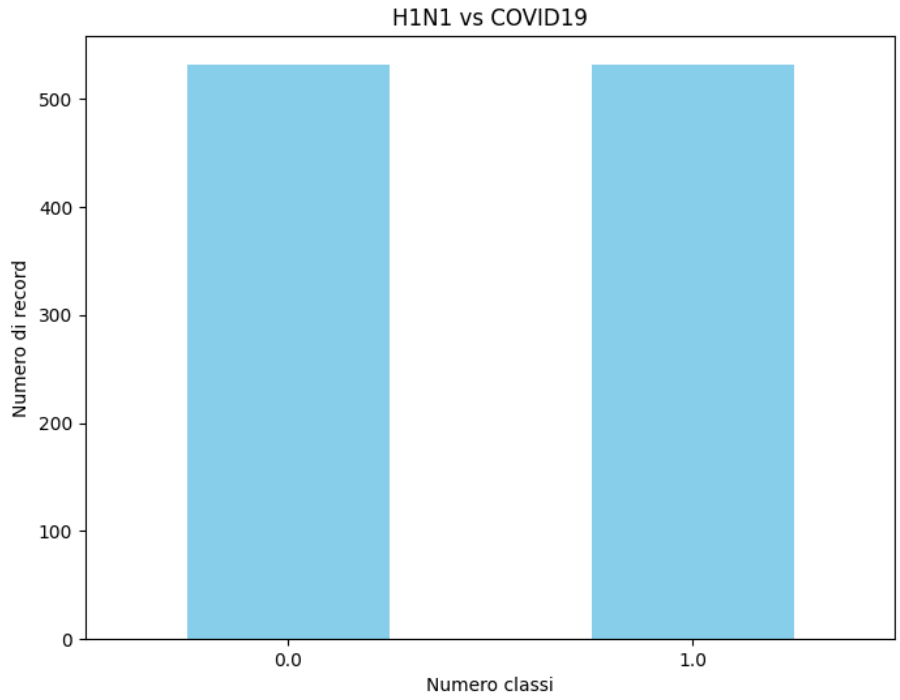


Figure 6: Grafico bilanciamento classi (OVERSAMPLING).

Come possiamo osservare dalla distribuzione dei valori, ora il dataset risulta avere gli stessi records sia per la classe *"H1N1"* che per la classe *"COVID19"*.

2.9 Normalizzazione del dataset

2.9.1 Encoding delle variabili categoriche

Come già visto nel paragrafo *2.4.5.*, il problema principale del dataset era la presenza di molti valori nulli, e data la presenza anche di molte variabili categoriche, in fase di aggiustamento dei valori nulli e successivamente durante la fase di encoding delle variabili è stato deciso di codificare le variabili categoriche utilizzando la codifica one-hot e l'utilizzo delle variabili *"dummy"*. Questa fase è stata necessaria sia per cercare di rimediare ai dati mancanti nel dataset, sia perchè i modelli di Machine Learning lavorano con valori numerici. Per la predizione è stato scelto che verrà utilizzata la colonna *"D"*, nella quale i valori riportati saranno "1" se positivo al *"Covid-19"* e "0" se positivo all'influenza *"H1N1"* o comunque non positivo al *"Covid-19"*.

2.9.2 Normalizzazione delle variabili continue

Per evitare potenziali bias e rendere anche le variabili confrontabili con le altre variabili è necessario, in fase di preprocessing dei dati, normalizzare i valori che rappresentati dalle variabili continue in modo tale da ridurli alla stessa "scala". Una variabile continua potrebbe rappresentare i dati in maniera differente rispetto un'altra variabile continua, questa maniera differente implica anche che le scale sulla quale sono rappresentate possono essere diverse. Per evitare bias ed errori dovuti alla differente scala di rappresentazione è utile quindi portarli ad un valore che sia confrontabile, rischiando tuttavia perdita di informazioni. La tecnica utilizzata per effettuare la normalizzazione delle variabili continue nell'ambito di questo report è quella della ***"Normalizzazione Min-Max"***, operazione che porta le variabili continue ad essere rappresentate in un intervallo deciso. Nel caso di questo report l'intervallo è *"0 - 1"*. La normalizzazione è stata eseguita grazie alla funzione *MinMaxScaler()* contenuta nella libreria *sklearn.preprocessing*.

2.10 Data Splitting

Ultima fase della preparazione e valutazione dei dati è la fase del data splitting, ossia la fase in cui si vanno a definire sia il set di dati di training del modello, sia il set di dati di testing del modello. Per eseguire la divisione del dataset è stata seguita la regola del *67/33*, il 67% del dataset sarà utilizzato per l'addestramento del modello, mentre il restante 33% del dataset sarà utilizzato per la fase di testing del modello.

3 Sviluppo del modello

3.1 Quale classificatore scegliere?

Dato che il task preso in esame è un task di classificazione e dato che il fine ultimo di questo progetto è quello di migliorare un approccio già descritto in letteratura, i classificatori scelti per eseguire il task preso in esame sono:

1. ***Decisional Tree***
2. ***Random-Forest***

La scelta dei due approcci è stata guidata sia dal fatto che nello studio in letteratura preso in esame vengono utilizzati i due approcci sopra elencati, sia guidata da uno degli obiettivi fissati per il miglioramento dell'approccio definito in letteratura, ossia il miglioramento dell'explainability del modello implementato.

3.2 Come valutare i modelli?

Per la valutazione dei modelli finali sono state scelte le varie metriche studiate durante il corso, in particolare l'***"Accuracy"***; la ***"Precision"***; il ***"Recall"*** e il ***"F1-Score"***. Per ogni modello verrà mostrata la ***matrice di confusione*** e ognuno di essi verrà testato su ciscuno dei due dataset (con bilanciamento e senza bilanciamento).

3.3 Come migliorare l'explainability?

Per il miglioramento dell'explainability del modello è stato scelto di riportare nelle sezioni successive i vari grafici rappresentanti la ***Feature Importance***, i valori dell'impatto delle feature calcolato utilizzando la teoria ***SHapley Additive exPlanations (SHAP)***, e l'implementazione di un prototipo di una possibile interfaccia del sistema finale.

3.4 Implementazione

Per l'implementazione del sistema, delle metriche di valutazione e per il caricamento dei dati sono state utilizzate le principali librerie utili al Machine Learning, tra cui:

1. ***"sklearn"***: per la creazione dei modelli e il preprocessing dei dati.
2. ***"pandas"***: per il caricamento del dataset e per la creazione e la gestione dei dataframe.
3. ***"imblearn"***: per il bilanciamento del dataset.
4. ***"numpy"***: per la gestione dell'imputazione sulle variabili continue.
5. ***"matplot"*** *e* ***"seaborn"***: per la creazione dei grafici.

Altri dettagli più specifici sull'implementazione del modello possono essere visualizzati nei file sorgenti python allegati a questo report.

3.5 Quale DataSet usare per l'addestramento?

Per addestrare i modelli è stato ritenuto opportuno utilizzare sia il dataset bilanciato che il dataset non bilanciato, in modo tale da avere una panoramica completa delle prestazioni dei modelli sia in presenza di classi bilanciate che non bilanciate.

4 Training e Valutazione

4.1 Metrica per il training

Per l'implementazione dei modelli è stato utilizzato il criterio della *"Gini Impurity"*. La Gini Impurity è una metrica che misura il grado di impurità delle classi presenti nel dataset e va a determinare il tasso di mescolanza delle classi prese in esame, un valore di impurità basso indica un set di dati puro.

4.2 Decision Tree

Per la creazione dell'albero decisionale è stato scelto di utilizzare il valore di profondità massimo dell'albero facendo un'analisi sulle prestazioni del modello sul set di training e sul set di validazione in funzione della variazione della profondità. Di seguito il grafico rappresentante come variano le prestazioni del modello in base alla profondità.

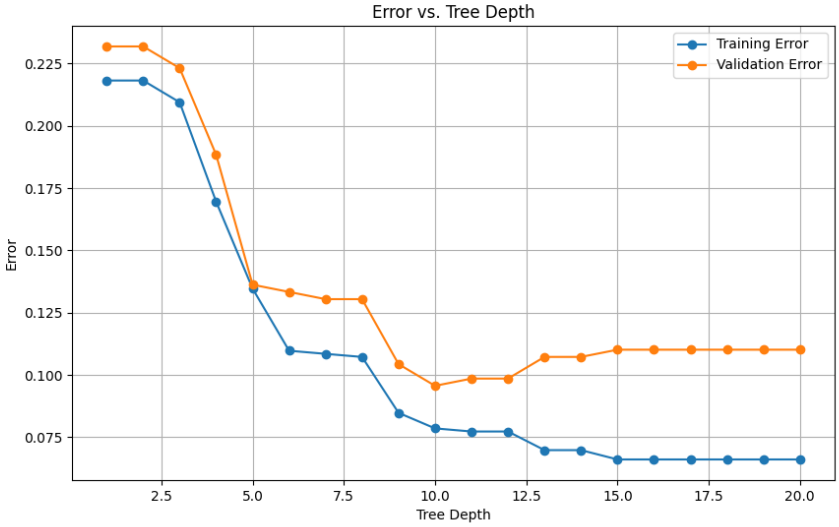


Figure 7: Aumento dell'errore all'aumentare della profondità dell'albero

Come possiamo notare prima che il modello vada in overfitting si raggiunge la profondità 10.

Di seguito verrà invece mostrato il medesimo grafico utilizzando il dataset bilanciato.

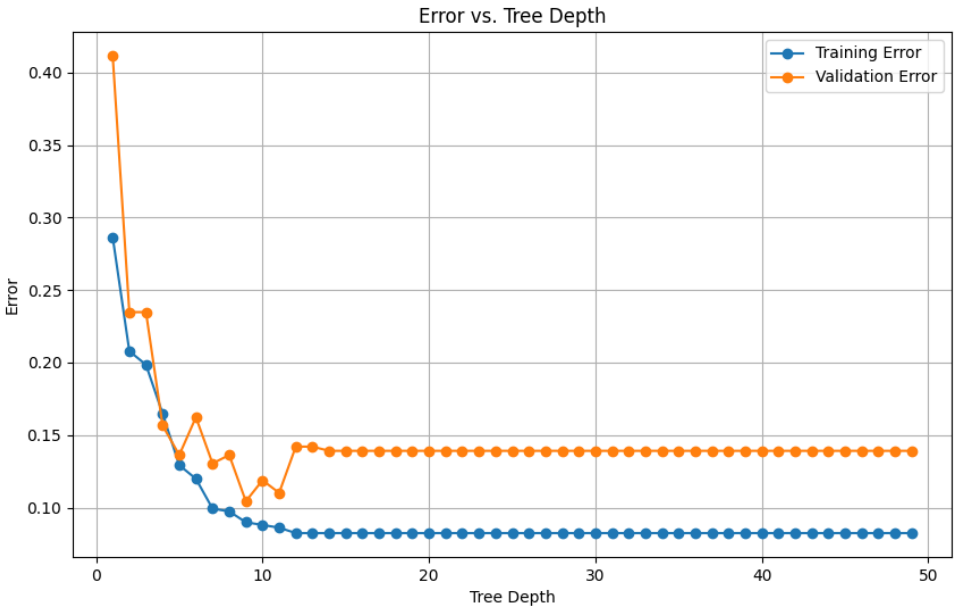


Figure 8: Aumento dell'errore all'aumentare della profondità dell'albero

Come possiamo notare in questo caso prima che il modello vada in overfitting si raggiunge la profondità 11, tuttavia la profondità migliore risulta essere 9.

4.2.1 Valutazione del Decision Tree

Per la valutazione del modello sono state utilizzate le metriche studiate durante il corso, in particolare di seguito verranno riportate la ***"matrice di confusione"*** e la ***"ROC Curve"***, oltre che i valori di ***"Accuracy"***; la ***"Precision"***; il ***"Recall"*** e il ***"F1-Score"***, sia per il modello addestrato usando il dataset sbilanciato che per il modello addestrando utilizzando la versione bilanciata del medesimo dataset.

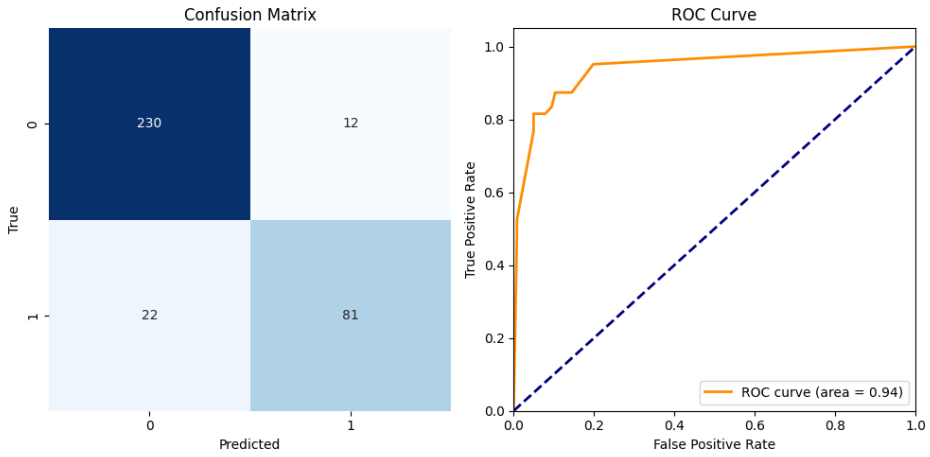


Figure 9: Grafico con dataset sbilanciato

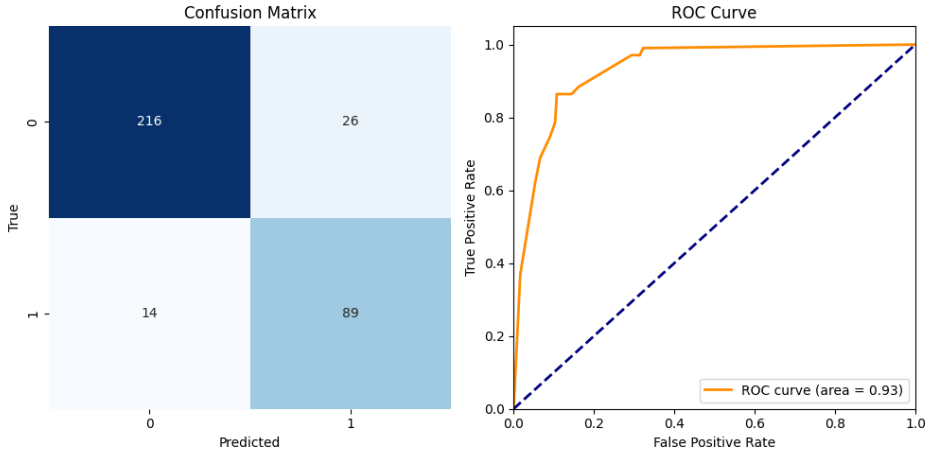


Figure 10: Grafico con dataset bilanciato

Di seguito sono riportate le prestazioni del modello senza bilanciamento del dataset:

1. ***Accuracy***: 0.90
2. ***Precision***: 0.87
3. ***Recall***: 0.79
4. ***F1 Score***: 0.83

Di seguito sono riportate le prestazioni del modello con bilanciamento del dataset:

1. ***Accuracy***: 0.88
2. ***Precision***: 0.77
3. ***Recall***: 0.86
4. ***F1 Score***: 0.82

Come mostrato dai valori il modello risulta avere accuratezza maggiore con il dataset non bilanciato, questo accade perchè probabilmente il set di valutazione ha una distribuzione differente della classe minoritaria, quindi il moodello risulta essere meno preciso.

4.3 Random Forest

Anche per l'implementazione del random forest è stato scelto di utilizzare il valore di profondità massimo facendo un'analisi sulle prestazioni del modello sul set di training e sul set di validazione in funzione della variazione della profondità. Di seguito il grafico rappresentante come variano le prestazioni del modello in base alla profondità utilizzando il dataset sbilanciato.

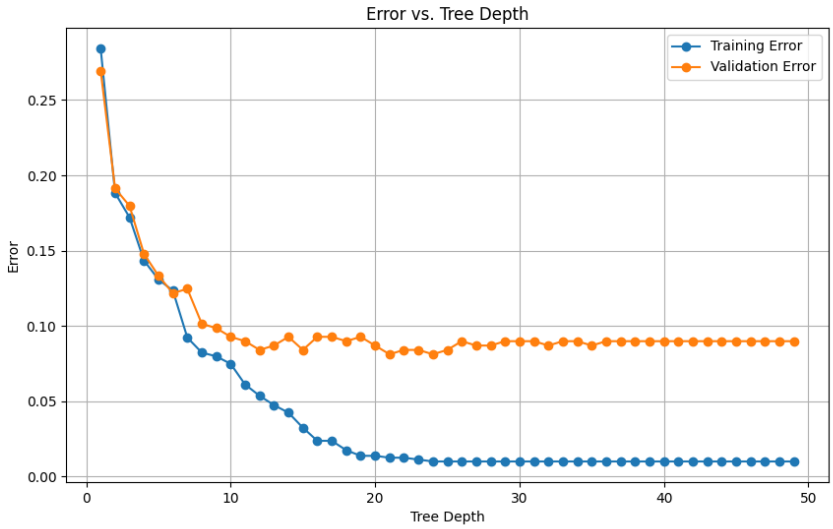


Figure 11: Aumento dell'errore all'aumentare della profondità dell'albero

Come possiamo osservare il valore migliore di profondità prima che il modello vada in overfitting è 15.

Di seguito il grafico rappresentante come variano le prestazioni del modello in base alla profondità utilizzando il dataset bilanciato.

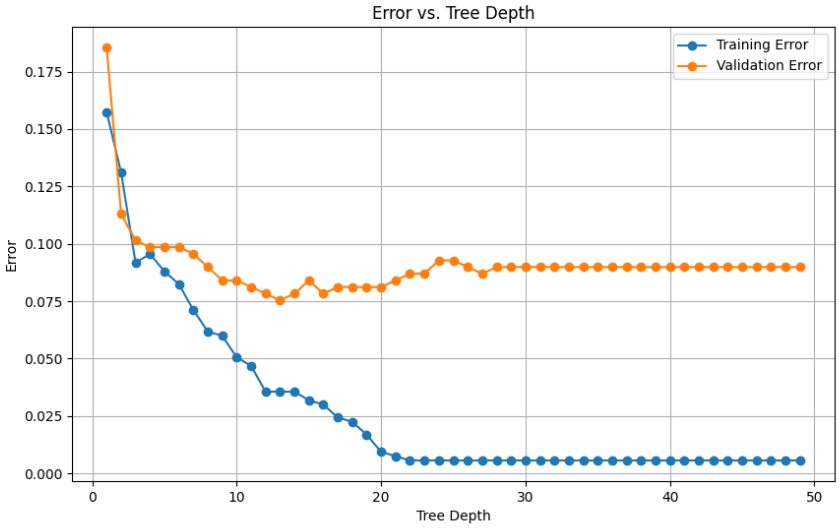


Figure 12: Aumento dell'errore all'aumentare della profondità dell'albero

Come possiamo osservare il valore migliore di profondità prima che il modello vada in overfitting è 13.

4.3.1 Valutazione del Random Forest

Per la valutazione del modello sono state utilizzate le metriche studiate durante il corso, in particolare di seguito verranno riportate la ***"matrice di confusione"*** e la ***"ROC Curve"***, oltre che i valori di ***"Accuracy"***; la ***"Precision"***; il ***"Recall"*** e il ***"F1-Score"***, sia per il modello addestrato usando il dataset sbilanciato che per il modello addestrando utilizzando la versione bilanciata del medesimo dataset.

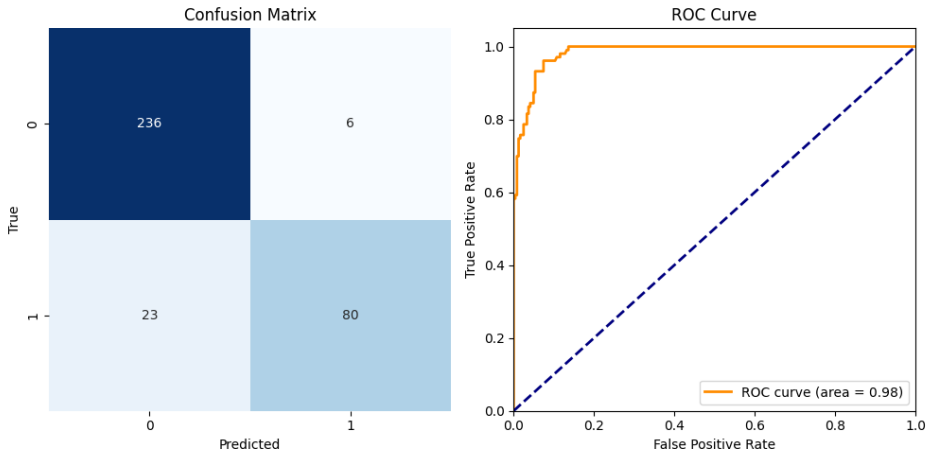


Figure 13: Grafico con dataset sbilanciato

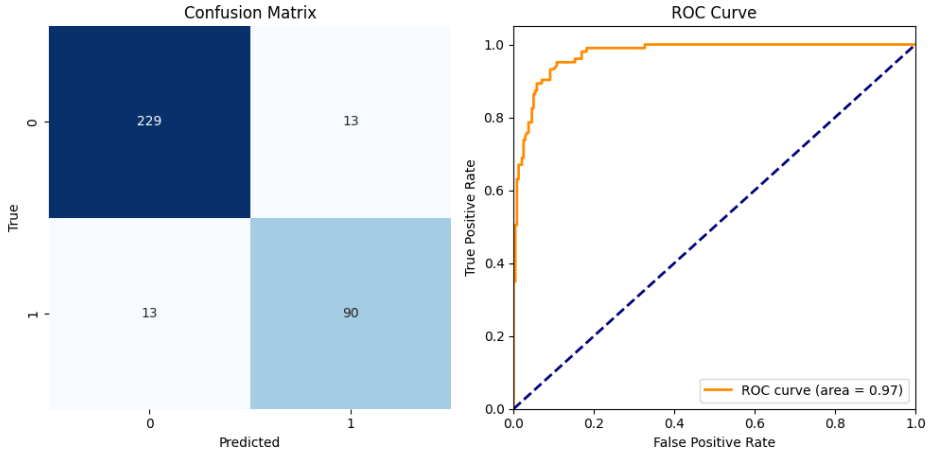


Figure 14: Grafico con dataset bilanciato

Di seguito sono riportate le prestazioni del modello senza bilanciamento del dataset:

1. ***Accuracy***: 0.92
2. ***Precision***: 0.93
3. ***Recall***: 0.78
4. ***F1 Score***: 0.85

Di seguito sono riportate le prestazioni del modello con bilanciamento del dataset:

1. ***Accuracy***: 0.92
2. ***Precision***: 0.87
3. ***Recall***: 0.87
4. ***F1 Score***: 0.87

Come mostrato dai valori in questo caso il modello risulta avere la stessa accuratezza, con valori differenti per le altre metriche di valutazione.

5 Explainability dei modelli implementati

5.1 Cos'è l'explainability e perchè migliorarla?

L'explainability è la metrica che misura quanto si riesce a spiegare il funzionamento di un algoritmo di IA, in particolare l'obiettivo del miglioramento dell'explainability è quello di rendere chiari e intuitivi i motivi per la quale un modello di IA ha dato un certo output e come mai ha preso una data decisione. Esiste anche un'altra metrica, chiamata Interpretability, che si concentra sulla spiegazione interna di algoritmo di IA. Migliorare sia l'explainability che l'interpretability risulta quindi fondamentale per l'implementazione di un modello che agisce in ambito medico, maggiori approfondimenti sono già stati fatti nelle sezioni *1.4.1* e *1.4.2*. Uno degli obiettivi principali posti come scopo finale di questo progetto è di cercare di migliorare l'aspetto di explainability dei modelli implementati rispetto a quelli visti in letteratura, quindi nello studio scelto come riferimento.

5.2 Come spiegare le predizioni dei modelli implementati

Esistono diversi metodi per migliorare l'explainability di un modello di ML. Per il seguente report è stato ritenuto opportuno di riportare per ogni modello implementato il grafico rappresentante la ***Feature Importance***, i valori dell'impatto delle feature calcolato utilizzando la teoria ***SHapley Additive exPlanations (SHAP)***, e l'implementazione di un prototipo di una possibile interfaccia del sistema finale.

5.2.1 Feature Importance

La feature importance è una metrica che valuta quanto una feature ha impattato sui risultati di una predizione, un punteggio più alto di importance indica che una feature ha più peso sulla predizione finale rispetto ad un'altra feature con importance più bassa. La feature importance permette dunque di capire come la variabile dipendente sia collegata con le altre feature, e fa capire anche quale variabile ha avuto un peso maggiore sulla predizione, questo aiuta a migliorare l'interpretabilità di un modello e soprattuto la spiegabilità delle predizioni.

5.2.2 SHapley Additive exPlanations (SHAP)

SHAP è una metrica, che si basa sulla teoria dei giochi, utilizzata per spiegare la predizione di un modello di Machine Learning. Essa viene calcolata come somma dell'impatto di ogni feature sulla predizione finale, lo scopo è riuscire a ricostruire e a spiegare in che modo il modello ha prodotto una data predizione.

5.2.3 Creazione di un prototipo dell'interfaccia

Utilizzare strumenti e metriche quali Feature Importance e SHAP porta ad un aumento della explainability, tuttavia senza un interfaccia UI del sistema usabile e che segua i principi fondamentali dell'UI/UX; che sia user-friendly e che permetta all'utente di capire e spiegarsi in che modo l'output del modello è stato generato, che faccia da tramite tra il modello e l'utente finale, non si avrebbe modo di rendere il sistema spiegabile e interpretabile a chi il sistema poi dovrà usarlo nella vita reale. Per questo è stato ritenuto opportuna l'implmentazione di un prototipo di un interfaccia usabile che permetta a tutti di poter capire e spiegare le predizioni dei modelli implementati.

5.3 Applicazione delle metriche ai modelli sviluppati

Di seguito verranno mostrate le varie metriche analizzate applicate ad ogni modello implementato.

5.3.1 Feature Importance: Decision Tree

Di seguito verrà riportato il grafico rappresentante la feature importance del Decision Tree con dataset bilanciato.

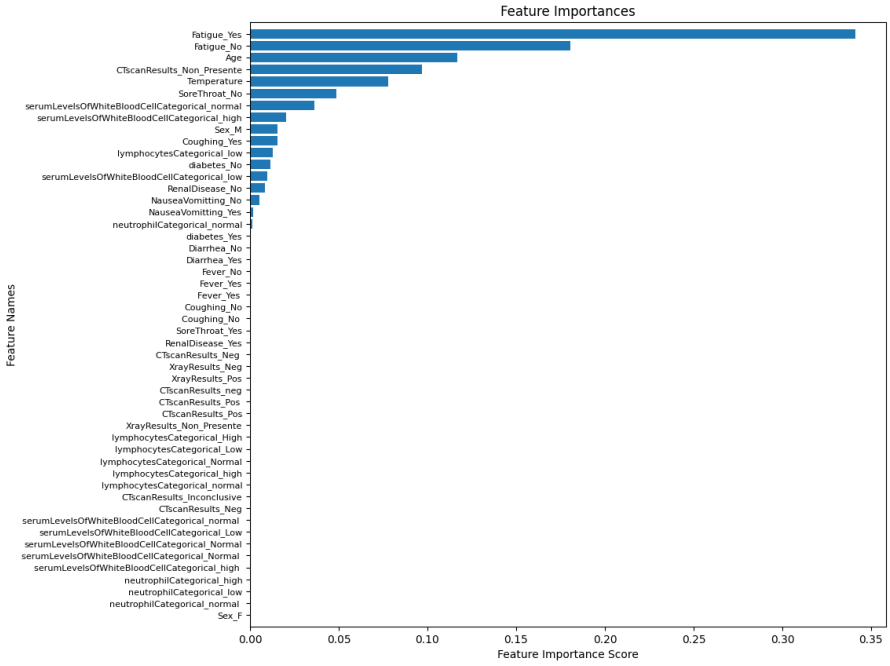


Figure 15: Feature Importance Decision Tree

Come possiamo osservare per il modello di Decision Tree con dataset bilanciato la feature con *"Importance Score"* più alto è risultata essere *"Fatigue_Yes"*.

5.3.2 Feature Importance: Decision Tree

Di seguito verrà riportato il grafico rappresentante la feature importance del Random Forest con dataset bilanciato.

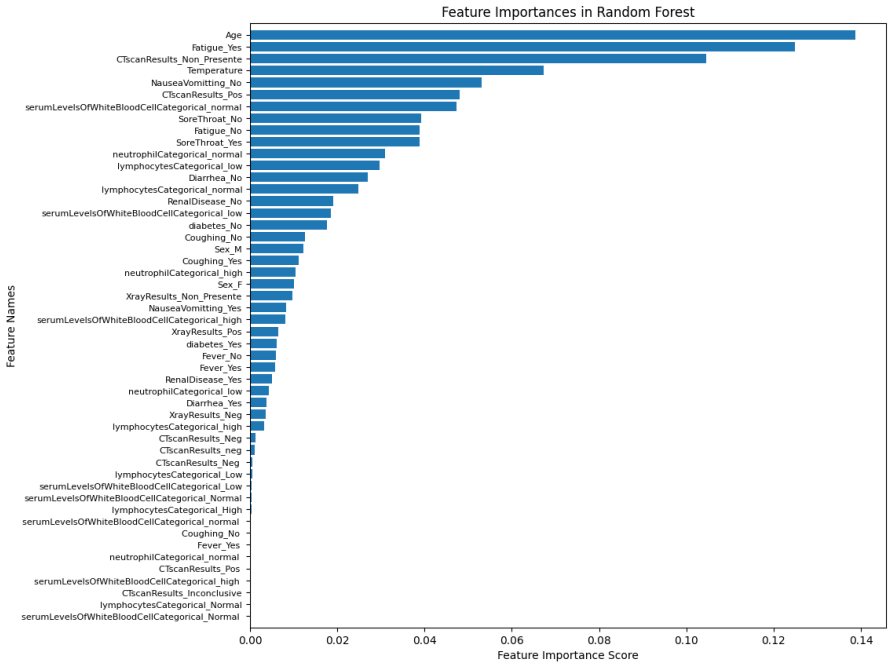


Figure 16: Feature Importance Random Forest

Come possiamo osservare per il modello di Random Forest con dataset bilanciato la feature con *"Importance Score"* più alto è risultata essere *"Age"*.

5.3.3 SHAP: Decision Tree

Di seguito verrà mostrata la feature importance calcolata tramite la teoria SHAP per il modello di Decision Tree.

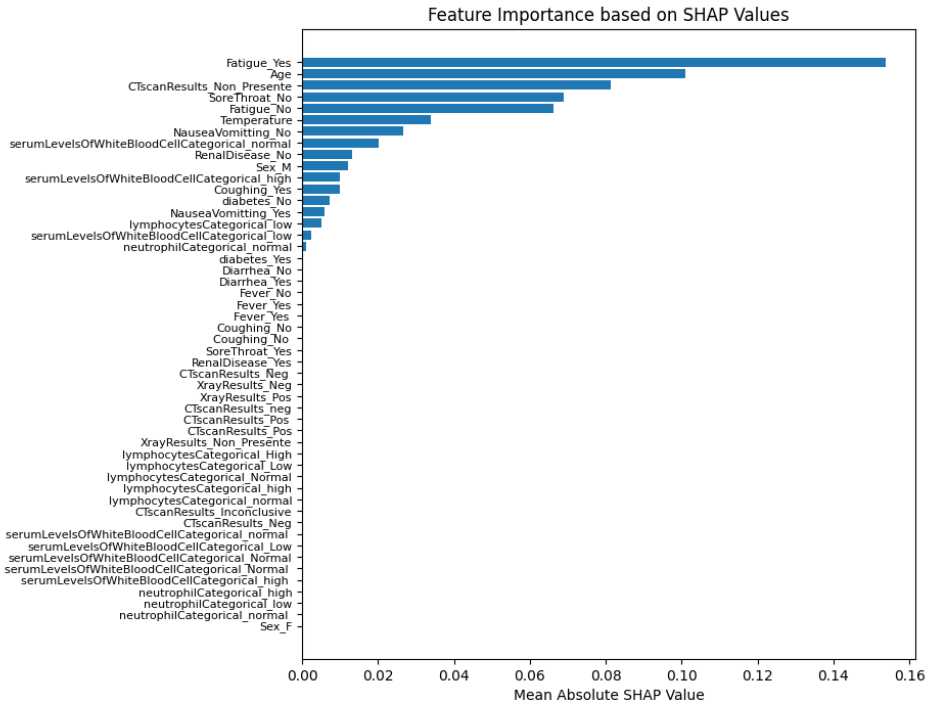


Figure 17: SHAP Decision Tree

Come possiamo notare il contributo principale alle predizioni viene dato dalla feature *"Fatigue_Yes"* e dalla feature *"Age"*.

5.3.4 SHAP: Random Forest

Di seguito verrà mostrata la feature importance calcolata tramite la teoria SHAP per il modello di Random Forest.

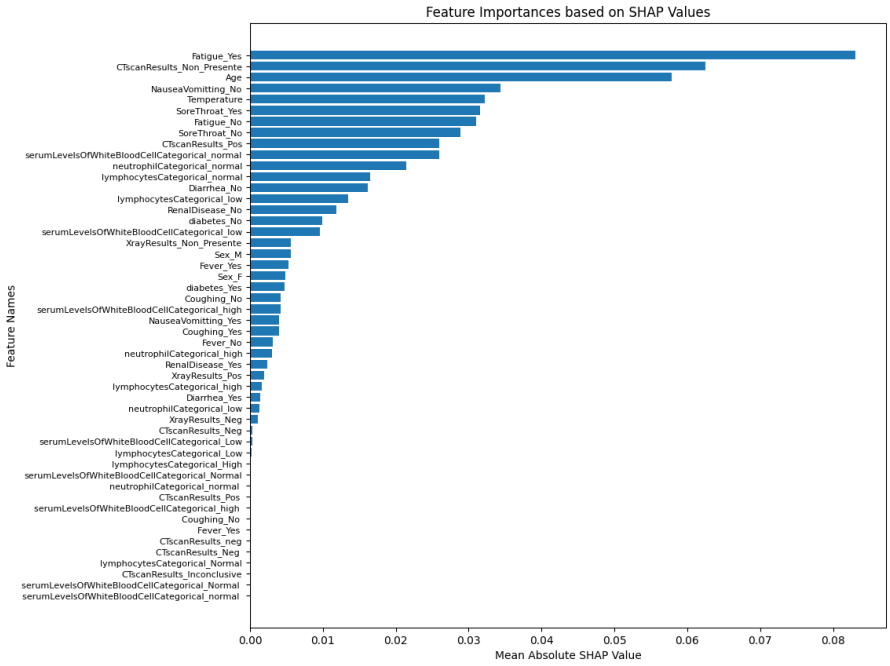
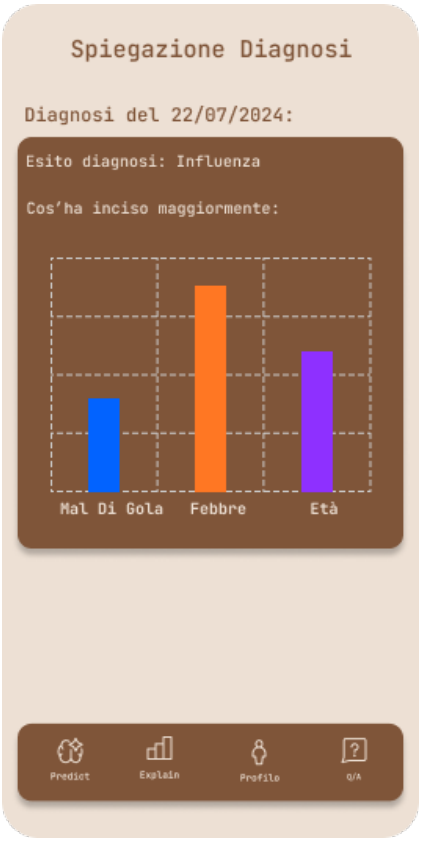


Figure 18: SHAP Decision Tree

Come possiamo notare anbche in questo caso il contributo principale alle predizioni viene dato dalla feature *"Fatigue_Yes"*.

5.4 L'interfaccia grafica del modello

Per il miglioramente dell'explainability del modello è stato ritenuto opportuno la crazione di un prototipo di un'interfaccia semplice e usabile, che riporti i risultati del modello e aiuti la comprensione dei risultati. Di seguito verrà riportato il prototipo.



6 Conclusioni

6.1 I problemi principali

L'obiettivo iniziale del progetto era quello di partire da un modello definito in letteratura e, dopo averne individuato i problemi principali, migliorarne le criticità o comunque provare a farlo. Dopo un analisi inziale dello studio nella quale sono stati evidenziati grandi problemi relativi ai dati presenti nel dataset e problemi legati alla spiegabilità del modello, è cominciata l'attività progettuale incentrata proprio sulla risoluzione di questi problemi. I dati utilizzati nella ricerca sono dati risalenti alla prima ondata, periodo in cui il covid era ancora in fase di studio e le cause della malattia ancora non erano chiare, nel dataset iniziale erano presenti molti dati mancanti e altri discriminatori.

6.2 Soluzione ai problemi riscontrati

Il lavoro principale è stato fatto sui dati, in quanto risultavano essere sporchi e pieni di potenziali bias sulle predizioni. Purtroppo a causa della scarsità dei dati reali e non sintetici di quegli anni è stato difficile non cadere comunque in bias, principalmente all'assenza di molti dei sintomi che negli anni si sono rivelati impattanti sulla trasmissione della malattia. Il problema dei dati sintetici è una delle cause principali che non permettono l'ampio utilizzo di modelli di Macine Learning nella medicina. Altro lato a cui questo progetto ha prestato attenzione è la spiegabilità delle predizioni, cosa che sopratutto in ambito medico risulta essere fondamentale. Grazie agli strumenti offerti da python è stato molto semplice dare una spiegazione alle predizioni del modello, e grazie all'utilizzo di *"Figma"*, tool utilizzato per la prototipazione di interfacce, è stato semplice creare un prototipo ideale di un interfaccia che permetta di spiegare bene i risultati dei modelli.

6.3 Risultati

I risultati del lavoro svolto sono due modelli con accuracy del 89% e del 90%, risultati comunque molto bassi per quanto riguarda ai in ambito medico perchè anche il 10% può comunque portare alla diagnosi di una malattia piuttosto che un'altra ad un paziente. Il lavoro svolto sui dati ha portato buoni risultati sulla qualità dei dati, con basso utilizzo di dati sintetici per l'addestramento dei modelli. Il lavoro svolto sull'explainability invece mi ha permesso di ampiamente capire quali fossero i problemi principali delle predizioni del modello. In generale i risultati ottenuti sono molto positivi, in quanto hanno permesso di utilizzare molti dei concetti studiati durante il corso.