

UNIVERSITÀ DEGLI STUDI DEL SANNIO

Dipartimento Di Ingegneria

Corso di Laurea in Ingegneria Informatica

Analisi e validazione di algoritmi di classificazione su un dataset di pazienti Parkinsoniani

RELATORE:
Prof.ssa **Lerina Aversano**

CANDIDATO:
Federico Stocchetti
Matr:863001333

ANNO ACCADEMICO 2020/2021

Indice

Elenco delle figure	ii
Elenco delle tabelle	iii
Introduzione	iv
1 Contesto e problem statement	1
1.1 Informatica medica	1
1.2 Data mining	3
1.3 Machine Learning	5
1.4 Parkinson	7
1.5 Problem Statement e stato dell'arte	8
2 Dati e approccio al problema	11
2.1 Dataset utilizzato	11
2.2 Fasi di lavoro	13
3 Setup e metodologia di lavoro	15
3.1 Linguaggio e librerie	15
3.1.1 Librerie di Python utilizzate	16
3.2 Preparazione dei dati	16
3.3 Bilanciamento del DataFrame	18
3.4 Feature selection	19
3.5 Feature scaling	24
3.6 Scelta dei classificatori	25
3.7 Scelta dei parametri e classificazione	28
3.8 Ensemble di classificatori	29
4 Analisi dei risultati	32
4.1 Risultati utilizzando tutte le feature	32
4.2 Risultati applicando feature selection a varie threshold	34
4.3 Risultati applicando tecniche di bilanciamento e normalizzazione	36
4.4 Ensemble di classificatori e risultati finali	38
4.5 Analisi conclusiva e minacce alla validità del modello	39

INDICE	i
5 Conclusione e sviluppi futuri	40
Ringraziamenti	42
Bibliografia	43

Elenco delle figure

1.1	Sorgenti dei dati utilizzati	10
3.1	Utilizzo di SMOTE a partire dalla libreria imblearn	19
3.2	Esempio di dendrogramma	22
3.3	Script per la generazione e il taglio del dendrogramma	23
3.4	Selezione di una feature dai cluster di features	23
3.5	Funzione di normalizzazione dei dati	25
3.6	Rappresentazione della diversa divisione dello spazio delle feature su un dataset di esempio	26
3.7	Classificatori prima del tuning dei parametri	27
3.8	Classificatori con il tuning dei parametri	28
3.9	Funzione per la creazione dei sets, la classificazione e la creazione di metriche di valutazione	29
3.10	Ranking dei classificatori in base all'accuratezza	30
3.11	Ensemble di classificatori	31
4.1	Ranking dei classificatori in base all'accuratezza sul dataset originale	33
4.2	Metriche di classificazione per l'algoritmo CatBoost	33
4.3	Tempo necessario per l'allenamento di GradientBoostingClassifier alle varie soglie di taglio delle features	35
4.4	Performance di Multi Layer Perceptron su un dataset non normaliz- zato e non bilanciato	36
4.5	Performance di Multi Layer Perceptron sul dataset normalizzato e bilanciato	37
4.6	Accuratezza dei classificatori sul dataset normalizzato e bilanciato . .	37
4.7	Performance complete dell'ensemble di classificatori	38

Elenco delle tabelle

3.1	Feature ricavate alle diverse soglie di correlazione.	24
4.1	Feature ricavate alle diverse soglie di correlazione.	34
4.2	Performance di tutti i classificatori dopo feature selection, normaliz- zazione, bilanciamento e tuning dei parametri	39

Introduzione

Nell'era moderna il progresso tecnologico nel settore dell'informatica coinvolge a cascata il progresso in molti settori adiacenti. In particolare nel settore medico gli avanzamenti tecnologici hanno reso possibili studi, diagnosi, operazioni chirurgiche e trattamenti di efficacia e affidabilità di gran lunga superiori al passato.

Nel campo della ricerca medica si sta sempre di più volgendo l'attenzione verso i Big Data sanitari per sviluppare nuove tecniche innovative di medicina di precisione, diagnosi automatica e sistemi a supporto delle decisioni mediche. Per queste ed altre applicazioni, che sfruttano i dati e i principi di Data Science per introdurre innovazione nel settore, sono utilizzati processi di Data Mining e tecniche di Machine Learning.

In questo lavoro di tesi verranno analizzati brevemente gli approcci utilizzati per la diagnosi automatica del morbo di Parkinson e verrà descritto un workflow di classificazione su un dataset reale di features vocali provenienti da pazienti affetti dalla malattia.

Saranno utilizzati metodi di Data Mining e Machine Learning per affrontare il processo di classificazione, con il fine ultimo di analizzare e validare gli output di questo processo.

Capitolo 1

Contesto e problem statement

1.1 Informatica medica

L'informatica medica è un campo molto ampio, che origina dalla gestione del flusso informativo proveniente da un paziente o cittadino, ma che si è evoluta nel tempo in una disciplina che è essenziale alle attività mediche di ricerca, trattamento del paziente e gestione dei processi medici.

I primi riferimenti all'informatica medica si hanno nella seconda metà del secolo scorso, momento in cui le discipline informatiche iniziano a contaminare vari ambiti per il supporto alla gestione dell'informazione. Qui si originano i primi processi per il supporto al mantenimento dei dati dei pazienti e al loro successivo utilizzo.

Negli anni successivi con l'avanzare dei progressi nel campo informatico si è iniziato a sviluppare i primi strumenti di supporto alle attività mediche come primi software per l'imaging tramite radiografia, strumenti per il design di protesi e software per la ricerca nella bibliografia medica, fino ad arrivare alla diffusione di internet su larga scala che ha permesso di connettere le "isole" di dati medici e di aumentarne così la fruibilità.

Oggi l'informatica medica è un campo di studi ancora in costante evoluzione, dai confini non ben definiti, i cui oggetti di studio non sono più limitati alla raccolta dei dati dei pazienti o al supporto gestionale dei processi medici, ma si espandono nel campo della ricerca farmacologica, della diagnosi assistita da processi automatici e della ricerca medica in generale.

Attualmente le applicazioni notevoli dell'informatica medica sono molteplici e si

sviluppano trasversalmente in vari sottocampi:

- CRI o Informatica per la ricerca clinica: ambito dell'informatica medica che mira a migliorare l'efficienza della ricerca clinica tramite l'utilizzo di metodi informatici, ad esempio tramite il supporto alla data collection per gli studi clinici o la creazione di pipeline per la ricerca data driven.
 - Bioinformatica : campo emergente dell'informatica medica in cui convergono informatica, statistica e biologia. Si basa sull'applicazione di tecniche informatiche alla sempre crescente quantità di dati biomedici e genetici per estrarre conoscenza e creare strumenti ad uso di personale medico ma anche pazienti. Vengono considerate popolazioni di pazienti, materiale genetico e dati sulle malattie per applicazioni come la diagnosi assistita, il miglioramento di terapie farmacologiche o il calcolo dei fattori di rischio per l'insorgenza di malattie.
 - Intelligenza Artificiale per la medicina: questo campo si concentra sull'utilizzo di intelligenza artificiale per migliorare i processi medici di diagnosi, trattamento del paziente e sviluppo farmaceutico. La ricerca in questo campo oggi si concentra principalmente sulla creazione di un sistema di supporto alle decisioni cliniche (CDSS) tramite l'utilizzo dei big data sanitari, sulla medicina personalizzata e sulla interpretazione di segnali biologici complessi, come il funzionamento del cervello.
Per via del forte riscontro che si ha applicando metodi di IA alla medicina questo è anche il settore dell'informatica medica che riceve più investimenti privati per la ricerca, con stakeholder del calibro di IBM, Microsoft, Google e Neuralink che competono per sviluppare soluzioni innovative per il mercato medico. In particolare Neuralink è impegnata nella creazione di una interfaccia cervello-macchina con l'obiettivo di un controllo diretto di protesi robotiche come braccia, occhi o orecchie.
 - Telemedicina: questo campo studia le tecniche per il controllo di pazienti a distanza o per permettere di svolgere processi medici di diagnosi e chirurgia in remoto. Questo settore di studi ricopre una importanza sempre crescente per il tentativo di disaccoppiare il luogo in cui si ricevono cure dal personale medico che lo esegue, aprendo nuove possibilità nei paradigmi di cura e monitoraggio
-

del paziente.

- Informatica per l'imaging medico: si occupa del processing delle immagini mediche e della creazione di modelli 3D partendo da rilevazioni effettuate tramite strumenti, allo scopo di assistere il personale medico nelle operazioni di diagnosi e chirurgia.

Rispetto ai primi anni, quindi, cambiano profondamente alcuni paradigmi di sviluppo, si passa dal conservare e distribuire in maniera ottimale i dati ad elaborarli per estrarre conoscenza, dal fornire supporto ai processi medici alla loro automatizzazione e innovazione.

1.2 Data mining

Il settore medico, sulla linea di un trend che accomuna molti altri settori, è sommerso dall'enorme quantità di dati prodotti che ammonta nel 2018 a circa 1218 EB e di cui si proietta una crescita del 36% fino al 2025 [2]. Per sfruttare a pieno le potenzialità che una così grande mole di dati offre per l'ambito medico si usano sempre più frequentemente tecniche di knowledge discovery o data mining, supportate da tecniche di machine learning supervisionato o non supervisionato.

"Data mining, also popularly referred to as knowledge discovery from data (KDD), is the automated or convenient extraction of patterns representing knowledge implicitly stored or captured in large databases, data warehouses, the Web, other massive information repositories or data streams" [3]

Le tecniche di data mining vengono quindi utilizzate per estrarre conoscenza, spesso in maniera automatica, dalle grandi quantità di dati presenti nei database di ospedali e cliniche o provenienti da ricerche mediche su media e larga scala.

Le più utilizzate sono, generalmente:

- Clustering: raggruppamento di oggetti o caratteristiche di un insieme in più sottoinsiemi di elementi coerenti tra di loro (clusters). Gli elementi di un cluster avranno caratteristiche in comune secondo le metriche prese in considerazione, ma al contempo ogni cluster avrà elementi dissimili dagli elementi degli altri
-

cluster

- Classificazione: assegnazione di una classe di appartenenza ad elementi di un insieme. Gli elementi di un insieme A vengono etichettati di classe C se possiedono, entro una certa varianza, le caratteristiche che definiscono la classe C. Le caratteristiche della classe possono essere ricavate per inferenza da un insieme di dati già classificati, spesso tramite metodi di machine learning.
- Regressione: stima della relazione tra una variabile dipendente, spesso l'output di un sistema, e una serie di variabili indipendenti. Il risultato di questo processo è un modello di regressione, capace di indicare il valore della variabile dipendente a partire da variabili indipendenti.
- Dependency modelling: individuazione di regole di associazione, regole logiche basate sull'analisi degli attributi degli elementi di un insieme secondo una serie di *if-then* statements. L'output di questo processo è una serie di regole di associazione per cui, dati due elementi X e Y si può asserire che X implica Y.
- Anomaly detection: individuazione degli outliers in un insieme di dati, ovvero di elementi che sono molto diversi dagli altri dello stesso insieme.

A prescindere dalla tecnica utilizzata, il processo di data mining può essere generalizzato nelle seguenti fasi di lavoro:

- Data collection and integration: raccolta dei dati rilevanti per l'analisi da eseguire dalle varie fonti e integrazione dei dati in un unico insieme di dati;
 - Data cleaning and preprocessing: rimozione delle inconsistenze nei dati, delle discrepanze dei valori e "pulizia" dei dati da valori errati o outliers.
 - Data selection and reduction: riduzione della dimensionalità dei dati e selezione dei dati da utilizzare rispetto ai dati raccolti. Questa fase è la precedente, che
-

possono sembrare banali, sono in realtà di vitale importanza poiché contribuiscono in maniera importante alla qualità dell'analisi che si andrà ad effettuare, secondo il principio di *"garbage in garbage out"*.

- Data transformation and feature reduction: in questa fase vengono selezionate le feature rilevanti dei dati da analizzare, poiché un numero minore di features riduce i tempi di elaborazione, la complessità dell'analisi e permette di produrre analisi più generali. I dati vengono poi trasformati tramite encoding e type conversion per essere accettati come input dell'algoritmo di analisi. In questa fase si possono operare anche discretizzazioni sui dati o normalizzazione in un intervallo ben definito di valori.
- Data mining: questa è la fase di lavoro in cui viene estratta la conoscenza dai dati. È essenziale scegliere l'algoritmo più adatto ai risultati che vogliamo ottenere, il formato dell'output e i parametri dell'algoritmo di analisi.
- Evaluation and validation: in questa fase avviene una review dei risultati dell'analisi per verificarne la correttezza e la validità. Potrebbe infatti verificarsi la situazione in cui l'analisi effettuata non abbia rilevanza statistica o semplicemente che sia una pessima analisi. Ad esempio potremmo avere un modello di classificazione che produce una accuratezza peggiore di una scelta casuale.

Nonostante questo sia considerato il workflow standard per il data mining esistono alcune variazioni che sono specifiche al campo di applicazione, come ad esempio il Cross Industry Standard Process for Data Mining (CRISP-DM), ma i *core elements* rimangono gli stessi. In questo lavoro di tesi verrà seguito il workflow prima descritto, con particolare enfasi sulla feature selection, essendo la dimensionalità degli attributi del dataset preso in considerazione molto alta.

1.3 Machine Learning

Il machine learning è una tecnica di apprendimento automatico utilizzata in molti domini operativi per risolvere problemi classici di predizione, classificazione e clustering. Il machine learning viene spesso associato all'Intelligenza Artificiale

ma i due concetti non sono perfettamente sovrapposti: il machine learning è una branca dell'intelligenza artificiale, in quanto indica una macchina che è capace di apprendimento automatico a partire da una sufficiente quantità di dati.

Gli algoritmi di machine learning puntano a costruire un modello predittivo o decisionale partendo da un set di dati, detto *training data*. Gli approcci all'apprendimento, quindi alla costruzione del modello, sono molteplici e per via della recente esplosione negli avanzamenti in questo campo alcuni sono oggetto di studio tutt'oggi e non sono consolidati come standard (ad esempio il *Self-supervised learning*). Gli approcci tipici al machine learning, consolidati ed utilizzati universalmente in quest'ambito sono:

- Reinforcement Learning: in questo approccio l'algoritmo decide autonomamente quali azioni o decisioni generano le ricompense maggiori, secondo un processo di *trial and error*. I componenti principali di questo approccio sono: l'agente, che prende le decisioni e impara, l'ambiente e le azioni eseguibili.

A differenza di altri approcci il RL si occupa di decisioni sequenziali, in cui l'azione da compiere dipende dallo stato attuale del sistema e ne determina quello futuro. Per via della sua natura il RL è spesso utilizzato nella teoria dei giochi, nella robotica, nella statistica e nei sistemi multi-agente.

- Unsupervised Learning: è un approccio al machine learning in cui all'algoritmo non vengono fornite etichette per il dataset di training. In questo modo il primo step sarà creare cluster di dati e ricavare attributi e pattern nei dati in maniera inferenziale, senza l'intervento umano, per poi procedere alla creazione del modello.

Per sua natura questo approccio trova applicazione dove è impossibile o non efficiente etichettare i dati manualmente, ad esempio nella Computer vision o Natural Language processing (NLP). Un esempio notevole di applicazione di Unsupervised Learning è GPT-2, modello di NLP sviluppato da OpenAI, addestrato tramite lo scraping di testp da 8 milioni di pagine web.

- Supervised Learning: approccio in cui l'algoritmo costruisce un modello matematico di un insieme di dati che contengono esplicitamente sia gli input che gli output desiderati. L'algoritmo impara così ad ottenere i risultati desiderati da esempi espliciti che mettono in correlazione degli input con tali risultati. Si utilizzano a questo scopo quindi dataset etichettati per risolvere una vasta
-

gamma di problemi, come regressione, predizione e classificazione.

La classificazione tramite classificatori può avvenire secondo vari algoritmi, che hanno diverse performance in base alla conformazione del problema e di come vengono trattati i dati.

Nello specifico si possono individuare tra i classificatori algoritmi basati su:

- Calcolo del gradiente;
- Regressione lineare o non lineare;
- Boosting;
- Calcolo della probabilità e legge di Bayes;
- Alberi di decisione;
- Reti neurali.

Da notare che gli algoritmi di boosting sono una categoria trasversale in quanto mettono insieme più *learners* che utilizzano diversi algoritmi in quello che è noto come *ensemble di classificatori*.

In questo lavoro di tesi verranno utilizzate prevalentemente tecniche di machine learning supervisionato per la classificazione binaria, nello specifico per individuare la presenza o l'assenza del morbo di Parkinson in un paziente.

1.4 Parkinson

Il morbo di Parkinson è una malattia neurodegenerativa che causa una lenta perdita di controllo nei movimenti e in un terzo dei casi anche danni cognitivi. È una condizione medica molto diffusa, la seconda malattia neurodegenerativa dopo l'Alzheimer per incidenza e colpisce

- Lo 0,4% delle persone di età inferiore ai 40 anni, 1 persona su 250;
 - L'1% delle persone di età superiore ai 65 anni, 1 persona su 100;
 - Il 10% delle persone di età superiore a 80 anni, 1 persona su 10.
-

L'età media dell'insorgenza della malattia è di circa 57 anni, ma ci sono forme di Parkinson giovanile caratterizzate da un esordio della malattia tra i 21 e i 40 anni. Mentre le cause dell'insorgenza della malattia sono ancora ignote ed oggetto di ricerca, sono note molte delle cause dei sintomi, dovuti alla degenerazione dei gangli basali.

I gangli basali sono aggregati di cellule nervose, situati in profondità nel cervello, che aiutano a coordinare i movimenti muscolari e sopprimere i movimenti involontari. Questo avviene tramite il rilascio di messaggeri chimici detti neurotrasmettitori, come la dopamina, che veicolano il messaggio tra le cellule nervose necessario per inviare un impulso, quindi un segnale, ai muscoli del corpo.

Con l'incorrere della malattia avviene la degenerazione dei gangli basali, che non riescono più a produrre dopamina e di conseguenza a generare gli impulsi per controllare il sistema muscolare, provocando i tipici sintomi associati a questa malattia come tremore, bradicinesia (movimenti lenti), perdita della coordinazione e dell'equilibrio, rigidità e debolezza muscolare. Oltre a questi sintomi troviamo insonnia, stipsi e problemi di deglutizione, ma anche sintomi mentali come depressione, demenza e allucinazioni. I sintomi insorgono gradualmente man mano che la degenerazione delle cellule cerebrali avanza, fino a portare inevitabilmente ad uno stato terminale.

Il Parkinson si presenta quindi come una malattia che coinvolge tutto il corpo, ma di cui sappiamo ancora molto poco. È infatti ignota la causa della malattia e al momento non esiste alcuna cura per i pazienti, il cui trattamento si limita alla fisioterapia e a farmaci per rallentare in parte l'avanzamento dei sintomi.

1.5 Problem Statement e stato dell'arte

Per via della natura della malattia, i pazienti affetti da morbo di Parkinson beneficiano da una diagnosi precoce, per essere un passo avanti alla malattia con i trattamenti[5]. Questo però non è sempre possibile: se è vero che nello stadio avanzato il Parkinson è facilmente diagnosticabile poiché i sintomi sono evidenti e tipici, è vero anche che in una fase iniziale molti dei sintomi sono lievi o quasi inesistenti, rendendo molto difficile il processo di diagnosi. La diagnosi è di tipo clinico, quindi avviene principalmente tramite valutazione dei movimenti del paziente e dei sintomi, arrivando al Parkinson tramite l'esclusione di altre malattie neurologiche.

È bene notare che questi stessi problemi di diagnosi si presentano anche nella forma avanzata della malattia, benché sia più evidente la sintomatologia, poiché non esiste attualmente esame o procedura di diagnostica per immagini che può confermare direttamente la diagnosi di Parkinson [4].

Per far fronte a questo problema il settore della ricerca si sta espandendo in nuovi approcci ed in particolare l'informatica medica si sta concentrando su tecniche di *automatic disease detection*, con particolare enfasi su procedure non invasive e telemedicina. Queste tecniche sono prevalentemente basate su algoritmi di machine learning, utilizzati per l'addestramento di un modello predittivo oppure di classificazione che impara da una grande quantità di dati quali sono i pattern da ricercare per individuare la presenza della malattia.

È bene notare che al momento della stesura di questa tesi non esiste un metodo definitivo per la diagnosi, ma esistono lavori di ricerca che pongono le basi per una sua successiva implementazione tramite l'approccio al problema da vari angoli.

Secondo uno studio condotto a Maggio 2021 infatti, sono 209 i tentativi di successo di implementare una diagnosi del Parkinson in maniera automatica [6] la cui totalità utilizza metodi di machine learning, con una accuratezza che varia tra l'85.6% e il 94.4% a seconda dell'approccio utilizzato.

È importante ricordare, nel considerare gli approcci utilizzati, che nello studio sopracitato sono analizzati solo i paper in lingua inglese e sono state escluse le pubblicazioni in altre lingue, al fine di facilitare il processo di text mining.

Gli studi presi in considerazione, per quanto simili nell'approccio alla diagnosi tramite machine learning, variano considerevolmente nel tipo di classificatore utilizzato e soprattutto nei dati di partenza utilizzati per il training, che possono essere:

- Registrazioni vocali, nel 26.3% degli studi;
 - Dati sui pattern di movimento, nel 24.4% ;
 - Dati sui pattern di scrittura o disegno a mano libera, nel 7.7% ;
 - Immagini ricavate tramite risonanza magnetica, nel 17.2% ;
 - Immagini ricavate tramite tomografia ad emissione di fotone singolo, nel 6.7%;
-

- Immagini ricavate tramite tomografia a emissione di positroni, nell'1.9% ;
- Combinazione dei precedenti, nell'8.6%.

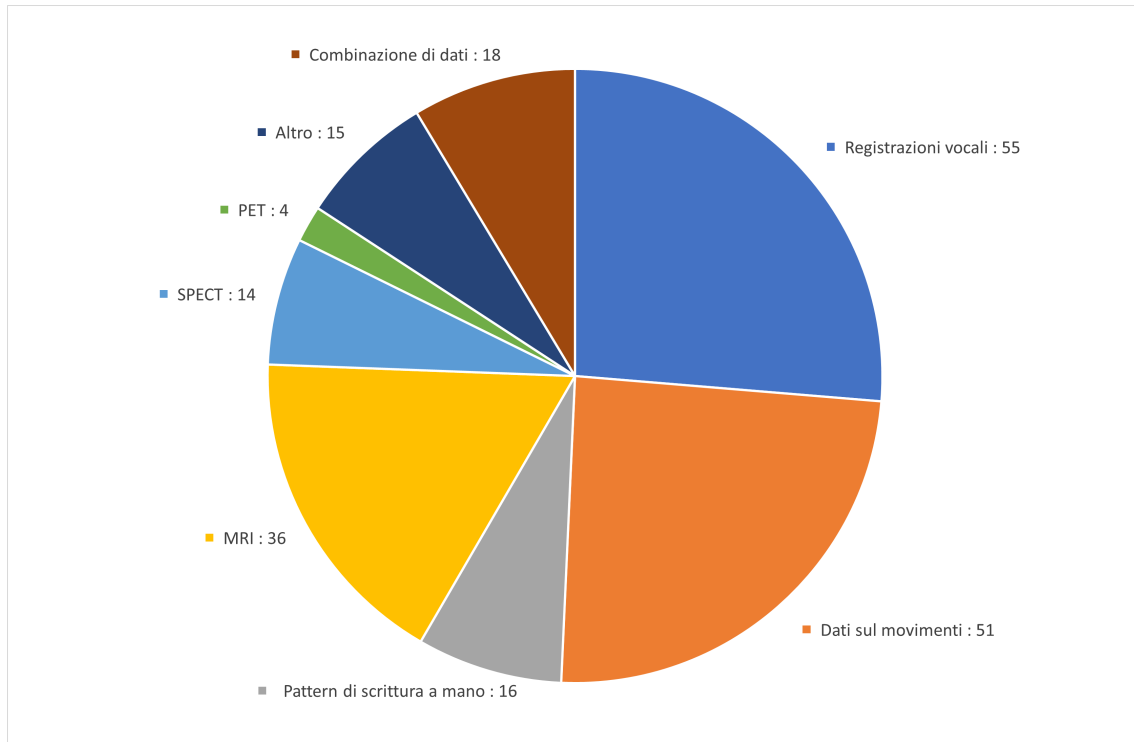


Figura 1.1: Sorgenti dei dati utilizzati

L'utilizzo di features vocali per il training di un algoritmo di machine learning può sembrare ad un primo impatto scorrelato dall'obiettivo di diagnosi del Parkinson. In realtà la malattia porta fin dagli stadi iniziali cambiamenti nella voce dei pazienti, in particolare sul tono, sul volume e sulle frequenze emesse, rendendo molto efficace questo metodo di valutazione

Capitolo 2

Dati e approccio al problema

Come riscontrato nello studio dello stato dell'arte per la diagnosi automatica del Parkinson, un approccio basato sull'analisi di feature vocali genera risultati notevoli in combinazione con l'utilizzo di classificatori. In questo lavoro di tesi verrà analizzato proprio questo approccio, su un dataset contenente feature vocali di pazienti Parkinsoniani.

2.1 Dataset utilizzato

I dati utilizzati provengono da un dataset donato alla University of California Machine Learning Repository da parte del Dipartimento di Neurologia dell'Università di Instambul. I dati sono stati raccolti a partire da 188 pazienti affetti da morbo di Parkinson, 107 uomini e 81 donne con età da 33 a 87 anni, e un gruppo di controllo di 64 individui sani, 23 uomini e 41 donne, con età che variano tra 41 e 82 anni. Le registrazioni vocali sono avvenute in ambiente controllato con un microfono impostato a 44.1 Khz. Ogni partecipante ha effettuato tre ripetizioni della vocale /a/ per un totale di 756 registrazioni vocali.

Il Dataset è composto dalle feature vocali estratte da tali registrazioni da parte degli autori del paper "A Comparative Analysis of Speech Signal Processing Algorithms for Parkinson's Disease Classification and The Use of The Tunable Q-Factor Wavelet Transform"[8], che hanno donato il dataset per pubblico utilizzo alla repository UCI.

All'interno del paper si può leggere come il criterio di estrazione delle feature vocali sia cambiato rispetto ad altri approcci, seguendo le indicazioni di un lavoro prece-

dente che individuava nuovi marker di tipo non lineare nei pattern del parlato dei pazienti parkinsoniani[9], individuabili tramite la trasformata del segnale definita come TQWT (Tunable q-factor Wavelet Transform).

Il dataset ha una dimensionalità molto elevata per via dell'elevato numero di feature estratte, sono infatti 755 gli attributi, categorizzabili in:

- Attributi del paziente: Identificativo del partecipante, genere e classe. Gli attributi genere e classe sono binari, variano tra 0 e 1 per indicare i due generi o l'assenza e la presenza della malattia nel partecipante;
 - Feature appartenenti alla baseline: 21 features che rappresentano le più usate all'interno degli studi su diagnosi basata su features vocali. Includono tremolio (jitter), frequenza fondamentale, parametri sulle armoniche (noise floor, Signal to Noise Ratio) e pitch;
 - Attributi per l'analisi in frequenza: sono in totale 11 e descrivono l'intensità del segnale vocale espressa in dB, le frequenze utilizzate e la larghezza di banda del segnale;
 - Misure sulle corde vocali: sono 22 e vengono stimate a partire dal segnale vocale e dei modelli. Includono parametri che indicano la quantità di rumore emessa dalle corde vocali, durata di apertura e chiusura della glottide e decomposizione empirica in segnali elementari;
 - MFCC: 84 parametri di analisi simili alle misure sulle corde vocali, ma specifiche per individuare come il parkinson modifica il tratto vocale;
 - Features della trasformata del segnale senza TQWT: 182 attributi;
 - Features appartenenti alla TQWT: 432 attributi.
-

Gli autori dello studio che ha pubblicato i dati hanno quindi preso in considerazione una vasta gamma di parametri ai fini dell'analisi, aggiungendo oltre a quelli considerati in altri studi ben 432 parametri appartenenti alla TQWT.

2.2 Fasi di lavoro

Volendo riformulare brevemente l'obiettivo del lavoro che segue, si cercherà di diagnosticare correttamente il morbo di Parkinson solo tramite le informazioni ricavate da un segnale vocale e si metteranno a confronto i vari metodi che è possibile utilizzare per farlo.

Volendo calare questo approccio nel mondo reale possiamo dire che, con una quantità molto maggiore e più diversificata di dati, ad esempio includendo anche pazienti che hanno malattie diverse dal parkinson che influenzano la voce, il modello sviluppato potrebbe essere integrato in un sistema di diagnosi automatica.

Si deve inoltre ricordare che essendo la diagnosi di tipo clinico, ovvero basata sui sintomi, un tool che utilizzi questo modello non si deve in alcun modo sostituire al personale medico nel processo di diagnosi, ma deve soltanto assisterlo nella decisione.

La creazione del modello si presenta quindi come una task di classificazione o predizione, in quanto si può predire il valore dell'attributo classe (presenza o assenza della malattia) ed è bene a questo punto analizzare gli step che serviranno a raggiungere l'obiettivo di una corretta classificazione e predizione.

Il primo step sarà un lavoro preliminare sui dati, volto alla preparazione degli stessi per il processo di classificazione. Ci si deve assicurare che i dati siano privi di errori, completi e in un formato conveniente per l'analisi. Bisogna inoltre ridurre la dimensionalità dei dati.

In un secondo step saranno valutati i classificatori più adatti allo scopo e ne saranno ottimizzati i parametri di classificazione.

Come ultimo passo viene effettuata l'analisi dei classificatori, per capire come migliorarne le performance e quale è opportuno utilizzare per avere i migliori risultati. Per la valutazione sono utilizzate le metriche standard relative ai modelli di classificazione:

- Precision: indica appunto la precisione della predizione ed esprime il rapporto

tra il numero di output corretti e il numero di output totale. In altre parole indica quante delle nostre predizioni sono corrette rispetto al totale delle predizioni effettuate. Può essere formalizzata, per la classificazione binaria, come:

$$\frac{truepositives}{truepositives + falsepositives}$$

- Recall: per la classificazione binaria indica il rapporto tra il numero di predizioni positive corrette il totale delle predizioni possibili positive. Indica, in altre parole, quante delle istanze di una classe abbiamo individuato rispetto al totale. Può essere formalizzata come:

$$\frac{truepositives}{truepositives + falsenegatives}$$

- Accuracy: Indica il rapporto tra il numero di predizioni corrette e il totale delle predizioni ed è una delle metriche più usate per la valutazione dei modelli. Viene definita come:

$$\frac{truepositives + truenegatives}{truepositives + truenegatives + falsepositives + falsenegatives}$$

- F-1 score: misura dell'accuratezza di un test, che rappresenta la media armonica tra precision e recall. È definito come:

$$2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Grazie all'utilizzo di queste metriche verrà quindi valutato il processo di classificazione e sarà effettuata una analisi dei risultati forniti dai modelli.

Capitolo 3

Setup e metodologia di lavoro

In questo capitolo si mostrerà il procedimento operativo utilizzato per il raggiungimento dei risultati finali. In generale il processo si può astarre ad alto livello come composto di soli tre step:

- Pre-elaborazione dei dati;
- Training dei classificatori;
- Testing e analisi dei risultati.

3.1 Linguaggio e librerie

Il linguaggio di programmazione prevalentemente utilizzato in questo lavoro è Python, con una piccola sezione in R.

Questa scelta si allinea con quello che è oggi lo standard nel campo della Data Science, in cui si utilizzano principalmente Python ed R per via di vari fattori, primo su tutti la grande quantità di risorse e letteratura a disposizione per le task più comuni. L'utilizzo di Python rende infatti immediato l'utilizzo di librerie per il caricamento e la visualizzazione dei dati, la manipolazione di strutture dati e l'esecuzione di task di Data Mining e Machine Learning, mettendo al contempo a disposizione dello sviluppatore tutto quello che è necessario per una programmazione sia procedurale che object oriented.

3.1.1 Librerie di Python utilizzate

Nella scrittura dello script di classificazione sono state utilizzate le seguenti librerie Python:

- Pandas: libreria fondamentale per caricamento, manipolazione e analisi dei dati all'interno dell'ambiente di lavoro. Tramite questa libreria possiamo leggere dati da un gran numero di sorgenti in diversi formati, nel caso di questo lavoro in formato CSV.

I dati letti in questo modo sono incapsulati in strutture dati di tipo Series, vettori di dati con un indice associato, o DataFrame, strutture bidimensionali che riflettono la composizione di una tabella indicizzata, con colonne e righe accompagnate da un indice. Le colonne di un DataFrame sono a loro volta delle Series che condividono lo stesso indice per l'ordine degli elementi.

I dati letti possono essere manipolati o analizzati in vari modi tramite operazioni sulle strutture dati che le contengono. Ad esempio possiamo riempire i valori nulli di un DataFrame, convertire i tipi degli oggetti o effettuare operazioni di filtraggio per recuperare un subset dei dati.

- Sklearn: appartenente al progetto Scikit-learn, questa libreria open source contiene algoritmi di apprendimento automatico e machine learning. All'interno del pacchetto sono presenti algoritmi di classificazione, regressione, clustering, macchine a vettori di supporto (SVM), classificazione di tipo bayesiano e probabilistica e algoritmi di k-mean regression e clustering. Questa libreria è progettata per essere utilizzata con le altrettanto note NumPy e SciPy, ed è spesso utilizzata in combinazione con i tool messi a disposizione da Pandas.

3.2 Preparazione dei dati

Per una corretta analisi bisogna verificare che i dati siano di buona qualità, poiché seguendo il principio di *garbage in garbage out* se la qualità dei dati in input è scadente, si otterrà un modello poco significativo per via degli effetti di rumore e dati mancanti.

In questa fase viene seguito il workflow relativo al Data Mining analizzato nel capitolo precedente. Ricordiamo che è composto dalle seguenti fasi:

- Data collection and integration;
- Data selection and reduction;
- Data cleaning;
- Data transformation and preprocessing.

Nella fase di data collection sono stati analizzati i dataset presenti su repository open source come UCI Machine Learning Repository e Kaggle. Sono stati trovati 10 datasets contenenti feature vocali di pazienti parkinsoniani e gruppi di controllo, ma una prima analisi ha rivelato la presenza di dati duplicati e ripubblicati, portando il numero effettivo di datasets disponibili sull'argomento a 4.

I 4 datasets trovati presentano caratteristiche molto differenti: tre di essi infatti fanno riferimento ad un numero esiguo di pazienti (<40) e contengono meno di 100 istanze, mentre quello rimanente contiene i dati di 252 partecipanti con 3 istanze per partecipante, quindi un totale di 756 istanze nel dataset. Le differenze però si trovano anche nelle features utilizzate e nella metodologia: la similitudine tra le features estratte dal segnale vocale tra i vari datasets è quasi zero, con differenze sostanziali sia nelle feature in sé sia per i parametri di estrazione di feature parametrizzate, ad esempio le trasformazioni in frequenza.

A porre un ulteriore ostacolo all'integrazione è la metodologia di acquisizione dei dati, per cui in due dei quattro datasets si sono utilizzati segnali vocali provenienti dalla stessa frase, ripetuta una volta per paziente, mentre negli altri due si è utilizzato rispettivamente la vocale /a/ e una diversa frase. Analizzando bene il contenuto dei datasets quindi, risulta impossibile un corretto processo di integrazione dei dati, poiché se anche fanno tutti riferimento allo stesso tipo di studio, sono stati acquisiti in maniera troppo diversa per essere messi insieme nella costruzione di un modello.

A fronte di questa analisi viene scelto il dataset con il maggior contenuto informativo, il file "pd_speech_features.csv" proveniente dal paper "A Comparative Analysis of Speech Signal Processing Algorithms for Parkinson's Disease Classification and The Use of The Tunable Q-Factor Wavelet Transform"[8], in quanto contiene 756 istanze di 252 partecipanti diversi e 755 features estratte. Questo dataset è stato raccolto con la precisa intenzione di effettuare questo studio, quindi analizzando i

valori troviamo che nessuno di essi è mancante e non ci sono ridondanze (istanze uguali) o dati chiaramente sbagliati, nel limite dell'analizzabile.

Prima di procedere con le successive fasi dell'analisi, viene esclusa la colonna 'id' del dataset, che contiene l'identificativo del partecipante. Sono presenti nel dataset 3 istanze per partecipante e includere questa colonna nell'analisi darebbe una informazione troppo diretta al classificatore, indicando che se una istanza del paziente X appartiene alla classe 1 (Parkinsoniano) allora anche le altre due istanze vi appartengono. Questo dato quindi toglie importanza all'analisi sulle features vocali e crea una discrepanza con un utilizzo reale, dove l'id di un nuovo paziente non ricopre un parametro di decisione per la diagnosi.

Vengono inoltre controllati i tipi di valori nelle colonne, poiché potrebbe esserci un mismatch che altera la classificazione, con ad esempio attributi numerici interpretati come categoriali. Utilizzando la funzione `print(data.dtypes)` viene stampata una Series contenente tutti i tipi, mostrando che non è presente alcun tipo di mismatch.

3.3 Bilanciamento del DataFrame

Il dataset risulta a questo punto completo per tutti i valori e pulito da errori, ma presenta una caratteristica che può inficiare pesantemente la creazione del modello, ovvero i dati non sono bilanciati. Questo significa che nel dataset sono presenti più istanze di una classe rispetto all'altra e quindi i classificatori allenati con questi dati avranno un *bias* verso la classe più numerosa all'interno del training set.

Per ovviare a questo problema esistono vari approcci, che dividiamo in tre principali categorie:

- **Oversampling:** tecnica in cui si cerca di incrementare artificialmente il numero di istanze della classe meno rappresentata tramite la creazione di istanze sintetiche appartenenti alla stessa classe. Al termine della procedura le classi avranno lo stesso numero di istanze. Il tradeoff in questa tecnica è che per quanto si possano raggiungere risultati molto migliori senza perdere informazioni sul dataset (senza rimuovere istanze), si introducono però nuove informazioni replicando alcune istanze della classe meno rappresentata. Tali informazioni potrebbero non riflettere la realtà e peggiorare il modello oppure essere poco rilevanti e migliorarlo con il solo effetto di bilanciare i dati.
-

- Undersampling: tecnica in cui si cerca di diminuire le istanze della classe/i più numerosa rimuovendone alcune dal dataset casualmente o seguendo un criterio specifico. Questo porta ad avere un numero uguale di istanze delle varie classi a discapito del contenuto informativo presente nelle istanze rimosse.
- Class weighting: alle classi viene assegnato un peso, con la classe più numerosa che ha un peso minore della classe meno numerosa. Dando un peso alle varie classi possiamo bilanciare l'analisi proveniente dal dataset senza manipolare le istanze.

In questo lavoro verrà utilizzato una particolare tecnica di oversampling detta SMOTE (*Synthetic Minority Oversampling Technique*).

Questa tecnica genera nuove istanze della classe di cardinalità minore a partire dal dataset di partenza. A differenza di altre tecniche di oversampling che possono introdurre overfitting, questa tecnica minimizza il problema inserendo valori nelle feature delle istanze che non cambiano la distribuzione dei valori nel totale. In altre parole non viene aggiunto contenuto informativo che non sia già presente nel dataset di partenza.

```
if balance_data:
    smote = SMOTE('not majority', random_state=1)
    x_train, y_train = smote.fit_resample(x_train, y_train)
    x_test, y_test = smote.fit_resample(x_test, y_test)
```

Figura 3.1: Utilizzo di SMOTE a partire dalla libreria imblearn

3.4 Feature selection

Una seconda problematica da affrontare prima di passare alla fase di training dei classificatori riguarda la feature selection. Le istanze presenti nel dataset utilizzato infatti hanno una alta dimensionalità per quanto riguarda gli attributi e questo, per quanto ci fornisca molti parametri di decisione può inficiare il processo di classificazione in vari modi.

Una alta dimensionalità dei dati porta a due problemi principali:

- Aumento del costo computazionale: la grande quantità di features presenti porta ad allungare considerevolmente i tempi di addestramento e utilizzo del modello, che deve tenere in considerazione molti più parametri.

Puramente come esempio possiamo allenare il classificatore GradientBoostingClassifier con l'intero dataset senza una riduzione dimensionale, impiegando un tempo di 71 secondi. Sulla stessa macchina è stato allenato lo stesso modello, utilizzando solo un subset rilevante delle features in soli 11 secondi, con risultati di performance del modello comparabili. Il modello che utilizza tutto il feature set ha quindi impiegato un tempo del 645% maggiore pur avendo risultati operativi simili.

Bisogna tenere in considerazione anche che se 71 secondi sembra un tempo, in valore assoluto, non esagerato, questo divario va ad aumentare man mano che vengono aggiunte istanze nel dataset, quindi in un contesto operativo dove si opera con questo stesso workflow ma un numero molto maggiore di dati non effettuare alcun tipo di feature selection porta grandi limitazioni in termini di scalabilità della soluzione.

- Overfitting: Un modello allenato con una grande quantità di features può risultare troppo specifico e non generalizzabile, rendendolo inaccurato quando un nuovo set di dati viene proposto. Questo avviene poiché non tutti gli attributi sono significativi al fine della nostra analisi; alcune features non portano contenuto informativo per la task che stiamo svolgendo e si comportano all'atto pratico come rumore, anche se non lo sono.

Un esempio può essere un attributo che indica a che ora è stato prelevato un campione audio: se per valutare in quanto tempo possiamo acquisire abbastanza campioni può essere utile un attributo del genere, porterà sicuramente poche o nessuna informazione per una analisi come quella svolta in questo lavoro, introducendo rumore e contribuendo all'overfitting del modello.

I benefici della feature selection su questo dataset sono quindi chiari, essendo denso di feature. Una selezione delle feature manuale è una delle opzioni migliori se si conosce il dominio applicativo e gli attributi superflui sono noti a priori, ma una selezione solo manuale non è quasi mai possibile, se non in contesti specifici; per questo ci si affida a metodi di feature selection automatica, che possono essere di tipo unsupervised o supervised.

Si parla di unsupervised feature selection se si applica un algoritmo di machine learning che cerca di comprendere autonomamente l'apporto informativo di ogni feature

e scartare quelle poco importanti, mentre si parla di supervised feature selection se tale processo è guidato.

Esistono molti metodi per la feature selection in letteratura ed in generale non vi è uno standard che si può applicare in tutti i casi, ma bisogna valutare caso per caso quale è il metodo più appropriato. Nel paper da cui sono tratti i dati viene utilizzato un algoritmo chiamato mrMr (minimum redundancy Maximum relevance). Questo algoritmo ha guadagnato popolarità negli ultimi anni dopo essere stato utilizzato con successo in uno studio recente [9] e si basa sul selezionare un feature set in cui le feature hanno il maggiore impatto sulla variabile (maximum relevance) target ma sono correlate al minimo tra di loro (minimum redundancy).

In questo lavoro viene utilizzato un algoritmo che segue lo stesso principio, implementato in maniera differente: si effettua un dendrogramma delle feature, che viene tagliato ad una certa altezza per selezionare un subset ottimale delle stesse. In altre parole, il dendrogramma ci permette di effettuare il clustering delle features e di capire quanto esse siano ridondanti sulla base della mutua correlazione.

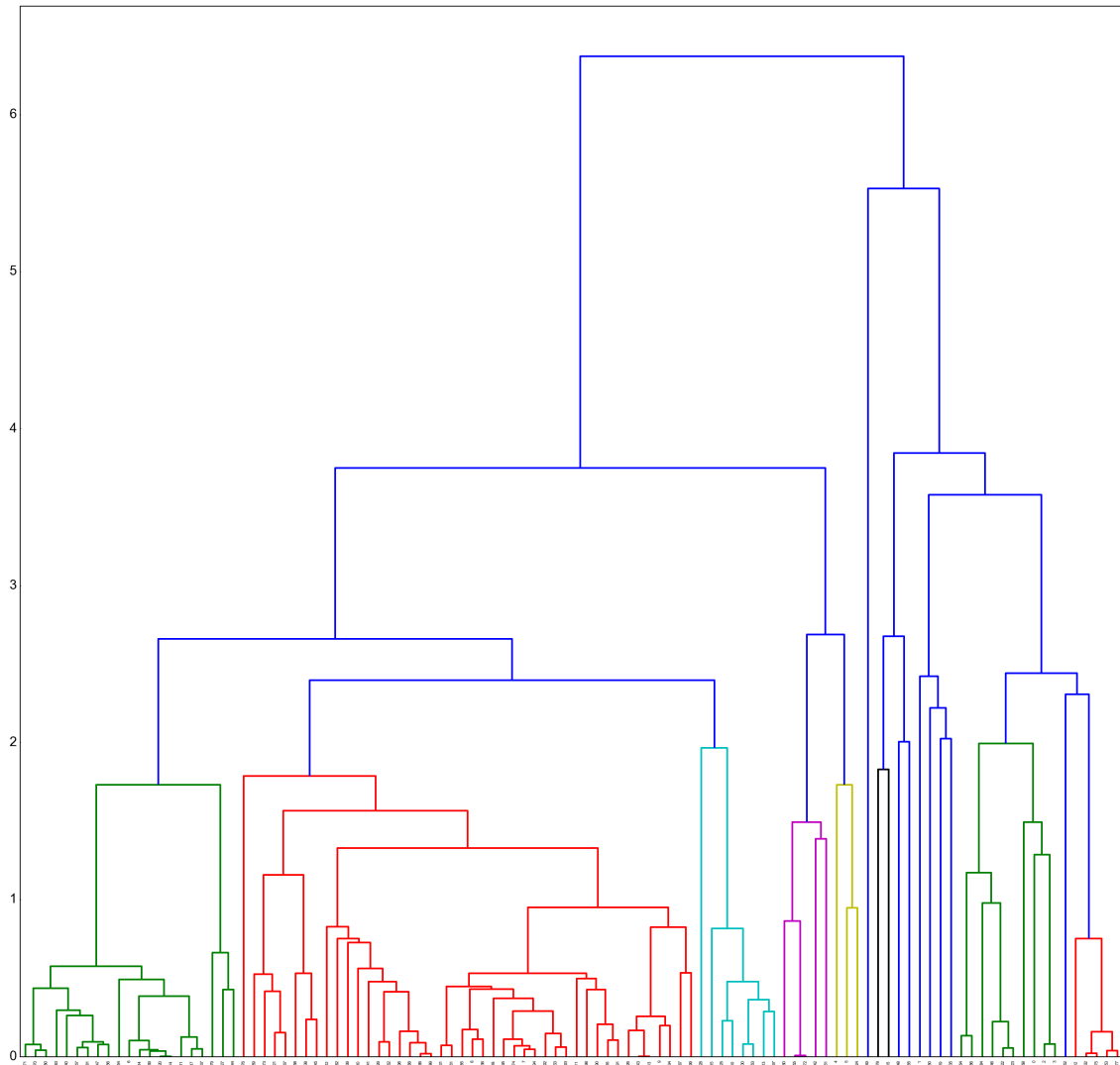
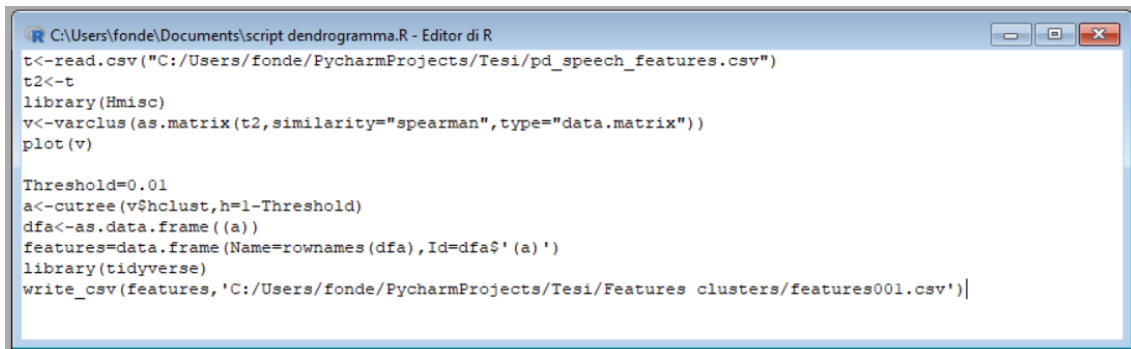


Figura 3.2: Esempio di dendrogramma

Tagliando il dendrogramma viene selezionata casualmente da ogni cluster una feature. Il fatto che il taglio sia casuale non inficia la rilevanza di tale feature poiché il dendrogramma è costruito raggruppando features simili, che hanno lo stesso effetto sulla variabile target.

La creazione ed il taglio del dendrogramma sono stati effettuati con R, per facilitarne il processo vista l'ampia presenza di letteratura e risorse a riguardo. Il taglio è stato effettuato a diverse altezze dell'albero in modo da testare gli effetti di avere un numero maggiore o minore di features con un diverso grado di correlazione. Il taglio è stato effettuato alle soglie di correlazione : 0.9 , 0.7 , 0.5 , 0.3 , 0.2 , 0.1 .



```

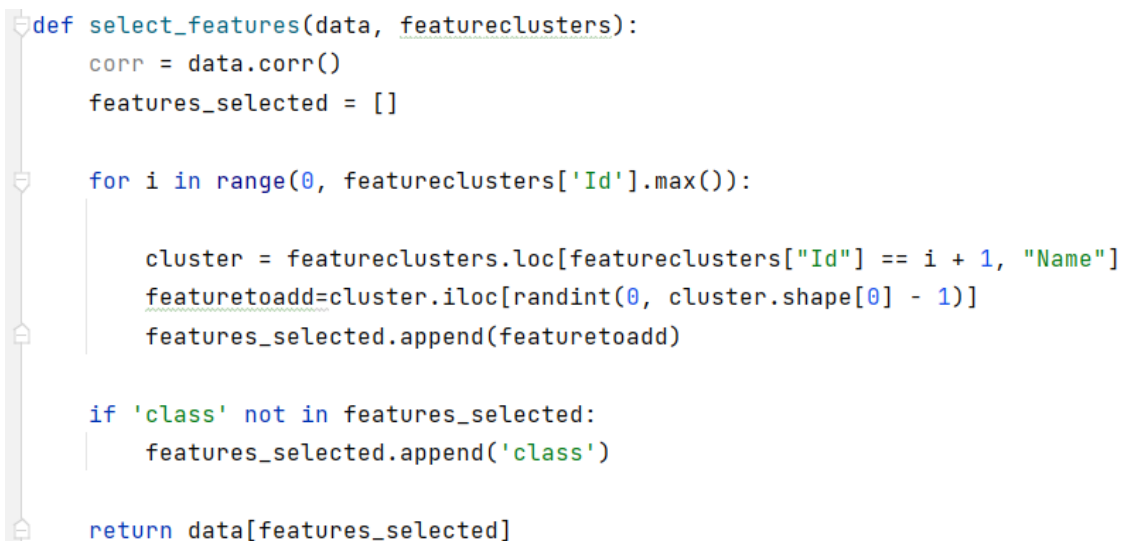
t<-read.csv("C:/Users/fonde/PycharmProjects/Tesi/pd_speech_features.csv")
t2<-t
library(Hmisc)
v<-varclus(as.matrix(t2,similarity="spearman",type="data.matrix"))
plot(v)

Threshold=0.01
a<-cutree(v$hclust,h=1-Threshold)
dfa<-as.data.frame((a))
features=data.frame(Name=rownames(dfa),Id=dfa$(a))
library(tidyverse)
write_csv(features,'C:/Users/fonde/PycharmProjects/Tesi/Features clusters/features001.csv')

```

Figura 3.3: Script per la generazione e il taglio del dendrogramma

Il risultato del taglio del dendrogramma è un file che contiene i cluster di feature, rami del dendrogramma che sono stati collassati in foglia ad una certa soglia di correlazione dallo script di cui sopra. Per estrarre una feature dal cluster utilizziamo Python, in quanto il resto del lavoro sarà svolto in questo ambiente ed è conveniente ai fini operativi poter manipolare questi dati in Python nelle diverse fasi del processo.



```

def select_features(data, featureclusters):
    corr = data.corr()
    features_selected = []

    for i in range(0, featureclusters['Id'].max()):

        cluster = featureclusters.loc[featureclusters["Id"] == i + 1, "Name"]
        featuretoadd=cluster.iloc[randint(0, cluster.shape[0] - 1)]
        features_selected.append(featuretoadd)

    if 'class' not in features_selected:
        features_selected.append('class')

    return data[features_selected]

```

Figura 3.4: Selezione di una feature dai cluster di features

Il risultato del taglio riduce il numero di feature sensibilmente. Di seguito sono mostrate il numero di features per il taglio a diverse soglie di correlazione

Come si può notare, più si abbassa la soglia di correlazione a cui effettuiamo il taglio, meno feature vengono selezionate. Questo accade perché ad una bassa soglia di correlazione sono di più le features che vengono considerate correlate e quindi appartenenti allo stesso cluster, creando quindi meno clusters ma con un numero

Soglia di correlazione	Numero di features
0.9	456
0.7	342
0.5	278
0.3	224
0.2	203
0.1	179

Tabella 3.1: Feature ricavate alle diverse soglie di correlazione.

maggiore di features al loro interno. Estrae una sola feature per cluster questo risulta direttamente nell'avere meno features.

3.5 Feature scaling

Un ulteriore step da effettuare prima della classificazione è la normalizzazione su una scala uniforme dei valori delle features nel dataset, anche detta *feature scaling*.

Questo passaggio è importante poiché l'algoritmo di classificazione potrebbe dare importanza agli attributi con valore più alto, anche se il range massimo in cui essi variano è diverso da attributo ad attributo. Normalizzare ci permette quindi di rimuovere un possibile bias del classificatore e di portare gli attributi sulla stessa scala.

Esistono vari metodi per la normalizzazione dei datasets:

- *Maximum absolute scaling*: porta i valori delle features in una scala compresa tra -1 e 1;
- *Min-max feature scaling*: porta la scala dei valori delle features nel range $[0,1]$;
- *Standardizzazione per z-score*: trasforma i dati in una distribuzione con media pari a 0 e deviazione standard pari a 1. Da ogni valore si ottiene il valore standardizzato sottraendone la media della feature corrispondente e dividendo per la deviazione standard della stessa.

Differisce dagli altri metodi poiché non riscalda i valori in un range preciso.

- *Robust scaling*: I dati vengono scalati feature per feature sottraendo la mediana e dividendo per l'intervallo interquartile (IQR). L'IQR è definito come la differenza tra il primo e terzo quartile, rappresentando effettivamente, in una

distribuzione gaussiana normale, la parte centrale della distribuzione contenente il 50% dei dati.

Questo tipo di normalizzazione è utile in presenza di outliers nei dati.

In questo lavoro viene utilizzata la tecnica *min-max feature scaling*, poiché si ha bisogno di un range di valori fissato e al contempo non si ha una forte presenza di outliers.

La normalizzazione è stata implementata in Python definendo una funzione `normalize(dataframe)`.

```
def normalize(df):  
    result = df.copy()  
    for feature_name in df.columns:  
        max_value = df[feature_name].max()  
        min_value = df[feature_name].min()  
        result[feature_name] = (df[feature_name] - min_value) / (max_value - min_value)  
    return result
```

Figura 3.5: Funzione di normalizzazione dei dati

3.6 Scelta dei classificatori

Una volta completate tutte le operazioni preliminari sui dati, si passa alla creazione di un modello di classificazione per l'esecuzione della task proposta nel problem statement.

Scegliere il modello di classificazione non è una operazione banale, in quanto diversi tipi di classificatori performano in maniera diversa in base alla conformazione dei dati. Proprio per questo motivo, uno degli scopi impliciti dell'analisi in corso è proprio quello di scegliere l'algoritmo di classificazione più adatto ad interpretare questo dataset.

Prima di procedere è opportuno esplorare brevemente come opera un classificatore; concettualmente le feature di una istanza presente nel dataset permettono di individuarla in uno spazio detto spazio delle features di dimensionalità pari alla cardinalità dell'insieme degli attributi. L'obiettivo di un classificatore è, a partire da un insieme di dati già classificati, definire una divisione in settori dello spazio delle features, in cui ogni settore implica che le istanze presenti al suo interno siano appartenenti ad una certa classe.

All'arrivo di una nuova istanza, il classificatore la posiziona nello spazio delle feature ricavato durante il training e valuta la classe in base al settore in cui si trova, operando una classificazione sulla base del modello ricavato in precedenza.

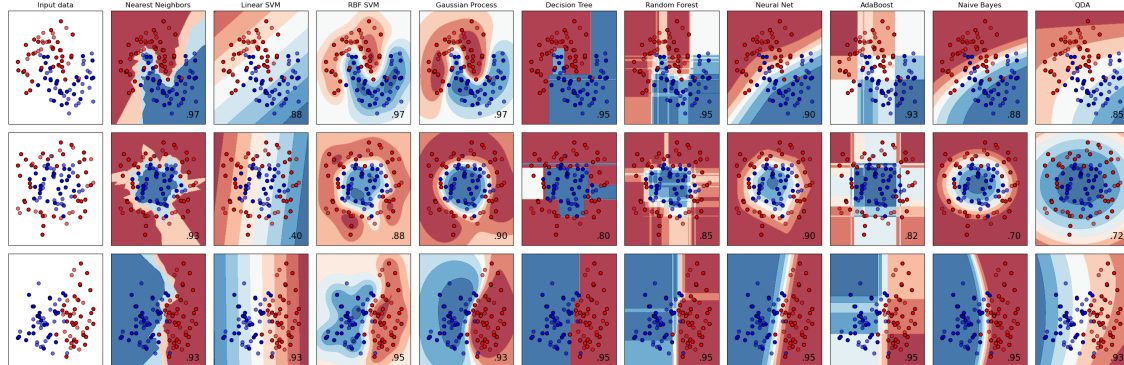


Figura 3.6: Rappresentazione della diversa divisione dello spazio delle feature su un dataset di esempio

Al fine dell'analisi sono stati scelti i classificatori più performanti disponibili tramite la libreria sklearn. Essi sono:

- *LogisticRegression*;
- *KNeighborsClassifier*;
- *Perceptron*;
- *MLPClassifier*;
- *SVC*;
- *GradientBoostingClassifier*;
- *DecisionTreeClassifier*;
- *RandomForestClassifier*;
- *CatBoostClassifier*;
- *GaussianNB*;
- *AdaBoostClassifier*;

- *GaussianProcessClassifier*;

I classificatori sono stati dichiarati nello spazio variabili globale e inseriti all'interno di un dizionario, una struttura dati formata da coppie chiave-valore, in modo da poter effettuare iterazioni sulla struttura dati e non sui singoli classificatori, uniformando il processo di scrittura del codice.

```
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, AdaBoostClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.linear_model import LogisticRegression, Perceptron, LogisticRegressionCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

classifiers = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Nearest Neighbors': KNeighborsClassifier(),
    'Perceptron': Perceptron(),
    'Multi Layer Perceptron': MLPClassifier(),
    'Linear SVM': SVC(),
    'Gradient Boosting Classifier': GradientBoostingClassifier(n_estimators=100),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(n_estimators=1000),
    'Naive Bayes': GaussianNB(),
    'Ada Boost': AdaBoostClassifier(),
    'Gaussian Process': GaussianProcessClassifier(),
}
```

Figura 3.7: Classificatori prima del tuning dei parametri

3.7 Scelta dei parametri e classificazione

Alla scelta dei classificatori si accompagna un'altra importante scelta che condiziona in maniera importante il successo della classificazione, ovvero la scelta dei parametri dei classificatori.

Tutti gli algoritmi utilizzati sono *tuneable* ovvero si può influenzare il processo di apprendimento tramite l'utilizzo di parametri, risultando in un miglioramento (o peggioramento delle performance).

A questo proposito si è operato aggiungendo estimatori all'algoritmo GradientBoostingClassifier, hidden layers al MultiLayerPerceptron e una modifica del numero di vicini nell'algoritmo KNN.

```
classifiers = {  
    'Logistic Regression': LogisticRegression(max_iter=1000),  
    'Nearest Neighbors': KNeighborsClassifier(3),  
    'Perceptron': Perceptron(),  
    'Multi Layer Perceptron': MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="relu", random_state=1),  
    'Linear SVM': SVC(),  
    'Gradient Boosting Classifier': GradientBoostingClassifier(n_estimators=1000),  
    'Decision Tree': DecisionTreeClassifier(),  
    'Random Forest': RandomForestClassifier(n_estimators=500),  
    'CatBoost': CatBoostClassifier(verbose=False),  
    'Naive Bayes': GaussianNB(),  
    'Ada Boost': AdaBoostClassifier(),  
    'Gaussian Process': GaussianProcessClassifier(),  
}
```

Figura 3.8: Classificatori con il tuning dei parametri

Una volta completati tutti i processi preliminari non resta che allenare i classificatori. A questo scopo bisogna creare train set e test set, che serviranno rispettivamente per allenare il classificatore e per valutarne le performance. È importante che i due set di dati non siano in alcun modo sovrapposti, poiché la valutazione delle performance deve avvenire su istanze di dati che il classificatore non conosce, per essere attendibili.

Per generare i set di train e test viene usata la funzione `train_test_split` della libreria `sklearn`, che effettua lo split in base ai parametri che vengono passati. In particolare lo split viene effettuato a 0.2 (quindi l'80% dei dati verrà usato per il training e il rimanente per il testing) e viene indicato il parametro `stratify`, che mantiene la rappresentazione delle classi nel dataset anche all'interno dei test e train sets.

```

def trainclassifiers(data, split, verbose, balance_data=True, normalization=True):

    if(normalization):
        data=normalize(data)
    X = data
    Y = X.pop('class')
    x_train, x_test, y_train, y_test = train_test_split(X, Y, stratify=Y, test_size=split, random_state=0)

    if balance_data:
        smote = SMOTE('not majority', random_state=1)
        x_train, y_train = smote.fit_resample(x_train, y_train)
        x_test, y_test = smote.fit_resample(x_test, y_test)

    models = {}
    for classifier_name, classifier in list(classifiers.items()):
        t_start = time.process_time()
        classifier.fit(x_train, y_train)
        t_end = time.process_time()
        train_time = t_end - t_start

        y_pred = classifier.predict(x_test)
        cmatrix = confusion_matrix(y_test, y_pred)
        dict_metrics = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
        models[classifier_name] = {
            'model': classifier,
            'metrics': dict_metrics,
            'train_time': train_time,
        }
        if verbose:
            print(classifier_name, dict_metrics)
    return models

```

Figura 3.9: Funzione per la creazione dei sets, la classificazione e la creazione di metriche di valutazione

Vengono inoltre salvate le metriche di valutazione della performance dei classificatori, generate tramite la funzione `classification_report` di `sklearn`, che testa il modello su un set di dati di test non presenti nel training.

L'output di questo processo è quindi una struttura dati, `models`, che contiene tutti i modelli allenati e le metriche di performance dei classificatori, sotto forma di dizionario.

3.8 Ensemble di classificatori

Si definisce *ensemble learning* il processo per il quale più modelli, detti esperti, sono combinati per risolvere un particolare problema di machine learning. In questo contesto un ensemble di classificatori è un classificatore che basa il suo output sull'output di più classificatori.

Questo metodo permette di migliorare le performance in un problema di classificazione rispetto all'usare i classificatori singoli e può essere eseguito in molti modi. Tecnicamente i metodi di boosting, utilizzati dai classificatori AdaBoost e Gradient-BoostingClassifier, sono metodi di ensemble, poiché mettono insieme le predizioni di più weak learners.

La libreria sklearn mette a disposizione tutti gli strumenti per creare un ensemble di classificatori, di cui esistono molte varianti basate su diversi algoritmi di coordinazione dei classificatori inglobati nell'ensemble. In questo lavoro verrà utilizzato un ensemble che utilizza una strategia di voto per decidere la classe di una istanza dei dati. Per decidere quali algoritmi utilizzare è stato effettuato un ranking delle performance dei classificatori basato sull'accuratezza e sono stati selezionati i tre migliori.

Classifier name	Accuracy
Nearest Neighbors	0.864706
Gradient Boosting Classifier	0.864706
CatBoost	0.847059
Multi Layer Perceptron	0.829412
Perceptron	0.811765
Ada Boost	0.782353
Random Forest	0.776471
Gaussian Process	0.770588
Logistic Regression	0.758824
Linear SVM	0.758824
Naive Bayes	0.752941
Decision Tree	0.741176

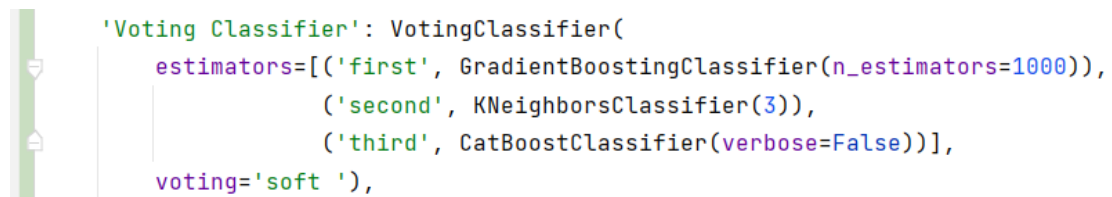
Figura 3.10: Ranking dei classificatori in base all'accuratezza

I tre modelli selezionati sono GradientBoostingClassifier, AdaBoost e KNeighbors. Per creare un ensemble basato sul voto bisogna anche scegliere la modalità di voto:

- *Hard voting*: all'arrivo di una istanza ogni classificatore vota per la classe predetta. L'output dell'ensemble sarà la classe con più voti;

- *Soft voting*: all'arrivo di una istanza ogni classificatore vota per la classe predetta. Ogni voto viene moltiplicato per un coefficiente proporzionale alla confidenza che il classificatore ha in quella predizione prima di essere sommato agli altri voti. La classe con il voto più alto è l'output dell'ensemble.

In questo caso viene scelto il soft voting, poiché i classificatori scelti sono molto diversi tra loro e questo è il metodo di voting più indicato per questa configurazione.

A screenshot of a code editor with a light green sidebar on the left containing a file explorer icon and a search icon. The main area displays Python code for creating a Voting Classifier ensemble. The code is as follows:

```
'Voting Classifier': VotingClassifier(  
    estimators=[('first', GradientBoostingClassifier(n_estimators=1000)),  
                ('second', KNeighborsClassifier(3)),  
                ('third', CatBoostClassifier(verbose=False))],  
    voting='soft '),
```

Figura 3.11: Ensemble di classificatori

Capitolo 4

Analisi dei risultati

In questo capitolo sono mostrati i risultati del lavoro di classificazione svolto. Volendo riassumerne brevemente i passaggi si è partito da un dataset di feature vocali di pazienti parkinsoniani e *healty control* con l'obiettivo di creare un modello di classificazione che riesca a diagnosticare la malattia. Il dataset è stato pulito, normalizzato sulla scala $[0,1]$ e ne sono state selezionate le features tramite metodi di clustering con dendrogramma. Una volta preparati i dati sono stati scelti e ottimizzati dei classificatori per effettuare la classificazione e dei più performanti è stato effettuato un ensemble per migliorare ancora di più l'accuratezza dell'analisi.

Si è scelto inoltre di utilizzare l'accuratezza come metrica di confronto principale tra i classificatori, sulla linea dei lavori che attualmente costituiscono lo stato dell'arte, ma altri parametri come recall, precision e training time non vengono trascurati dall'analisi dei risultati.

4.1 Risultati utilizzando tutte le feature

Allo scopo di creare una baseline di comparazione, lo script di classificazione è stato eseguito sul dataset originale, senza alcun tipo di feature selection, normalizzazione e bilanciamento

Questo è altamente scorretto nella pratica e viene fatto solo a scopo dimostrativo.

Classifier name	Accuracy
Gradient Boosting Classifier	0.894737
Voting Classifier	0.885965
CatBoost	0.877193
Random Forest	0.859649
Ada Boost	0.842105
Logistic Regression	0.763158
Decision Tree	0.763158
Naive Bayes	0.754386
Multi Layer Perceptron	0.745614
Linear SVM	0.745614
Nearest Neighbors	0.692982
Perceptron	0.254386
Gaussian Process	0.254386

Figura 4.1: Ranking dei classificatori in base all'accuratezza sul dataset originale

Guardando soltanto all'accuracy dei primi modelli si potrebbe essere tratti in inganno a pensare che ci siano buoni risultati di classificazione, ma ciò è molto lontano dal vero. Come primo aspetto bisogna considerare che molti dei modelli hanno performance sub-ottimali, con i due modelli peggiori che sono addirittura sotto la soglia della classificazione casuale.

Un secondo aspetto da considerare è che i tempi di training sono elevatissimi, arrivando a 954 secondi per il CatBoost. Questo è indicativo del fatto che i modelli hanno troppe feature da valutare.

```
##### CatBoost #####
Trained in : 954.8125
[[18 11]
 [ 3 82]]
      precision    recall  f1-score   support

     0       0.86      0.62      0.72         29
     1       0.88      0.96      0.92         85

 accuracy              0.88         114
 macro avg       0.87      0.79      0.82         114
weighted avg       0.88      0.88      0.87         114
```

Figura 4.2: Metriche di classificazione per l'algoritmo CatBoost

Prendendo nuovamente CatBoost come esempio, per le sue ottime performan-

ce di accuracy, si può notare che si ha una forte disparità nelle performance della classificazione su istanze di classe 0 e 1. La classificazione sulla classe 0, la meno numerosa, risulta infatti molto meno efficace. Si osservano quindi in questi risultati gli effetti diretti dello sbilanciamento del dataset.

Queste metriche sono un best-case, poiché è stato selezionato uno dei top performer, ma guardando agli altri modelli troviamo persino casi, come il KNearestNeighbours, in cui la differenza di precision e recall nella classe 0 è molto minore del 50% degli stessi valori per la classe 1, rendendo la accuracy media accettabile, ma il modello inutilizzabile in un contesto reale.

4.2 Risultati applicando feature selection a varie threshold

Il primo step che è stato effettuato è stata la feature selection. È stato utilizzato un dendrogramma basato sulla correlazione mutua delle features, tagliato a varie soglie, che variano da alta correlazione (0.9) a bassissima correlazione (0.1) .

Analizzando i dati di accuracy dei top performer è facile notare che non sembra

Soglia di correlazione	Numero di features	Accuracy del top performer	Top performer
nessuna soglia	755	0.894737	Gradient Boosting Classifier
0.9	456	0.894737	Gradient Boosting Classifier
0.7	342	0.885965	CatBoost
0.5	278	0.868421	CatBoost
0.3	224	0.877193	Gradient Boosting Classifier
0.2	203	0.842105	RandomForest
0.1	179	0.859649	RandomForest

Tabella 4.1: Feature ricavate alle diverse soglie di correlazione.

esserci una grande perdita di contenuto informativo dalle feature tagliate, considerando che le performance dei classificatori migliori si avvicinano molto alla baseline creata con l'utilizzo di tutte le feature.

Tagliando le feature il modello beneficia di una riduzione della complessità ed un aumento nella generalità. Inoltre, in un contesto operativo, avere un modello che necessita di meno features significa avere meno dati da raccogliere, velocizzando anche i processi precedenti alla classificazione.

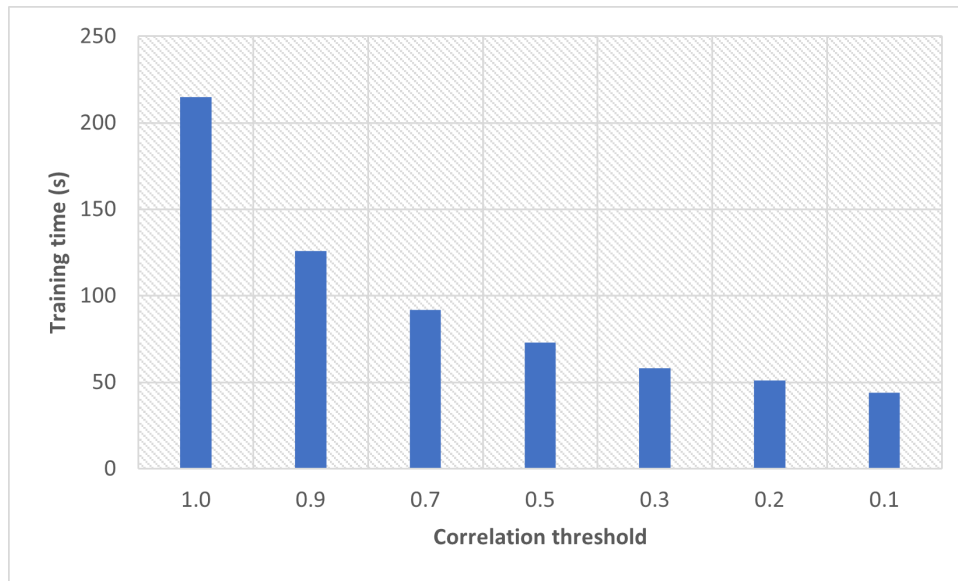


Figura 4.3: Tempo necessario per l'allenamento di GradientBoostingClassifier alle varie soglie di taglio delle features

Appare evidente, dai dati ricavati nel testing, che la diminuzione di complessità portata dalla feature selection ha portato ad una forte riduzione del training time. Questo è vero per tutti i modelli, ma è particolarmente vero per i modelli che fanno utilizzo di boosting, i più performanti su questo dataset, portando un ulteriore incentivo ad effettuare la feature selection.

In ultima analisi, c'è un tradeoff tra la performance del modello e l'aggressività del processo di feature selection, che porta inevitabilmente a dover effettuare una scelta. Per questa analisi viene presa in considerazione la soglia di correlazione 0.9 per la feature selection, la meno aggressiva, poiché porta la minore perdita di contenuto informativo nonostante la grande riduzione di complessità del modello e di training time.

Questo non esclude che in un contesto dove si vogliano utilizzare meglio le risorse non si possa considerare una soglia più aggressiva, a patto di poter sostenere la degradazione di performance.

4.3 Risultati applicando tecniche di bilanciamento e normalizzazione

I risultati ottenuti applicando feature selection ed ottimizzazione dei parametri sono promettenti, ma emergono comunque alcune problematiche importanti.

```
##### Multi Layer Perceptron #####
Trained in : 2.640625

          precision    recall  f1-score

0         0.00         0.00         0.00
1         0.75         1.00         0.85

accuracy          0.75
macro avg         0.37         0.50         0.43
weighted avg      0.56         0.75         0.64
```

Figura 4.4: Performance di Multi Layer Perceptron su un dataset non normalizzato e non bilanciato

La prima di queste problematiche è la performance dei classificatori classe per classe. Il fatto che il dataset non sia bilanciato porta ad avere una predizione che favorisce una classe rispetto all'altra, con carenza di performance nel classificare la classe meno rappresentata nel training set.

Oltre a questo problema, il fatto che il dataset non sia normalizzato sbilancia ulteriormente la previsione verso la classe che ha attributi maggiori. Questo comportamento è poco evidente nei top performer, ma molto evidente nel resto dei classificatori, che come nel caso della figura 4.4 possono finire a dare previsioni che hanno sempre lo stesso output.

```
##### Multi Layer Perceptron #####
Trained in : 2.49375

              precision    recall  f1-score

0.0           0.86       0.73       0.79
1.0           0.77       0.88       0.82

accuracy              0.81
macro avg           0.81       0.81       0.80
weighted avg        0.81       0.81       0.80
```

Figura 4.5: Performance di Multi Layer Perceptron sul dataset normalizzato e bilanciato

Nella figura 4.5 si può vedere immediatamente l'effetto che ha avuto, sullo stesso classificatore, la normalizzazione e bilanciamento dei dati usati nel training. Si è infatti passati da una classificazione inesistente ad un output piuttosto consistente sopra l'80% di accuratezza.

```
Classifier name  Accuracy
Gradient Boosting Classifier  0.900000
Ada Boost       0.870588
Nearest Neighbors  0.864706
CatBoost        0.858824
Multi Layer Perceptron  0.805882
Random Forest   0.782353
Decision Tree    0.770588
Logistic Regression  0.764706
Gaussian Process  0.758824
Naive Bayes      0.741176
Linear SVM        0.735294
Perceptron       0.676471
```

Figura 4.6: Accuratezza dei classificatori sul dataset normalizzato e bilanciato

Dall'analisi dell'accuratezza dei classificatori risulta evidente che la normalizzazione e il bilanciamento hanno reso migliore la performance di tutti i modelli. Inoltre, analizzando modello per modello, i valori di precision e recall sono diventati più consistenti tra le due classi.

Risulta quindi, a seguito delle operazioni sopracitate, possibile costruire un modello di classificazione che operi con efficacia.

4.4 Ensemble di classificatori e risultati finali

Grazie a feature selection, normalizzazione, bilanciamento e tuning dei classificatori si è arrivati ad un risultato di classificazione preciso e che mantiene consistenza con le varie classi. È stato che creato, in precedenza, un classificatore di ensemble basato sul soft voting, di cui ora si analizzeranno i risultati di classificazione.

```
##### Voting Classifier #####
Trained in : 75.640625
[[125 16]
 [ 6 135]]
```

	precision	recall	f1-score	support
0.0	0.95	0.89	0.92	141
1.0	0.89	0.96	0.92	141
accuracy			0.92	282
macro avg	0.92	0.92	0.92	282
weighted avg	0.92	0.92	0.92	282

Figura 4.7: Performance complete dell'ensemble di classificatori

Come è possibile osservare, un metodo di ensemble utilizzato su un dataset processato correttamente può restituire risultati nettamente migliori dei singoli classificatori. In particolare questo tipo di ensemble riesce ad avere un valore di recall del 96% per la classe 1, il più alto registrato tra i classificatori, e una accuratezza del 92%, più alta del 2% rispetto al best performer non-ensemble.

Classifier	Accuracy	Precision	Recall	F1-score
Voting Classifier	0.92	0.92	0.92	0.92
Gradient Boosting Classifier	0.90	0.90	0.90	0.90
Ada Boost	0.87	0.87	0.87	0.87
K-Nearest Neighbours	0.86	0.89	0.86	0.86
CatBoost	0.85	0.86	0.86	0.86
Multilayer Perceptron Classifier	0.80	0.81	0.81	0.80
Random Forest	0.78	0.80	0.78	0.78
Decision Tree	0.77	0.78	0.77	0.77
Logistic Regression	0.76	0.77	0.76	0.76
Gaussian Process	0.75	0.76	0.76	0.76
Naive Bayes	0.74	0.75	0.74	0.74
Linear SVM	0.73	0.74	0.74	0.73
Perceptron	0.67	0.78	0.68	0.64

Tabella 4.2: Performance di tutti i classificatori dopo feature selection, normalizzazione, bilanciamento e tuning dei parametri

4.5 Analisi conclusiva e minacce alla validità del modello

Il problem statement da cui è originato questo lavoro mostrava l'esigenza di un modello di classificazione sulla base di features vocali per la diagnosi automatica del Parkinson. Al termine del lavoro si può dire che, seguendo una pipeline per il corretto trattamento dei dati, tale modello può essere creato per essere efficiente e funzionale, tramite varie tecniche sulla base di esigenze di scalabilità o di accuratezza.

Si è visto come il miglior metodo per la costruzione del modello è un classificatore di tipo *ensemble* basato su soft voting, che può raggiungere performance molto alte, comparabili con altre tecniche di classificazione per diagnosi automatica.

Nonostante i risultati promettenti, è bene guardare a ciò che può minacciare la validità dei modelli creati, in modo da comprendere come si può iterare, volendo, per migliorarne la solidità

Capitolo 5

Conclusione e sviluppi futuri

A conclusione di questo lavoro, ne si può fare breve analisi considerando il problem statement da cui è originato, che mostrava l'esigenza di un modello di classificazione sulla base di features vocali per la diagnosi automatica del Parkinson. Al termine del lavoro si può dire che, seguendo una pipeline per il corretto trattamento dei dati, tale modello può essere creato per essere efficiente e funzionale, tramite varie tecniche in base alle esigenze di scalabilità e di accuratezza.

Si è visto come il miglior metodo per la costruzione del modello è un classificatore di tipo *ensemble* basato su soft voting, che può raggiungere performance molto alte, accuratezza pari a 0.92, comparabili o superiori ad altre tecniche di classificazione per diagnosi automatica.

È bene anche valutare i limiti di questo lavoro: se bene i risultati ottenuti siano promettenti, il pool di dati e di partecipanti utilizzati è molto ridotto, presentando problemi di rilevanza statistica dei dati ottenuti, ma anche aprendo a nuove possibilità nel futuro in cui si raccolgano altri dati del genere.

Non è difficile immaginare infatti che con una mole più grande di dati si aprano le porte a tecniche di machine learning non supervisionato e diversi tipi di classificatori ensemble, o che possano cambiare le performance di alcuni classificatori, scalando con i dati.

Un ulteriore limite del modello è dato dalla mancanza di dati di training per effettuare diagnosi differenziale, ovvero per capire se il paziente ha determinate features vocali per via del Parkinson o per via di altre malattie.

In conclusione, la diagnosi automatica del Parkinson tramite features vocali risulta possibile ed efficace, ma altri dati potrebbero aiutare a costruire un quadro più

completo per la risoluzione ottimale di questo problema.

Ringraziamenti

Bibliografia

- [1] University of San Diego. What is medical informatics? [Online]. Available: <https://onlinedegrees.sandiego.edu/what-is-medical-informatics/>
- [2] David Reinsel, John Gantz, John Rydning, “The Digitization of the World From Edge to Core.” *IDC White Paper*.
- [3] J. P. Micheline Kamber, Jiawei Han, *Data Mining: Concepts and Techniques*, ser. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2012.
- [4] Hector A. Gonzalez-Usigli. Malattia di parkinson (parkinson disease, pd). [Online]. Available: <https://www.msmanuals.com/it-it/casa/disturbi-di-cervello,-midollo-spinale-e-nervi/disturbi-del-movimento/malattia-di-parkinson-parkinson-disease,-pd>
- [5] Fernando L. Pagan, “Improving outcomes through early diagnosis of Parkinson’s disease.” *Pubmed*.
- [6] Jie Mei, Christian Desrosiers, Johannes Frasnelli, “Machine Learning for the Diagnosis of Parkinson’s Disease: A Review of Literature.” *Frontiers*, 2021.
- [7] Liqiong Yang , Yijia He , Xianwei Zou , Jianping Yu , Man Luo , Xiaolei Liu , Greg Mirt , Lei Sun, Fan Xu, “Vocal changed in Parkinson’s disease patients,” *OAT*, 2019.
- [8] C. Okan Sakar, Gorkem Serbes, Aysegul Gunduz, Hunkar C. Tunc, Hatice Nizam, Betul Erdogan Sakar, Melih Tutuncu, Tarkan Aydin, M. Erdem Isenkul, Hulya Apaydin, “A comparative analysis of speech signal processing algorithms for Parkinson’s disease classification and the use of the tunable Q-factor wavelet transform.” *Applied Soft Computing Journal*, 2018.

-
- [9] A. Tsanas, M. Little, P.E. McSharry, L.O. Ramig, “New nonlinear markers and insights into speech signal degradation for effective tracking of Parkinson’s disease symptom severity,” *International Symposium on Nonlinear Theory and its Applications (NOLTA)*, 2010.
- [10] Zhenyu Zhao, Radhika Anand, Mallory Wang. Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. [Online]. Available: <https://eng.uber.com/research/maximum-relevance-and-minimum-redundancy-feature-selection-methods-for-a-marketing-machine-learning-platform/>
-