Politecnico di Torino

Laurea Magistrale in Ingegneria Informatica

Orthopedic patients analysis

Docente:

Prof. Francesco Vaccarino

Studente:

Rocco Italiano

Anno Accademico 2018/2019

Indice

1. Introduzione e contenuto del dataset

L'obiettivo di questa tesina è di applicare le nozioni apprese durante il corso di Data Spaces a un set di dati utilizzando degli algoritmi di Machine Learning, per produrre un risultato utile in termini scientifici.

L'oggetto di studio della tesina è il dataset "Biomechanical features of orthopedic patients" donato nel 2011 dal Dott.re Henrique da Mota e costruito durante un periodo di ricerca presso un gruppo di ricerca applicata in Ortopedia. Il set di dati è consultabile e scaricabile gratuitamente da UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/vertebral+column).

Il dataset raccoglie i dati di 310 pazienti e li classifica in tre categorie: Normal (NO), Disk Hernia (DH) e Spondylolisthesis (SL).

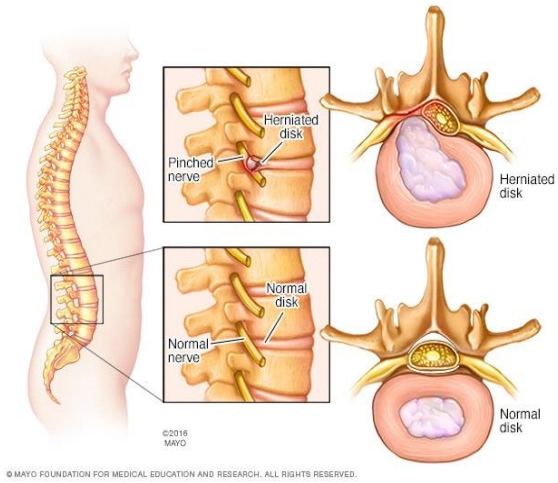


Figura 1: Differenza tra disco con l'ernia e disco normale

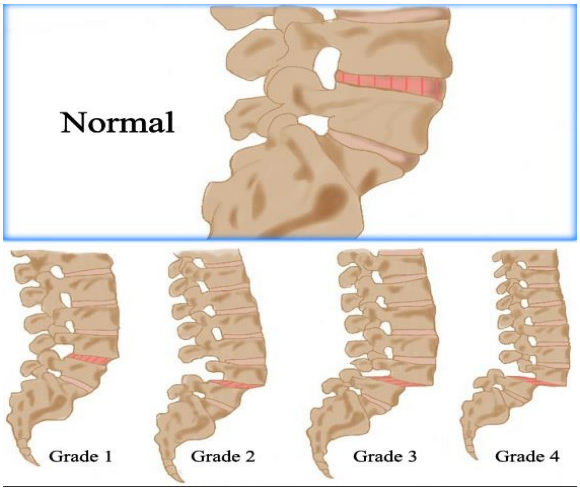


Figura 2: Raffigurazione della spondilolisi che consiste nell'interruzione di continuità dell'istmo vertebrale

Ogni paziente è rappresentato da sei attributi di natura biomeccanica, derivanti dalla forma e orientamento della pelvi e della colonna lombare, nel seguente ordine:

- **pelvic incidence**
- **pelvic tilt**
- **lumbar lordosis angle**
- **sacral slope**
- **pelvic radius**
- **grade of spondylolisthesis**

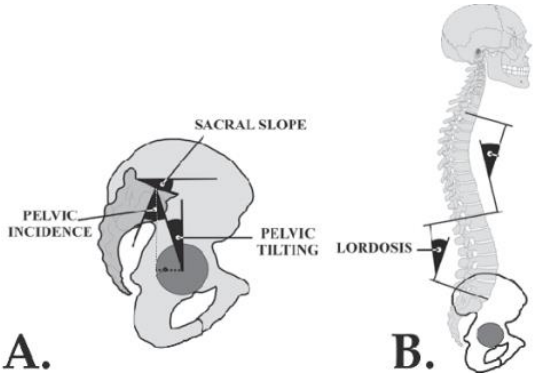


Figura 3: Raffigurazione degli attributi del dataset

In totale abbiamo sei variabili numeriche che descrivono le caratteristiche di ogni paziente. Il mio obiettivo è cercare di trovare il modello che classifichi con la maggior precisione possibile i pazienti suddividendoli per classe, utilizzando gli attributi che descrivono le caratteristiche di ogni paziente.

Quindi procederò separando i predittori che sono gli attributi numerici, dal target che è la classe alla quale appartiene il paziente.

2. Tools utilizzati

Ho scelto di usare come linguaggio di programmazione Python che è uno dei linguaggi più utilizzati nel campo del Machine Learning, che fornisce una documentazione completa online e una serie di librerie molto sviluppate per il Machine Learning. In particolare ho implementato il codice Python utilizzando il Notebook Jupyter che è un'applicazione web che permette di creare documenti che contengano testo formattato e codice eseguibile all'interno del documento, con visualizzazione di output e grafici.

Ho sperimentato un po' con il linguaggio di programmazione R, che è anch'esso un ottimo linguaggio per l'analisi dei dati, ma ho preferito Python perché l'ho trovato più intuitivo.

Le principali librerie e moduli che ho usato per produrre questo lavoro sono i seguenti:

```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import matplotlib.pyplot as plt
%matplotlib inline
```

Figura 4: Screenshot dei comandi utilizzati per importare le librerie

- **pandas** che è una libreria per la manipolazione di dati in formato sequenziale o tabellare ,in particolare mi è stato utile per il caricamento di formati standard per dati tabellari, per la semplicità di aggregazione di dati ed esecuzione e visualizzazione di operazioni numeriche e statistiche.
- **matplotlib** che è un modulo per la generazione di grafici 2D , in particolare ho usato il modulo pyplot che è un'interfaccia procedurale che mi ha permesso di emulare i grafici che vedremo in seguito nella tesina.
- **numpy** che è un'estensione di supporto per la manipolazione di array e matrici multidimensionali che mi ha fornito funzioni matematiche di alto livello con cui operare.
- **seaborn** che è una libreria basata su matplotlib che mi ha permesso di creare interfacce grafiche di alto livello per la rappresentazione grafici statistici informativi.
- **scikit-learn** (sklearn) che è una libreria che fornisce versioni efficienti di un gran numero dei più comuni algoritmi per il Machine Learning sia supervisonato (Decision-Tree, Naive Bayes, KNN, Logistic Regression, SVM) che non supervisionato (K-means, DB-scan).
- **%matplotlib inline** che è un comando grazie al quale è possibile visualizzare grafici che sono l'output di celle di codice in esecuzione, mi è stato molto utile visto che ho utilizzato un ambiente di sviluppo interattivo come Jupyter.

3. Esplorazione dei dati

Come primo step ho importato il set di dati ed ho analizzato i dati, per capire la loro natura e avere una visione generale del dataset.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
pelvic_incidence              310 non-null float64
pelvic_tilt                   310 non-null float64
lumbar_lordosis_angle         310 non-null float64
sacral_slope                  310 non-null float64
pelvic_radius                 310 non-null float64
degree_spondylolisthesis      310 non-null float64
class                         310 non-null object
dtypes: float64(6), object(1)
memory usage: 17.0+ KB
```

Figura 5: L'immagine raffigura le informazioni principali del dataset

Inizialmente è possibile osservare che il dataset è composto da 310 entries e per ogni entry abbiamo 7 colonne, di cui 6 numeriche che rappresentano le caratteristiche biomeccaniche del paziente e una categorica che è la classe con cui il paziente è stato etichettato. E' molto interessante notare come non ci siano dati mancanti per tutti i pazienti.

Nella Figura 6 è possibile osservare lo spaccato iniziale delle prime 5 righe del dataset.

```
df=pd.read_csv('column_3C_weka.csv')
df.head()
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027818	22.552586	39.609117	40.475232	98.672917	-0.254400	Hernia
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	Hernia
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	Hernia
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	Hernia
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	Hernia

Figura 6: Spaccato iniziale del dataset

Ma adesso addentriamoci in qualche statistica un po' più interessante.

```
df.describe()
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
count	310.000000	310.000000	310.000000	310.000000	310.000000	310.000000
mean	60.496653	17.542822	51.930930	42.953831	117.920655	26.296694
std	17.236520	10.008330	18.554064	13.423102	13.317377	37.559027
min	26.147921	-6.554948	14.000000	13.366931	70.082575	-11.058179
25%	46.430294	10.667069	37.000000	33.347122	110.709196	1.603727
50%	58.691038	16.357689	49.562398	42.404912	118.268178	11.767934
75%	72.877696	22.120395	63.000000	52.695888	125.467674	41.287352
max	129.834041	49.431864	125.742385	121.429566	163.071041	418.543082

Figura 7: Screenshot raffigurante alcune statistiche degli attributi numerici del dataset

La tabella in alto (Figura 7) è un riepilogo di alcune misure statistiche per ogni predittore numerico del dataset:

- **count** indica il numero di record presenti nel dataset per ogni attributo biomeccanico.
- **mean** indica il valore medio intorno a cui si attesta ogni gruppo di attributi.
- **std** indica la deviazione standard di ogni gruppo di attributi, dalla quale si può intuire il grado di dispersione di dati intorno alla media.
- **25%**, **50%** e **75%** sono i percentili o quartili che indicano il valore sotto il quale si verifica una certa percentuale di osservazioni. In particolare il 50° percentile, coincide con la mediana e a differenza della media fornisce informazioni sulla distorsione della distribuzione; è anche un'altra definizione di media che è più robusta in presenza di outliers.
- **max** e **min** indicano l'attributo che assume il valore maggiore e minore per ogni gruppo di attributi.

Vediamo adesso questi risultati numerici rappresentandoli graficamente attraverso dei box-plot.

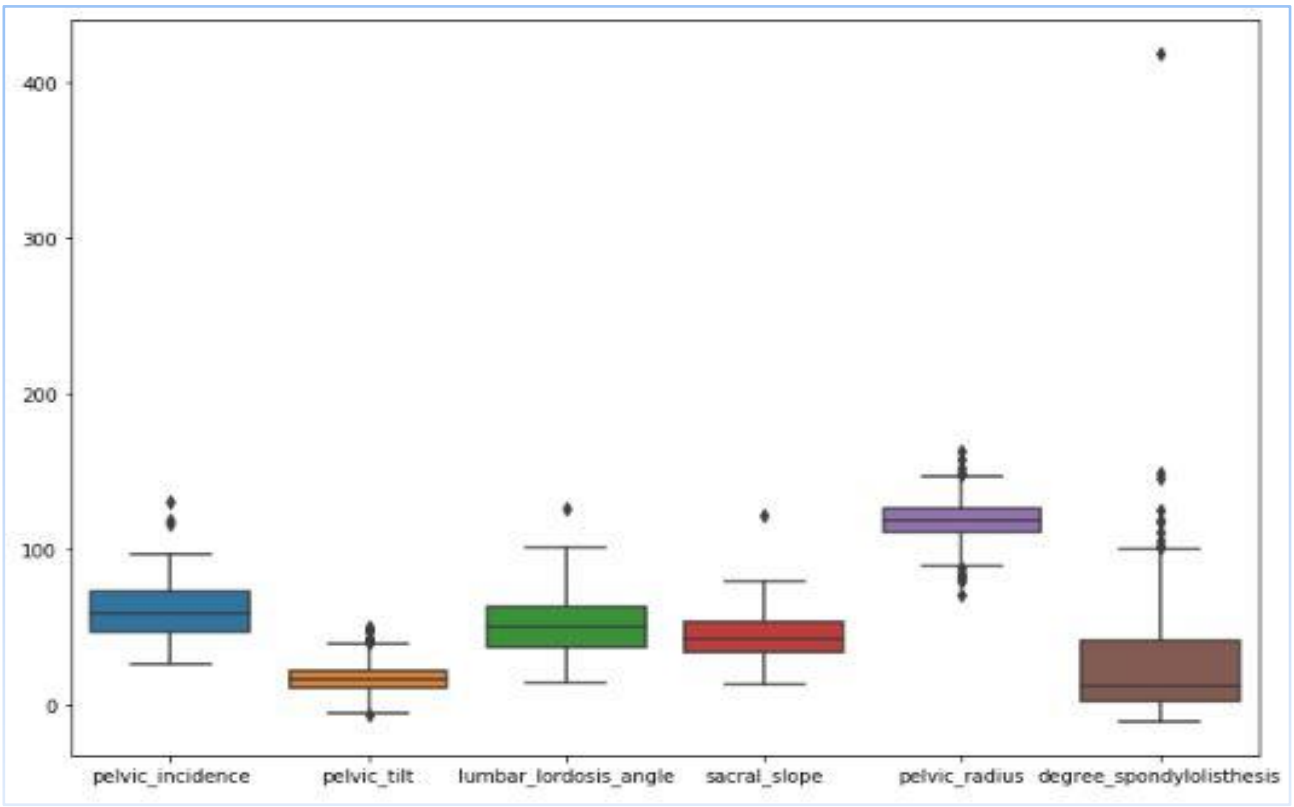


Figura 8: Rappresentazione grafica di ogni gruppo di attributi utilizzando dei box-plot

In tutte le distribuzioni sono presenti degli outliers, in particolare salta all'occhio l'outlier massimo dell'attributo "degree_spondylolisthesis" che è molto lontano dagli altri attributi. Analizzando meglio i valori relativi di quel paziente ho notato dei valori abbastanza strani rispetto agli altri pazienti della stessa categoria, quindi pensando fosse un errore di trascrizione in fase di raccolta del dataset ho deciso di eliminarlo.

Eliminando il valore anomalo si ha una visione migliore della rappresentazione grafica, come mostrato in Figura 9:

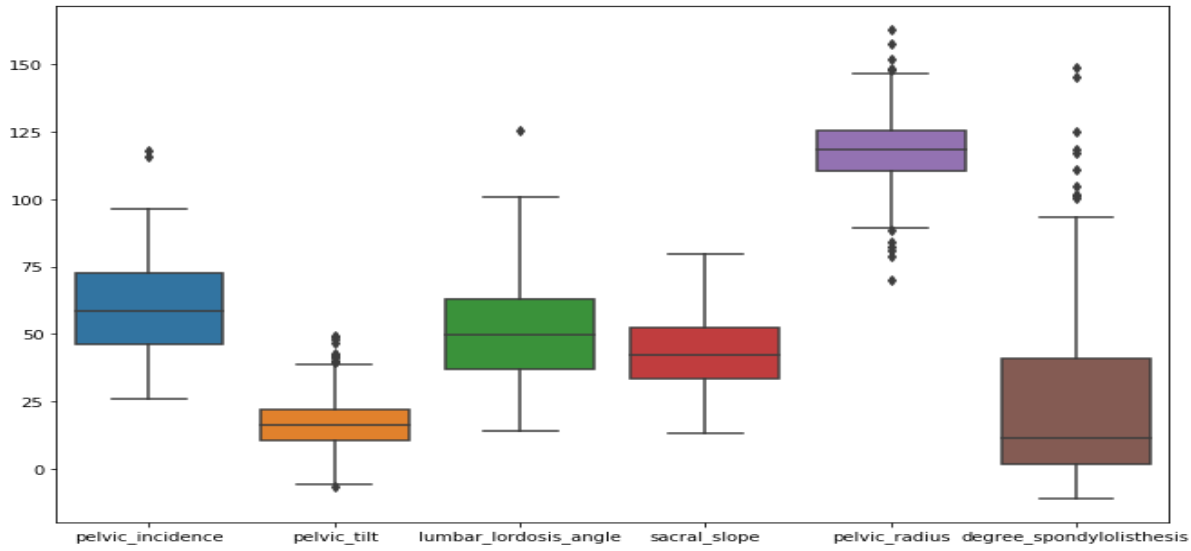


Figura 9: Rappresentazione grafica dei gruppi di attributi, dopo aver ripulito il dataset dal valore anomalo

Dalla distanza tra ciascun quartile e la mediana è possibile notare come l'attributo 'degree_spondylolisthesis' presenta una netta distribuzione asimmetrica positiva (perché la mediana è più piccola della media) rispetto agli altri gruppi attributi, inoltre è possibile osservare come la deviazione standard di questo attributo sia molto elevata e questo indica che i dati si accumulano in zone ben lontane dal valore atteso.

Dal grafico seguente (Figura 10) si può inoltre osservare come variano i valori delle variabili biomeccaniche per ogni classe. In particolare si nota che la classe "Spondylolidthesis" ha valori maggiori rispetto alle altre classi per ogni variabile. Inoltre è possibile osservare come la classe "Spondylolisthesis" abbia un intervallo di rappresentazione maggiore rispetto alle altre classi, questo è sicuramente dovuto alla maggiore dispersione dei dati per quanto riguarda questa classe, ma potrebbe essere anche dovuto a un maggior numero di pazienti che appartengono alla suddetta classe rispetto alle altre due.

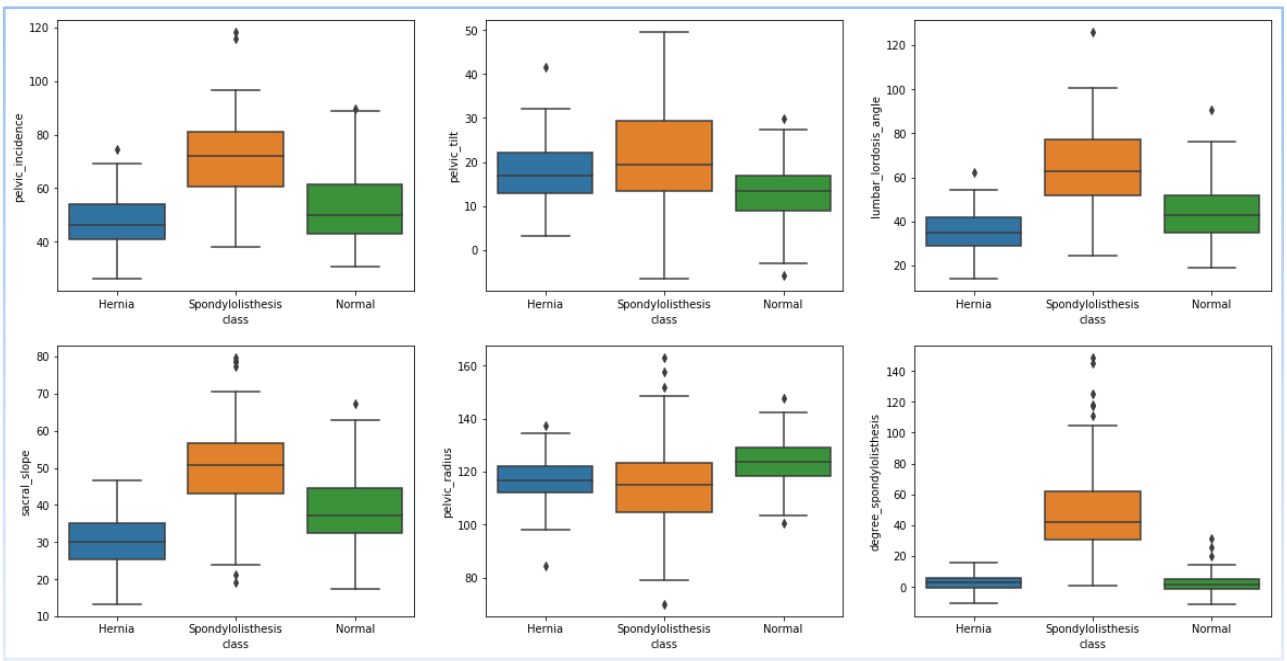


Figura 10: Rappresentazione grafica, attraverso dei box-plot, delle variabili biomeccaniche per ogni classe

La Figura 11, ci toglie ogni dubbio sul bilanciamento delle classi:

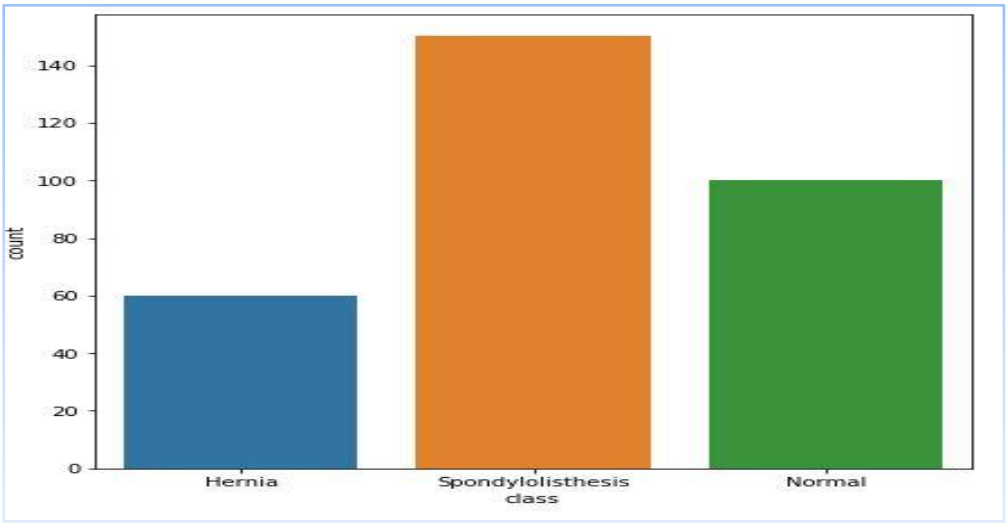


Figura 11: Rappresentazione grafica, tramite istogramma, del numero di pazienti suddivisi per classe

La distribuzione dei pazienti per classe ci dice che abbiamo 60 (19,4%) pazienti a cui è stata diagnosticata l'ernia, 149 (48,2%) pazienti con spondilolisi e 100 (32,3%) pazienti con una colonna vertebrale normale; quindi il dataset contiene un numero di pazienti per classi non bilanciato.

Mettendo da parte le classi proviamo a vedere adesso la correlazione che c'è tra le diverse variabili, per avere un'idea anche sul legame tra una variabile e l'altra.

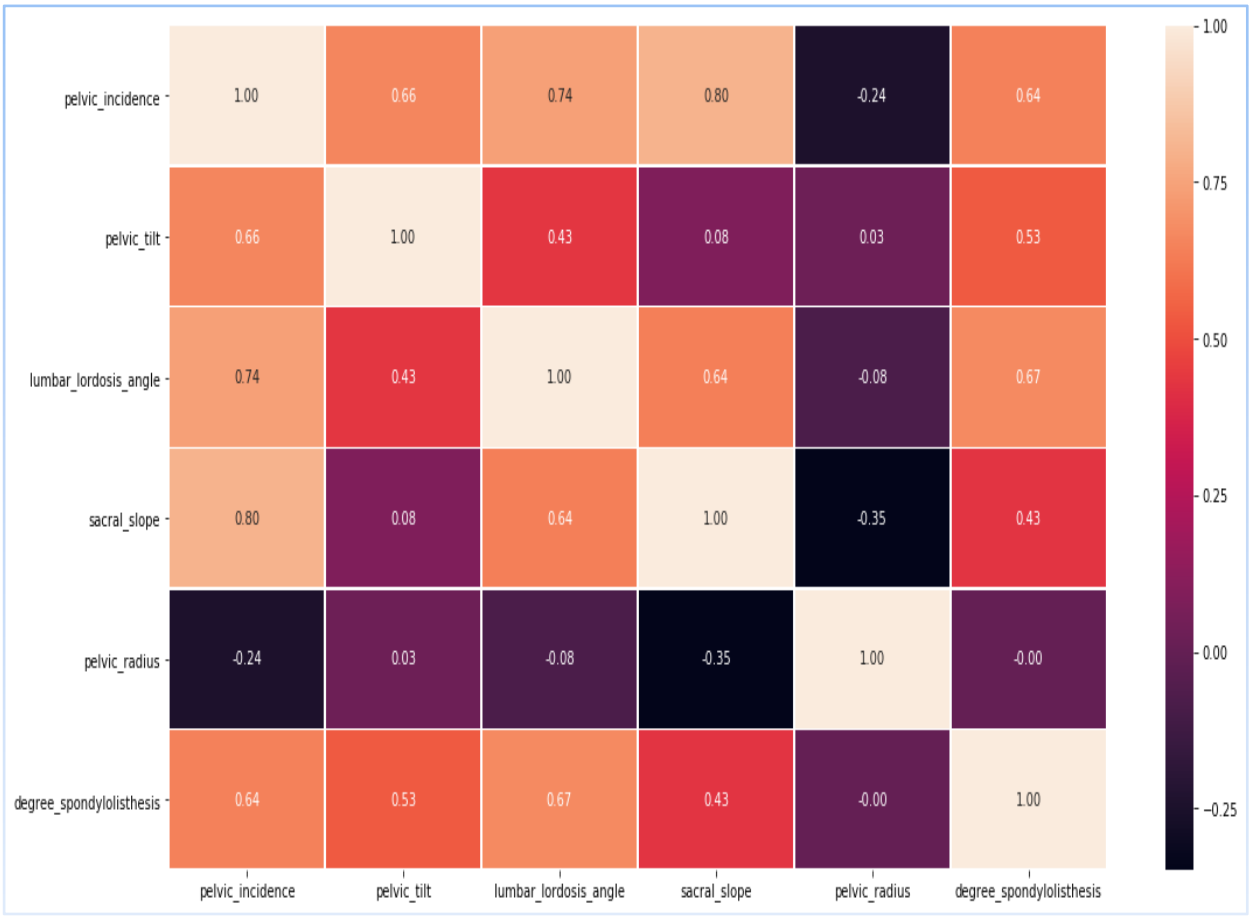


Figura 12: Rappresentazione della matrice di correlazione delle variabili biomeccaniche, attraverso heatmap

La Figura 12 rappresenta la matrice di correlazione e in base all'indice di correlazione di Pearson è possibile capire il grado di correlazione che c'è tra due variabili. Come si può notare è una matrice simmetrica; un alto indice di correlazione ci indica che le due variabili in questione sono positivamente correlate, un basso indice di correlazione indica che le variabili sono negativamente correlate e un indice prossimo allo zero ci indica che le variabili possono essere considerate non correlate.

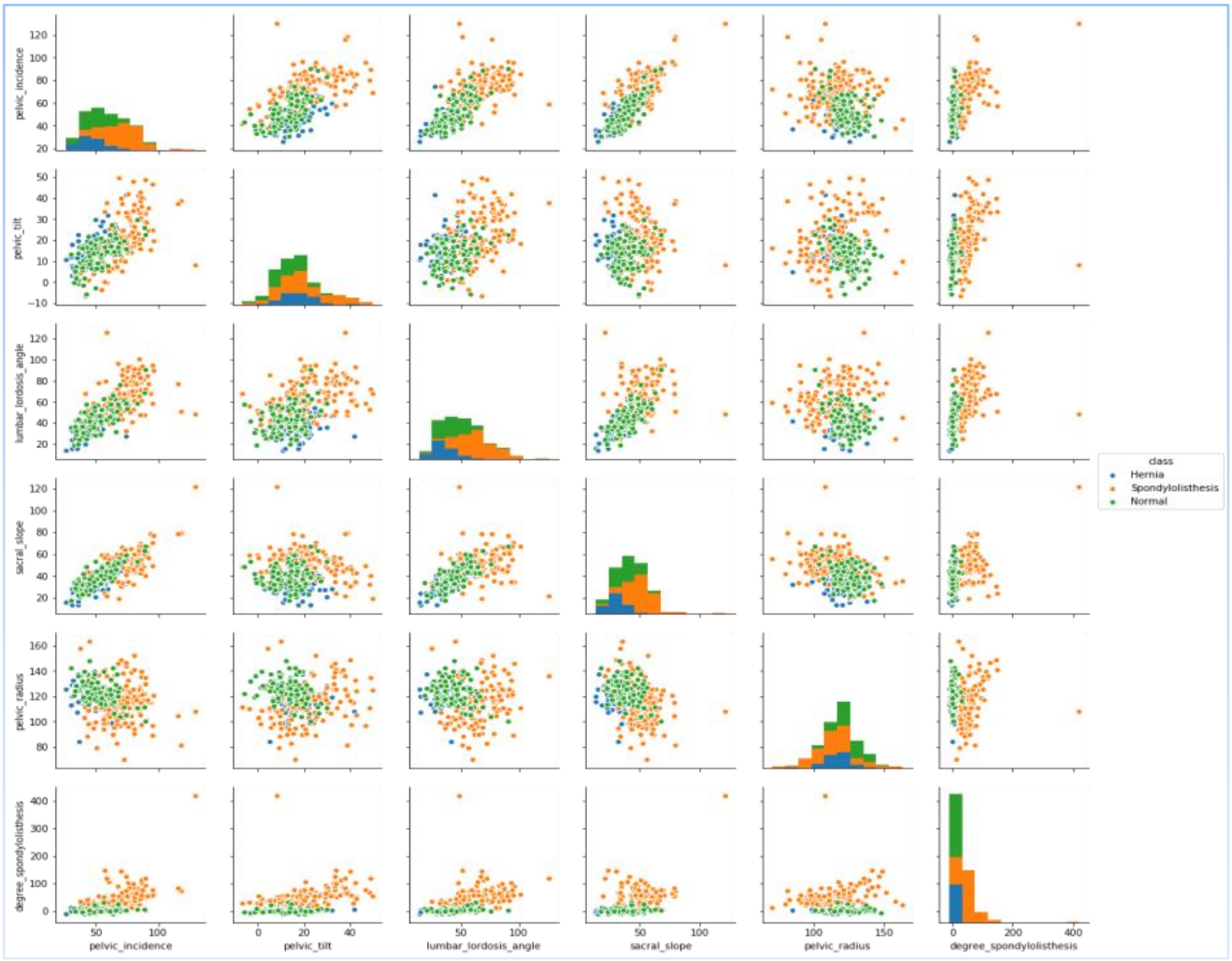


Figura 13: Rappresentazione grafica della matrice di correlazione, attraverso pairplot

La Figura 13 ci illustra graficamente il modello matematico che lega le variabili biomeccaniche. In particolare abbiamo coppie di variabili positivamente correlate con correlazione forte (sacral_slope-pelvic_incidence (0.80), lumbar_lordosis_angle-pelvic_incidence (0.74)) ed è possibile visualizzare la crescita pressocchè lineare. Si hanno anche delle coppie di variabili non correlate (pelvic_radius-pelvic_tilt(0.03), sacral_slope-pelvic_tilt(0.08)) che è difficile rappresentare con un modello matematico. Ci sono inoltre delle coppie inversamente correlate (pelvic_radius-pelvic_incidence(-0.24),pelvic_radius-sacral_slope(-0.35)). Sulla diagonale principale sono rappresentate, per ogni classe, le distribuzioni di probabilità dei dati lungo un certo attributo, calcolate sull'intero dataset.

4. Feature Extraction

Per cercare di capire se con un minor numero di variabili si possa ottenere un risultato che sia approssimabile a quello che ho usando tutte le variabili, applicherò delle tecniche di feature extraction con l'obiettivo di migliorare l'efficienza computazionale dell'algoritmo di apprendimento e di migliorare la performance in fase di predizione alleviando i problemi legati alla *curse of dimensionality*. In particolare le tecniche che userò per fare ciò saranno la Principal Component Analysis (PCA) e la Linear Discriminant Analysis (LDA). Sia PCA che LDA sono tecniche di trasformazione lineare, applicando queste tecniche trasformerò il dataset originale in un sottospazio di features di dimensionalità ridotta con l'obiettivo di mantenere le informazioni più rilevanti del dataset.

4.1. Principal Component Analysis

La Principal Component Analysis è un algoritmo di apprendimento non supervisionato utilizzato per ridurre la dimensionalità di dati. In particolare la PCA ha come obiettivo quello di trovare le direzioni in cui i dati presentano la massima varianza (una maggiore varianza implica una migliore separabilità dei dati) e proiettarli in un nuovo sottospazio con un numero di dimensioni minore rispetto a quello originale. Gli assi ortogonali (componenti principali) del nuovo sottospazio possono essere interpretati come le direzioni in cui si ha la massima varianza con il vincolo che i nuovi assi siano ortogonali l'un l'altro (essendo ortogonali minimizzano la correlazione tra le diverse features e una minore correlazione vuol dire minore ridondanza).

I passi che ho seguito per implementare la PCA sono i seguenti:

- Standardizzo il dataset originale che contiene **D** dati.
- Costruisco la matrice di covarianza e la decompongo in autovettori e autovalori.
- Ordino in modo decrescente gli autovalori per avere un ranking dei corrispondenti autovettori.
- Seleziono i **K** autovettori che corrispondono ai **K** autovalori che spiegano la varianza maggiore, dove **K** è la dimensionalità del nuovo sottospazio delle features (**K** <= **D**).
- Costruisco la matrice **W** con le nuove proiezioni usando i **K** autovettori.
- Trasformo il dataset iniziale di dimensioni **D** nel nuovo sottospazio di dimensione **K**, usando la matrice **W**.

Spesso può accadere che i dati d'origine, che si hanno a disposizione, siano caratterizzati da unità di misura non paragonabili tra loro oppure da ampiezze dei sottocampioni molto diverse. In tali condizioni, non è possibile lavorare con il campione noto ma è necessario standardizzare le variabili. In particolare la PCA nel calcolo dei nuovi assi si basa sulla deviazione standard di ogni variabile, così le variabili con una deviazione standard alta avrebbero un peso maggiore nel calcolo dei nuovi assi rispetto a variabili con deviazione standard minore. Per evitare questo si ricorre alla standardizzazione dei dati, in modo che tutte le variabili abbiano media pari a 0 e deviazione standard pari a 1 e quindi lo stesso peso nel calcolo delle componenti principali.

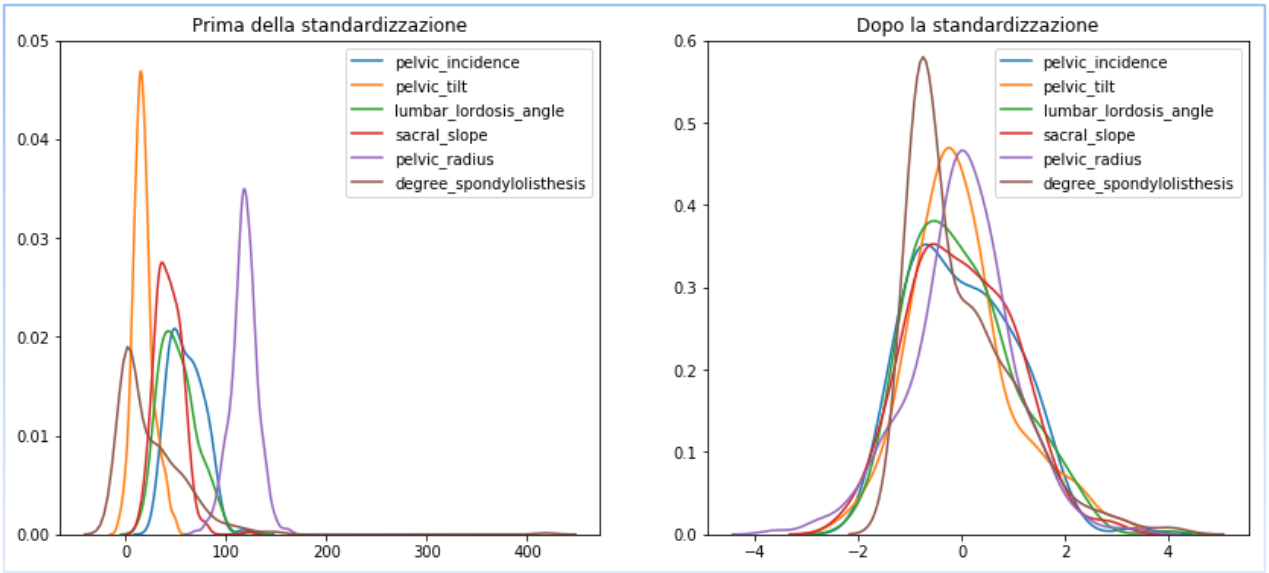


Figura 14: Rappresentazione grafica delle variabili prima e dopo la standardizzazione

Una volta standardizzati i dati mi calcolo la matrice di covarianza ed estraggo gli autovettori che rappresentano le componenti principali (le direzioni di massima varianza) e gli autovalori che ci dicono numericamente quanta varianza è rappresentata nella corrispondente direzione. Nel caso del nostro dataset si ottengono 6 autovettori e 6 autovalori dalla matrice di covarianza 6x6.

Ordinando in modo decrescente gli autovalori è possibile osservare la varianza spiegata da ogni componente principale e la varianza cumulativa.

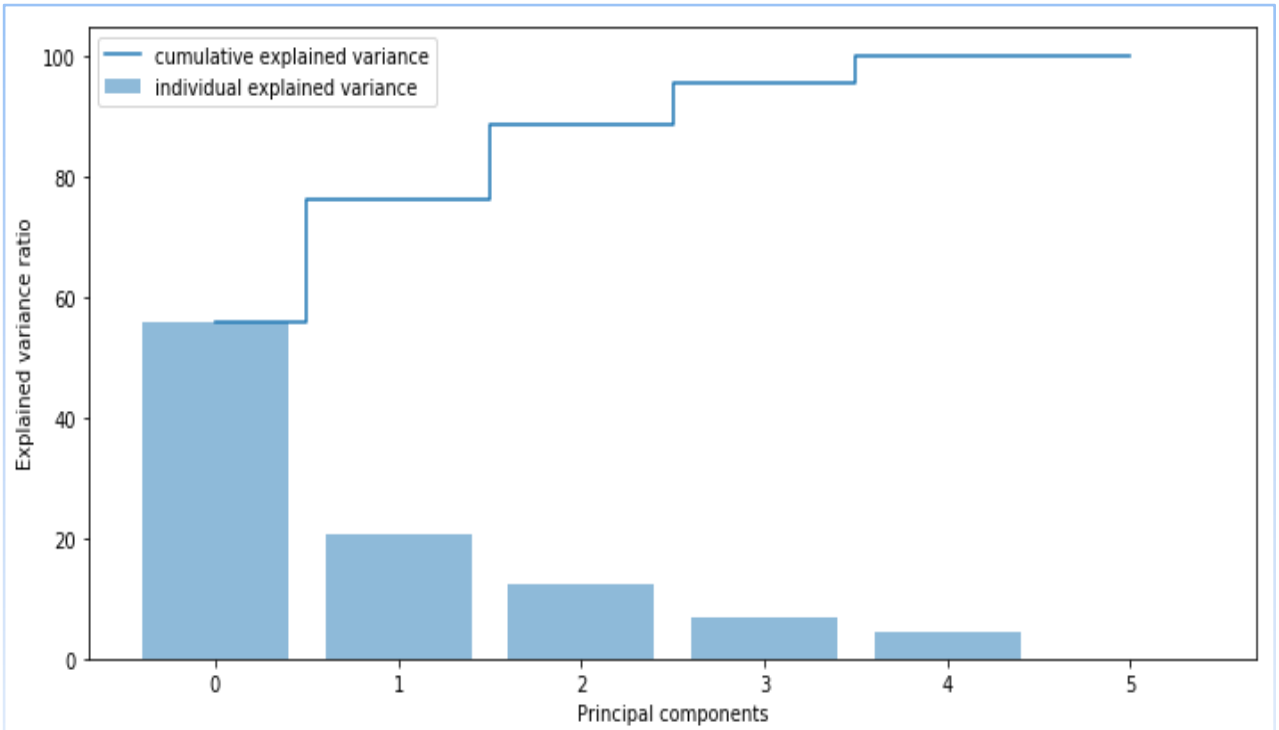
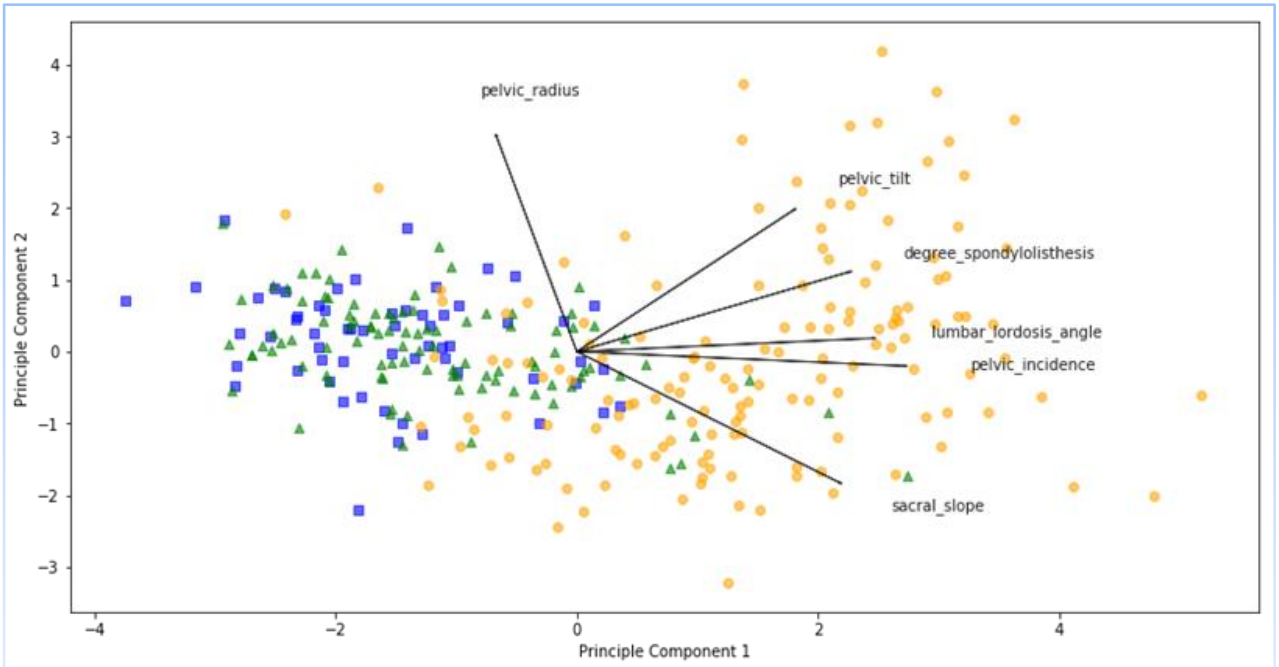


Figura 15: Rappresentazione grafica e numerica della varianza individuale e cumulativa delle componenti principali

Il plot risultante in Figura 15, ci indica come la prima componente principale da sola spiega quasi il 56% della varianza. Le prime tre componenti principali insieme spiegano circa il 90%. Considerando le prime 5 componenti si raggiunge il 100% di varianza spiegata e si può osservare come la sesta componente principale rappresenti valore infinitesimo di varianza spiegata.




```
Features per importanza:

 [(3.07, 'pelvic_radius'), (2.84, 'sacral_slope'), (2.72, 'pelvic_incidence'), (2.67, 'pelvic_tilt'), (2.51, 'degree_spondylo
listhesis'), (2.46, 'lumbar_lordosis_angle')]
```

Figura 16: Rappresentazione grafica e numerica delle singole componenti

Siccome voglio rappresentare i dati in 2 dimensioni scelgo le prime due componenti principali che insieme spiegano una varianza del 76%. Costruisco la matrice **W** utilizzando le prime due componenti principali, grazie alla quale trasformo il dataset iniziale di dimensioni 309x6 in un dataset di dimensioni 309x2, facendo la moltiplicazione tra la matrice dei dati iniziali di dimensioni 309x6 e la matrice **W** di dimensioni 6x2.

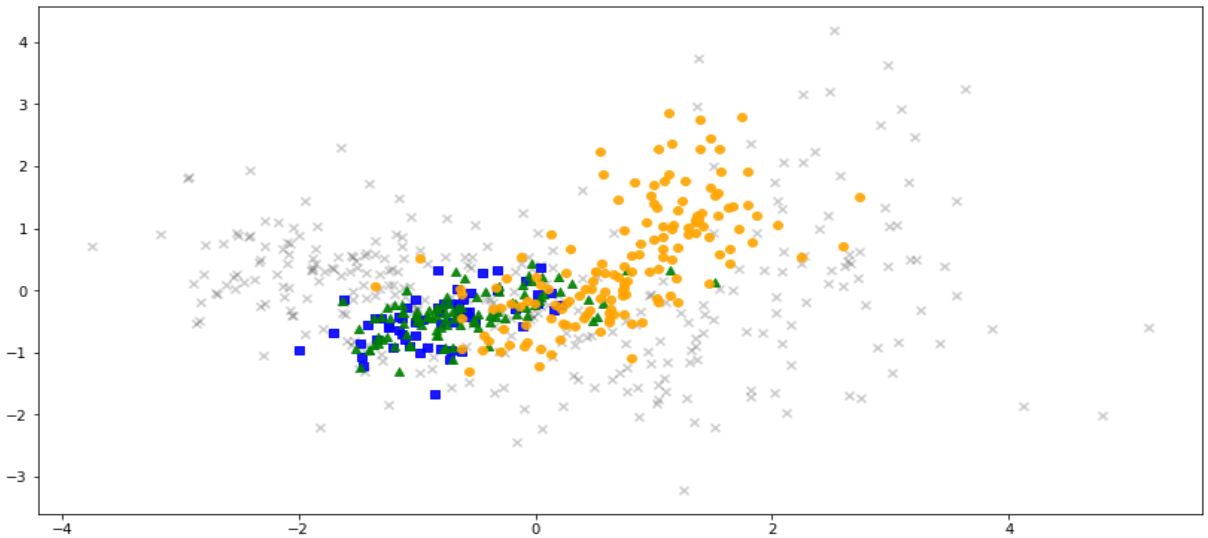


Figura 17: Rappresentazione grafica dei dati prima di applicare la PCA (in sottofondo) e dopo aver applicato la PCA

Per avere un'idea più chiara nella Figura 17 è possibile osservare i dati originali prima di proiettarli rispetto alle due componenti principali(punti scuri a forma di x) e dopo averli proiettati rispetto alle due componenti principali (punti luminosi colorati).

4.2. Linear Discriminant Analysis

La Linear Discriminant Analysis è l'estensione supervisionata della Principal Component Analysis. L'obiettivo della LDA è trovare il sottospazio delle features e quindi le linear discriminant (LD1,LD2) sfruttando la conoscenza delle classi dei dati, in modo da massimizzare lo scatter between classes e minimizzare lo scatter within classes.
Da notare che a differenza della PCA nella LDA non c'è bisogno di normalizzare i dati perché anche se li normalizzassi la suddivisione delle classi non cambierebbe.
I passi che ho seguito per implementare l'algoritmo sono i seguenti:

- Per ogni classe, calcolo il vettore delle medie delle features di dimensione **D**.
- Calcolo le scatter matrices (between-class scatter matrix, within-class scatter matrix).
- Calcolo gli autovettori e i corrispondenti autovalori per le scatter matrices.
- Ordino gli autovettori per autovalori decrescenti e scelgo i **K** autovettori corrispondenti agli autovalori con valore maggiore.
- Costruisco la nuova matrice **W** di dimensioni **D x K** usando i **K** autovettori.
- Trasformo la matrice originale rappresentante i dati nel nuovo sottospazio, usando la matrice degli autovettori **W.**

Come possiamo vedere la LDA è simile alla PCA dal punto di vista della decomposizione della matrice in autovettori e autovalori, che permetteranno di proiettare i dati in un sottospazio delle features. La differenza è che la LDA prende in considerazione le classi con le quali sono etichettati i dati, che sono rappresentate sotto forma di vettore delle medie.
Una volta calcolato i vettori delle medie, procedo calcolando la between-class scatter matrix che rappresenta la distanza tra le medie delle diverse classi e la within-class scatter matrix che rappresenta la distanza tra la media di ogni classe e i suoi campioni.
Una volta calcolate le scatter matrices si può osservare quanta 'class-discriminatory information' è catturata da ciascuna linear discriminant (autovettore).

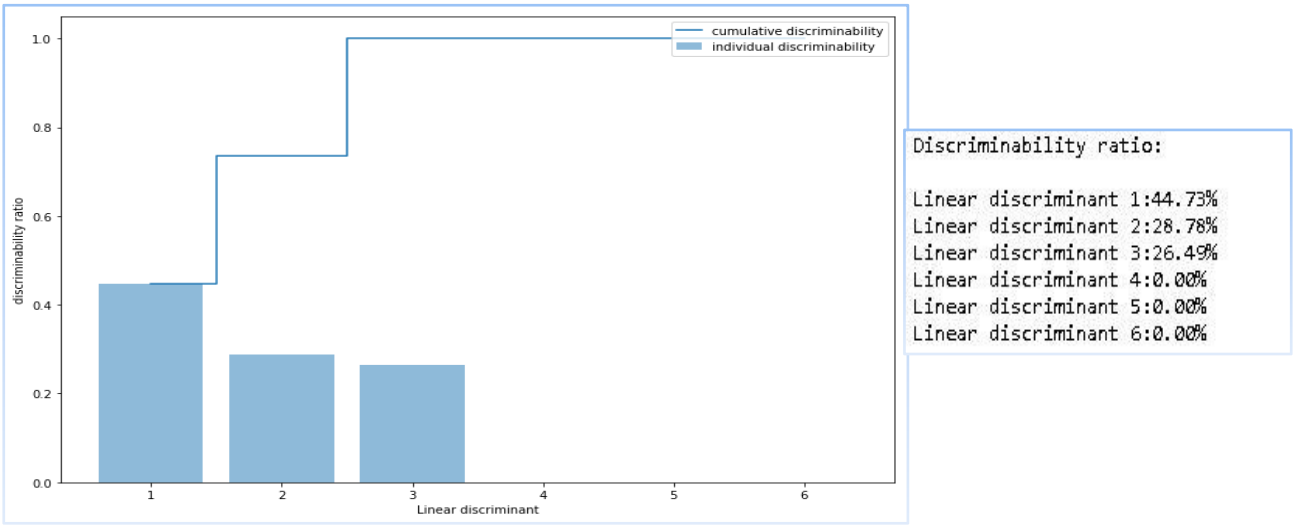


Figura 18: Rappresentazione grafica e numerica della 'discriminability ratio' individuale di ogni linear discriminant

Si può notare come le prime tre linear discriminant rappresentano un valore molto vicino al 100 % di 'discriminability ratio'. Le ultime tre linear discriminant invece rappresentano un valore infinitesimo di 'discriminability ratio'.
Siccome voglio rappresentare i dati in 2 dimensioni scelgo le prime due linear discriminant che insieme rappresentano una 'discriminability ratio' del 73,51%. Costruisco la matrice **W** utilizzando le prime due linear discriminant, grazie alla quale trasformo la matrice originale di dimensioni 309x6 in una matrice di dimensioni 309x2.

Di seguito la Figura 19 e la Figura 20 ci permettono di osservare come vengono separati e rappresentati i dati utilizzando le due tecniche di feature extraction (PCA e LDA).

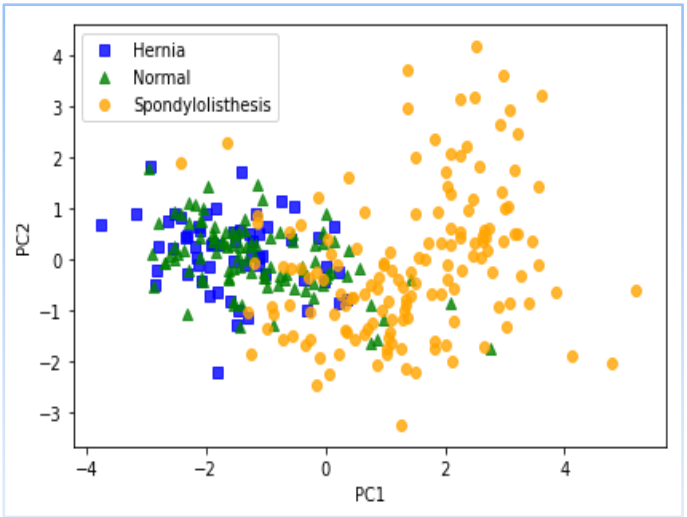


Figura 19: Proiezione dei dati, tramite scatter-plot, dopo aver applicato la PCA

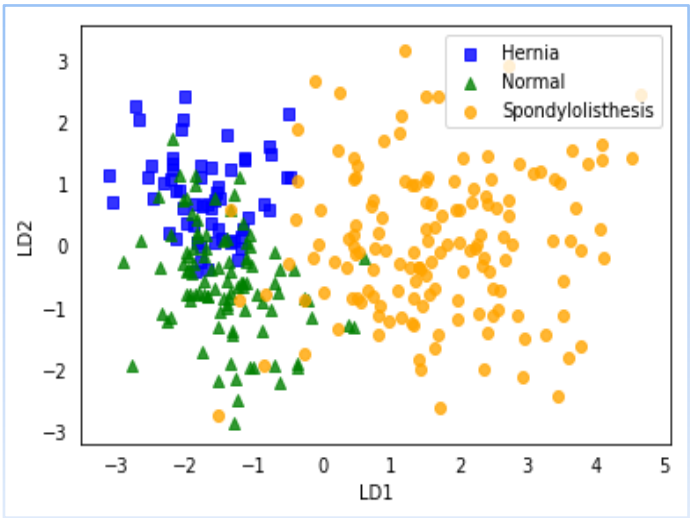


Figura 20: Proiezione dei dati, tramite scatter-plot, dopo aver applicato la LDA

Dal confronto dei grafici si può vedere come con la LDA si abbia una più chiara separazione dei dati in base alla classe di appartenenza, che non era per niente scontato visto che abbiamo un dataset di dimensioni ridotte.

5. Analisi dei dati mediante tecniche di classificazione

Al fine di predire a quale classe appartiene un paziente, ho implementato diverse tecniche di classificazione per stabilire quale si adatta meglio al dataset.
Le tecniche di classificazione che ho usato sono le seguenti:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbors

Prima di applicare le tecniche di classificazione ho diviso il dataset in due set: il training set che utilizzerò per addestrare il modello di classificazione e il test set che mi permetterà di valutare la performance del modello.

5.1. Cross-validation

La tecnica che ho utilizzato per suddividere il dataset in training set e test set è la cross-validation. In particolare, ho usato la k-fold cross-validation che consiste nel dividere il dataset originale in k (ho scelto k=10) parti di uguali numerosità e , ad ogni passo, la k-esima parte del dataset viene ad essere il test set, mentre la restante parte costituisce il training set. Questo processo viene ripetuto k volte, ogni volta con un diverso sottoinsieme riservato per la valutazione (ed escluso dall'addestramento). Dalla media dei valori predittivi ottenuti nei k esperimenti si calcola la media cumulativa finale che misura l'accuratezza del modello.
La cross-validation è una tecnica più robusta rispetto alla suddivisione del dataset in modo random in due sottoset (training set e test set), in cui si corre il rischio di utilizzare come training set e test set parti del dataset non veramente rappresentativi dell'intero dataset.

La Figura 21, rappresenta schematicamente il funzionamento della cross-validation.

Figura 21: Rappresentazione schematica del funzionamento della cross-validation

5.2. Logistic Regression

Quando la variabile risposta Y assume soltanto valori compresi tra 0 e 1 risulta essere inappropriato l'utilizzo di modelli di regressione lineare in quanto nella sua formulazione ($Y = \alpha + \beta X$), il modello lineare implica che i valori della variabile risposta possano andare da $-\infty$ a $+\infty$. In pratica la regressione logistica è una tecnica che consiste in una divisione binaria delle variabili, cioè un risultato può appartenere alla classe 0 o alla classe 1.

Nei modelli di regressione logistica si usa la cosiddetta funzione logistica:

$$P(Y = 1) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

con un pò di manipolazioni si ottiene:

$$\frac{P(Y = 1)}{1 - P(Y = 1)} = e^{\beta_0+\beta_1 X}$$

La quantità $\frac{P(Y=1)}{1-P(Y=1)}$ è chiamata odds. Dato un valore compreso tra 0 e + ∞, l'odds ci restituisce un valore compreso tra 0 e 1.
Applicando il logaritmo naturale dell'odds si ottiene:

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 X$$

che prende il nome di logit o log-odds ed è lineare in in X.

Nella regressione logistica dunque il coefficiente β_1 è legato alla variazione del logit e non alla variazione della probabilità, cui è legato non linearmente, che varia entro l'intervallo [0,1]. Il valore di β_1 ci indica se la P(Y =1) cresce ($\beta_1 > 0$) o decresce ($\beta_1 < 0$) al crescere di X.

Essendo che nel dataset abbiamo che i dati appartengono a 3 classi diverse, intuitivamente ci si aspetta che la regressione logistica multinomiale, che è un'estensione della regressione logistica binaria che permette di classificare i dati in più di 2 classi, funzioni meglio della regressione logistica binaria.

	Classificatore	train_score	test_score
0	MultinomialLogisticRegression	0.874494	0.854839
1	LogisticRegression	0.834008	0.83871

Figura 22: Confronto tra i risultati ottenuti usando la Regressione Logistica Binaria e la Regressione Logistica Multinomiale

Analizzando i risultati ottenuti otteniamo che nel caso della Regressione Logistica Multinomiale si ha una classificazione migliore infatti si ha un apprendimento migliore da parte del modello in fase di training e in fase di test quasi il 4% in più di dati classificati correttamente rispetto all'uso della regressione logistica binaria.

5.3. Decision Tree

La tecnica del Decision Tree consiste nel classificare i dati facendo uso di un albero decisionale di profondità finita. Ogni nodo dell'albero consiste in un test che coinvolge un attributo (cioè una delle variabili biomeccaniche), e ogni ramo discendente da ogni nodo rappresenta uno dei possibili

risultati del test(cioè uno dei possibili valori dell'attributo). Quindi classificare utilizzando la tecnica del Decision Tree significa fare una serie di test, iniziando dal nodo radice e terminando con il nodo foglia. Al fine di determinare l'attributo che classifica al meglio i dati si possono utilizzare varie misure tra cui il Gini index che misura l'impurità/disordine del dataset corrispondente al nodo **t**. Possiamo definire il Gini index di un certo nodo **t** dell'albero in costruzione, e rispetto alla corrispondente partizione del dataset di training, con la seguente formula:

$$\mathbf{GINI}\ (\mathbf{t}) = 1 - \sum_{j} [p(j|t)]^2$$

dove p(j|t) rappresenta la frequenza relativa della classe j al nodo t.

	Classificatore	train_score	test_score
0	DecisionTreeClassifier (max_depth=3)	0.861564	0.825161
1	DecisionTreeClassifier (max_depth=5)	0.928443	0.795914

Figura 23: Confronto tra i risultati ottenuti utilizzando Decision Tree di profondità diversa

Dalla Figura 23 si osservano i risultati ottenuti utilizzato due livelli di profondità dell'albero diverso, nel primo caso vediamo un albero con profondità 3 mentre nel secondo caso un albero con profondità 5. Si può osservare che aumentando la profondità il classificatore tende all'overfitting, infatti cresce il train_score e diminuisce il test_score. Con una profondità dell'albero di decisione pari a 3 si ottiene un risultato apprezzabile infatti abbiamo un buono score in fase di training del modello, che differisce dal test score di una misura poco superiore al 3%.
La Figura 24 illustra graficamente l'albero decisionale:

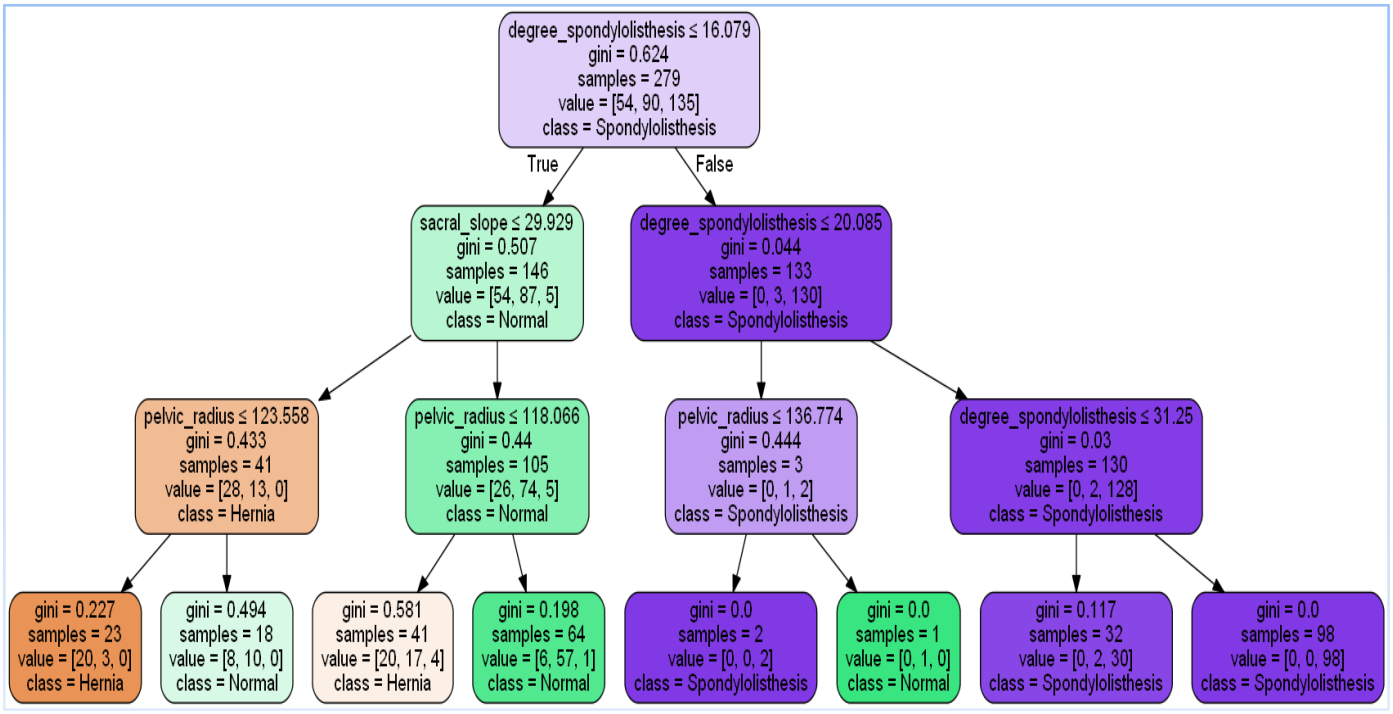


Figura 24: Rappresentazione grafica dell'albero decisionale di profondità 3

Il risultato ottenuto in Figura 24 ci illustra un albero decisionale con 8 nodi terminali. Ogni nodo ci dà le seguenti informazioni:

- **split**: criterio di suddivisione
- **gini**: spiega il grado di impurità del nodo, per valori bassi il nodo tende ad essere più puro e quindi tutti i nodi tendono ad appartenere ad una sola classe (informazione più interessante), mentre per valori del Gini index più alto abbiamo disordine maggiore e i record tendono ad essere distribuiti tra le diverse classi (informazione meno interessante).
- **samples**: rappresenta il numero di campioni che ogni nodo prende in considerazione
- **value**: rappresenta il numero di campioni appartenenti a ciascuna delle 3 classi
- **class**: rappresenta la classe che maggiormente rappresenta un determinato nodo.

Si può osservare come l'attributo "degree_spondylolisthesis" tende a dividere nei due rami principali la classificazione dei campioni della classe "Spondylolisthesis" rispetto alle altre due classi. Infatti per valori maggiori di 16 dell'attributo "degree_spondylolisthesis" abbiamo tendenzialmente che i campioni appartengono alla classe "Spondylolisthesis", invece per valori più piccoli di 16 vengono classificate le altre due classi "Hernia" e "Normal".

5.4. K-Nearest Neighbors

L'algoritmo di classificazione K-Nearest Neighbors si basa sul concetto di classificare un campione incognito considerando la classe dei k campioni più vicini all'insieme di addestramento. Al nuovo campione verrà assegnata l'etichetta della classe a cui appartengono la maggior parte dei k campioni più vicini. Il grande vantaggio di questo algoritmo sta nel fatto che il modello è il training set, riducendo quindi a zero il tempo di creazione del modello e inoltre è un algoritmo incrementale. Tra gli svantaggi ci sono il fatto che è molto suscettibile alla presenza di outliers. La scelta di k è molto importante affinché il campione venga assegnato alla classe corretta; se k è troppo piccolo, la classificazione può essere sensibile al rumore, se k è troppo grande invece la classificazione può essere computazionalmente costosa e l'intorno può includere campioni appartenenti ad altre classi.

La Figura 25 rappresenta come varia l'accuratezza del classificatore in base al numero di k scelto.
Si può notare che scegliendo K=8 si ottiene l'accuratezza migliore che è pari a circa l'85 %.

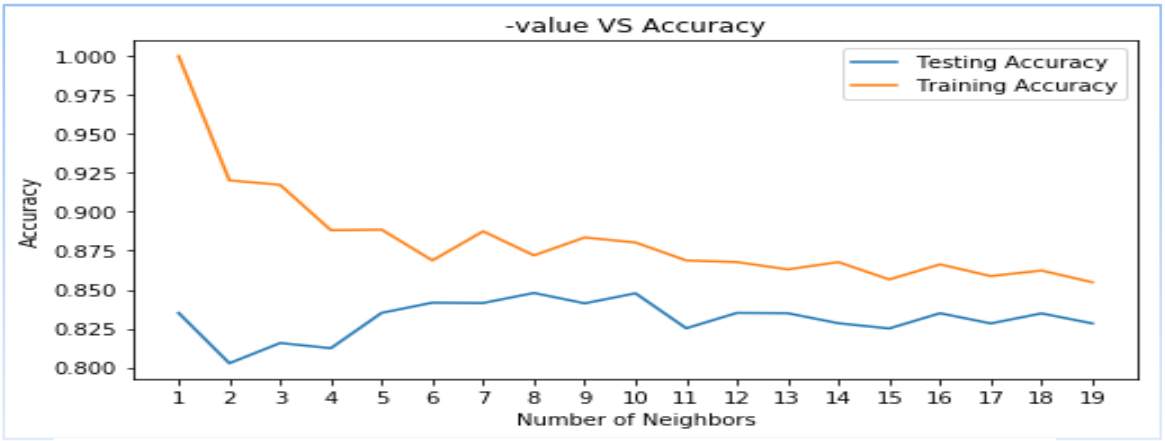


Figura 25: La figura rappresenta come varia l'accuratezza del classificatore al variare del parametro k

Ho provato a verificare l'algoritmo di classificazione usando 3 valori di k diversi (k=3,k=8,k=16):

	Classificatore	train_score	test_score
0	KNeignborsClassifier (k=3)	0.9173	0.815699
1	KNeignborsClassifier (k=8)	0.871988	0.847849
2	KNeignborsClassifier (k=16)	0.866235	0.834839

Figura 26: Confronto tra i risultati ottenuti utilizzando K-Nearest-Neighbors con 3 diversi risultati di k

Considerando i risultati ottenuti illustrati in Figura 26 , nel caso di k=3 si può notare che c'è una differenza di circa il 10% tra train_score e test_score e questo fa pensare che per k=3 il classificatore tende a creare un modello troppo accurato in fase di training che poi non mantiene i risultati promessi in fase di test . Per k=16 si ha un buono risultato, ma il miglior risultato si ottiene per k=8 con un buono score e una differenza percentuale di poco superiore al 3% tra train_score e test_score.

6. Ulteriori analisi

Per trarre ulteriori analisi riguardanti i modelli di classificazione scelti, ho utilizzato la Learning Curve e la Confusion Matrix. Da quest'ultima è possibile derivare altre misure come la "Recall" e la "Precision", per valutare le performance delle tecniche di classificazione utilizzate.

6.1. Learning Curve

La Learning Curve (curva di apprendimento) ci permette di rappresentare graficamente come varia l'accuratezza del nostro classificatore al crescere del numero di campioni preso in considerazione. Plottando l'accuratezza del training set e del test set possiamo facilmente capire se il nostro modello soffre di un'alta varianza o di un alto bias, e quindi se è possibile risolvere il problema aumentando il numero di dati dai quali il modello può apprendere. Di seguito sono plottate le curve di apprendimento per i tre classificatori presi in considerazione:

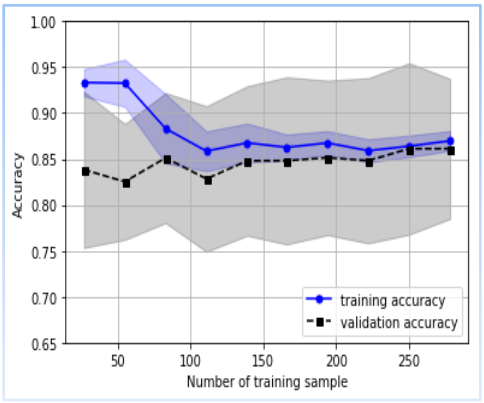


Figura 27: Learning Curve con classificatore Multinomial Logistic Regression

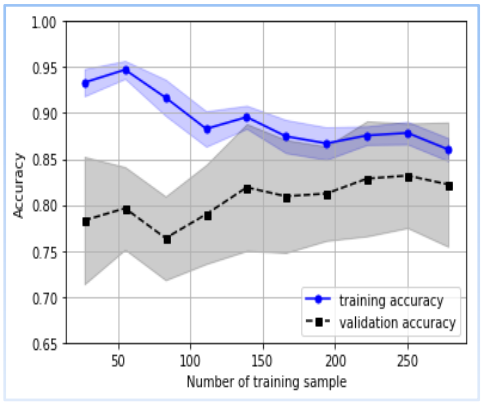


Figura 29: Learning Curve con classificatore Decision Tree con profondità 3

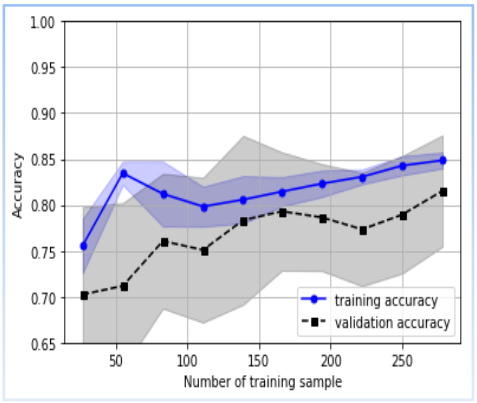


Figura 28: Learning Curve con classificatore K-NN (k=8)

Per quanto riguarda la Figura 27, in cui usiamo il Multinomial Logistic Regression, osserviamo che il nostro modello lavora veramente bene già quando abbiamo un set di dati di 150 campioni e si può notare come tende ad aumentare l'accuratezza con il crescere delle dimensioni del dataset.

La Figura 28, in cui usiamo il classificatore Decision Tree con profondità 3, ci illustra come questa tecnica di classificazione inizialmente soffre di overfitting visto il gap marcato che c'è tra le due curve di accuratezza; in particolare notiamo come questo gap tende ad assottigliarsi con il crescere del numero di campioni presi in considerazione, fino a raggiungere un'accuratezza di quasi l'85% quando abbiamo preso in considerazione più di 250 campioni.

La figura 29 ci illustra che il classificatore K-Nearest-Neighbors, raggiunge la sua maggiore accuratezza quando abbiamo preso in considerazione circa 150 campioni per poi perdere accuratezza. Con l'aumentare del numero di dati presi in considerazione sembra che l'accuratezza del training set e del test set tendono a convergere di nuovo. In questo caso con un dataset più grande ci saremmo potuti togliere qualche dubbio e capire se le due curve andrebbero a convergere di nuovo.

6.2. Confusion Matrix , Recall e Precision

La confusion matrix è una matrice o tabella, che ci dice se un certo campione è stato classificato correttamente o se è stato classificato come appartenente ad un'altra classe e quindi possiamo valutare anche in base a queste stime le performance del nostro modello.

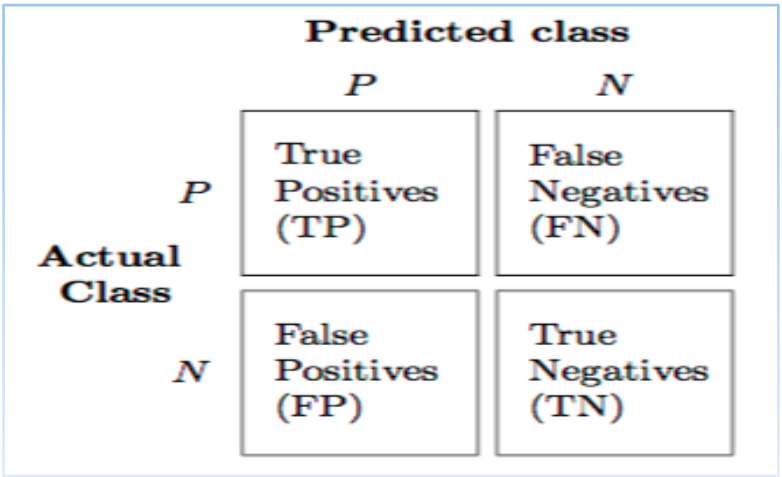


Figura 30: Rappresentazione della Confusion Matrix per classificazione binaria

La Figura 30 illustra schematicamente la Confusion Matrix. La definizione di "positivo" o "negativo" è convenzionale, solitamente si definisce come "positiva" la classe alla quale siamo maggiorente interessati.

In particolare i valori di ogni cella hanno il seguente significato:

- **True Positives** (TP): casi positivi correttamente identificati
- **True Negatives** (TN): casi negativi correttamente identificati
- **False Positives** (FP): casi negativi identificati come positivi
- **False Negatives** (FN): casi positivi identificati come negativi

Successivamente dalla Confusion Matrix ho ricavato “Precision” e “Recall”, che sono due misure per avere informazioni aggiuntive sulla performance del nostro modello.

$$\textbf{RECALL} = \frac{TP}{\text{TP} + \text{FN}}$$

$$\textbf{PRECISION} = \frac{TP}{\text{TP} + \text{FP}}$$

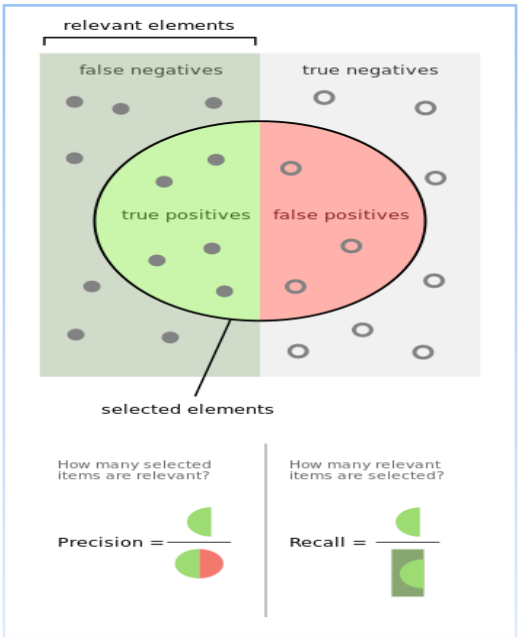


Figura 31: Rappresentazione grafica di Recall e Precision

La “ Precision” ci dice quanti dei campioni selezionati sono rilevanti, mentre la “Recall” o “Sensitivity” ci dice quanti campioni rilevanti sono stati selezionati.

Di seguito sono presentate le Confusion Matrix per ognuno dei classificatori utilizzati:

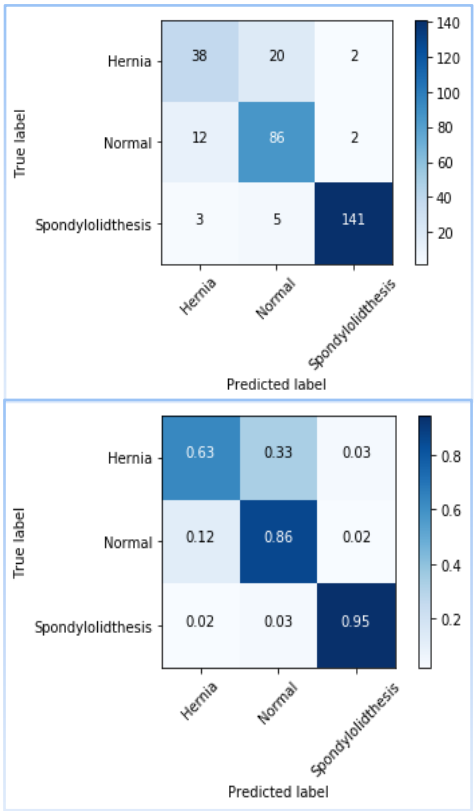


Figura 32 : Rappresentazione della Confusion Matrix per il modello che usa il Multinomial Logistic Regression (in alto la matrice non è normalizzata, in basso la matrice è normalizzata)

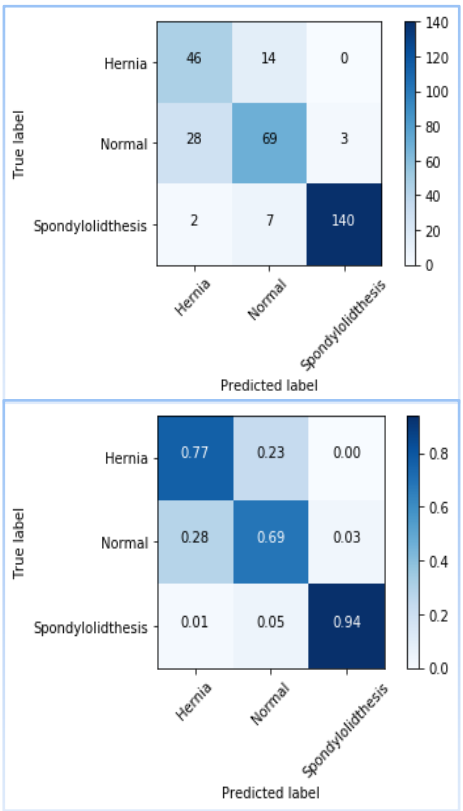


Figura 33: Rappresentazione della Confusion Matrix per il modello che usa il Decision Tree con profondità 3 (in alto la matrice non è normalizzata, in basso la matrice è normalizzata)

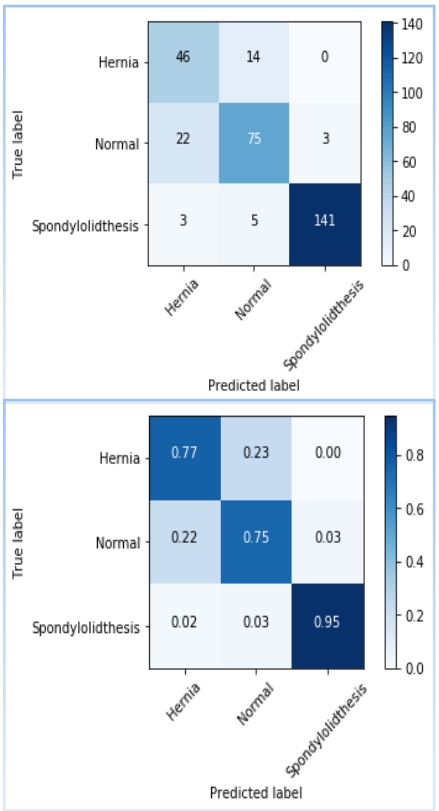


Figura 34: Rappresentazione della Confusion Matrix per il modello che usa il K-NN con k=8 (in alto la matrice non è normalizzata, in basso la matrice è normalizzata)

Dalle Confusion Matrix in Figura 32, osserviamo che il classificatore Multinomial Logistic Regression classifica correttamente il 95% dei dati per la classe “Spondylolisthesis”, l’86% per la classe “Normal” e il 63% per la classe “Hernia”.

	Recall %	Precision %
Hernia	63.333333	71.698113
Normal	86.000000	77.477477
Spondylolisthesis	94.630872	97.241379

Figura 35: Risultati della Recall e Precision, separatamente per ogni classe, con classificatore Multinomial Logistic Regression

In media abbiamo una “Recall” pari all’81,32 % e una “Precision” pari all’82,13% usando il classificatore Multinomial Logistic Regression.

Le Confusion Matrix in Figura 33, ci dicono che il classificatore Decision Tree classifica correttamente il 94% dei campioni della classe “Spondylolisthesis”, il 69% per la classe “Normal” e il 77% per la classe “Hernia”.

	Recall %	Precision %
Hernia	76.666667	60.526316
Normal	69.000000	76.666667
Spondylolisthesis	93.959732	97.902098

Figura 36: Risultati della Recall e Precsion, separatamente per ogni classe, con classificatore Decision Tree di profondità 3

Usando come classificatore il Decision Tree con profondità 3, in media abbiamo una “Recall” pari al 79,87 % e una “Precision” pari al 78,36%.

Dalle Confusion Matrix in Figura 34, possiamo osservare che il classificatore K-NN, classifica correttamente il 95% dei casi per la classe “Spondylolisthesis”, il 75% dei casi per la classe “Normal” e il 77% per la classe “Hernia”.

	Recall %	Precision %
Hernia	76.666667	64.788732
Normal	75.000000	79.787234
Spondylolisthesis	94.630872	97.916667

Figura 37: Risultati della Recall e Precsion per ogni classe separatamente con classificatore K-NN con k=8

In media abbiamo una “Recall” pari all’82,09% e una “Precision” pari all’80,82% usando il classificatore K-NN con k=8.

7. Conclusioni

Attraverso le tecniche di feature extraction utilizzate per ridurre la dimensionalità dei dati, ed essendo che il mio obiettivo è rappresentare i dati attraverso due componenti principali, si ottiene un risultato discreto utilizzando la PCA mentre si ha un risultato molto buono utilizzando la LDA.

	Classificatore	train_score	test_score
0	MultinomialLogisticRegression	0.673499	0.670108
1	LogisticRegression	0.673141	0.663656
2	DecisionTreeClassifier (max_depth=3)	0.726005	0.686129
3	KNeignborsClassifier (k=3)	0.792874	0.618387
4	KNeignborsClassifier (k=8)	0.726362	0.647204
5	KNeignborsClassifier (k=16)	0.711257	0.663656

Figura 39: Rappresentazione dei risultati ottenuti per ogni classificatore, dopo aver ridotto la dimensionalità tramite PCA

	Classificatore	train_score	test_score
0	MultinomialLogisticRegression	0.846818	0.838172
1	LogisticRegression	0.849697	0.841613
2	DecisionTreeClassifier (max_depth=3)	0.876666	0.851183
3	KNeignborsClassifier (k=3)	0.901118	0.793011
4	KNeignborsClassifier (k=8)	0.865875	0.83828
5	KNeignborsClassifier (k=16)	0.868393	0.85129

Figura 38: Rappresentazione dei risultati ottenuti per ogni classificatore, dopo aver ridotto la dimensionalità tramite LDA

Dal confronto tra la Figura 38 e la Figura 39 si può osservare come utilizzando la LDA si hanno dei risultati nettamente migliori rispetto all'utilizzo della PCA.

Inoltre si può osservare dalla Figura 40, come i risultati ottenuti utilizzando solo le prime due componenti della LDA siano molto vicini ai risultati che si ottengono utilizzando il dataset originale, senza aver applicato tecniche di feature extraction.

	Classificatore	train_score	test_score
0	MultinomialLogisticRegression	0.846818	0.838172
1	LogisticRegression	0.849697	0.841613
2	DecisionTreeClassifier (max_depth=3)	0.876666	0.851183
3	KNeignborsClassifier (k=3)	0.901118	0.793011
4	KNeignborsClassifier (k=8)	0.865875	0.83828
5	KNeignborsClassifier (k=16)	0.868393	0.85129

Figura 41: Riproposizione della Figura 39, per effettuare un confronto con la Figura 40

	Classificatore	train_score	test_score
0	MultinomialLogisticRegression	0.875582	0.857634
1	LogisticRegression	0.837828	0.812258
2	DecisionTreeClassifier (max_depth=3)	0.861564	0.825161
3	KNeignborsClassifier (k=3)	0.9173	0.815699
4	KNeignborsClassifier (k=8)	0.871988	0.847849
5	KNeignborsClassifier (k=16)	0.866235	0.834839

Figura 40: Rappresentazione dei risultati ottenuti per ogni classificatore, senza aver utilizzato tecniche di feature extraction

In alcuni casi si hanno dei risultati migliori utilizzando i dati dopo averli trasformati con la LDA, come nel caso della Logistic Regression binaria, nel caso del Decision Tree di profondità 3 e nel caso di K-NN con K=16. Si hanno anche dei risultati peggiori come nel caso del modello che utilizza K-NN con k=3 e k=8 e nel caso della Multinomial Logistic Regression.

Considerato che non c'è molta differenza nei risultati, è conveniente ridurre la dimensionalità dei dati attraverso la LDA, sia perché permette di migliorare l'efficienza computazionale dell'algoritmo di apprendimento, sia perché a volte permette di migliorare la performance in fase di predizione .

Tuttavia ho potuto costatare che i pazienti etichettati con la classe “Hernia” e “Normal” hanno caratteristiche simili e non c’è una suddivisione netta tra le due classi mentre i pazienti etichettati con la classe “Spondylolisthesis” risultano più separati rispetto alle altre due classi.
In generale, per il dataset analizzato, le tecniche utilizzate sono abbastanza efficienti, infatti nella maggior parte dei casi l’accuratezza si attesta intorno all’85%.

I modelli generati saranno quindi in grado di prevedere la classe dei pazienti per nuovi record che verranno aggiunti al dataset, a patto che abbiano la stessa struttura con stessi attributi e classi.