INGEGNERIA DELLA CONOSCENZA

“Supporto donne in maternità”

Autori

Gianluca Colella 719762 g.colella27

Antonio Ciriolo 718856 a.ciriolo3

GitHub: https://github.com/GianluDR/ICONProject

Introduzione

Lo scopo principale della nostra applicazione è quello di fornire un sostegno alle donne durante il periodo di maternità nel riconoscimento dei rischi per la propria salute o quella del bambino, specialmente nelle zone meno provviste di assistenza sanitaria. È stato utilizzato appunto un dataset di dati rilevati da dispositivi IoT in vari ospedali e cliniche di aree rurali del Bangladesh. L’obiettivo è riconoscere e valutare la gravità dei problemi e, se necessario, offrire consigli e raccomandazioni appropriate. Per raggiungere questo, abbiamo sviluppato un’applicazione in Python che combina un sistema esperto basato su base di conoscenza con l’impiego di diversi algoritmi di apprendimento, sia supervisionato che non supervisionato.

Analisi dei dati:

Abbiamo recuperato i dati dal seguente dataset, strutturati nella seguente maniera:

https://archive.ics.uci.edu/dataset/863/maternal+health+risk

- Age(Età)
- SystolicBP -> SystolicBloodPressure(Pressione sistolica)
- DiastolicBP -> DiastolicBloodPressure(Pressione diastolica)
- BS -> BloodSugar(Glicemia)
- BodyTemp(Temperatura corporea)
- HeartRate(Battiti al minuto)
- RiskLevel(Livello di rischio salute)

In un secondo momento, abbiamo eseguito una verifica per individuare la presenza di valori mancanti o duplicati nei dati. Non abbiamo riscontrato valori mancanti, ma abbiamo identificato la presenza di ben 562 duplicati. Prima di prendere una decisione definitiva, abbiamo condotto un'analisi completa dell'intero dataset e abbiamo effettuato numerosi test sia includendo che escludendo i duplicati. Tenendo conto del fatto che è possibile che più individui abbiano effettivamente avuto registrazioni identiche, abbiamo optato per mantenerli all'interno del dataset. In seguito, abbiamo eseguito una verifica alla ricerca di valori anomali mediante il calcolo dello z-score per ciascuna colonna contenente valori numerici (ad eccezione di RiskLevel).

$$\textit{Z-score} = (x - \textit{mean}) / \textit{std. Deviation}$$

Lo z-score o standard score è una misura statistica utilizzata per valutare quanto un dato valore si discosti dalla media di un insieme di dati. In modo più preciso, lo z-score ci fornisce informazioni sul

numero di deviazioni standard con cui un punto dati si trova al di sopra o al di sotto della media.

Abbiamo individuato alcune anomalie nei valori: sono state riscontrate leggere elevatezze nella glicemia e nella temperatura corporea rispetto alla norma. Tuttavia, potrebbero essere delle effettive misurazioni reali, ma per precauzione, abbiamo effettuato un secondo controllo utilizzando l'algoritmo di clustering K-means per identificare eventuali altri valori anomali.

PCA

La Principal Component Analysis è una tecnica utilizzata per semplificare i dati. Consente di calcolare un insieme di componenti principali in base a un gruppo di variabili originali, combinandole linearmente mediante coefficienti ottenuti dalla matrice degli autovalori della matrice di correlazione delle variabili. Queste componenti principali sono non correlate tra loro, hanno varianza pari agli autovalori della matrice di correlazione e conservano l'informazione iniziale dei dati. La prima componente principale cattura la massima varianza nei dati, la seconda cattura la varianza residua massima e così via. Questo permette di ridurre la dimensionalità dei dati utilizzando le prime componenti principali (con un numero inferiore di variabili rispetto a quelle originali), conservando comunque una percentuale significativa dell'informazione iniziale.

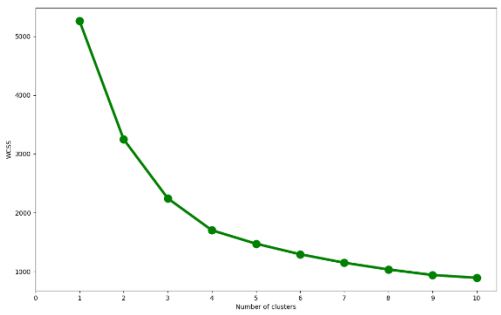
Nel nostro caso, abbiamo utilizzato PCA per ridurre lo spazio delle variabili a tre dimensioni con l'obiettivo di rendere possibile una visualizzazione grafica dei dati. Questa riduzione dimensionale ci ha permesso di proiettare i dati in uno spazio tridimensionale, dove è più agevole individuare eventuali outliers evidenti presenti nel

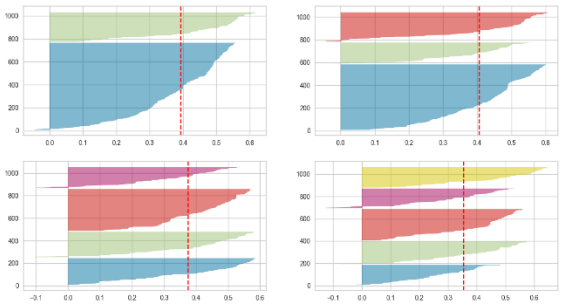
database. Inoltre, abbiamo usato il dataset standardizzato (più avanti viene spiegato come) per migliorare le prestazioni di K-means.

K-means

Il K-Means è un algoritmo che organizza dati simili in gruppi, trovando punti centrali chiamati "centroidi" e assegnando dati a ciascun gruppo in base alla loro vicinanza ai centroidi. È utile per trovare pattern o cluster all'interno di un insieme di dati. Può essere utilizzato per individuare outliers in quanto, durante il processo di clustering, i punti dati che sono significativamente distanti dai centroidi dei cluster possono essere considerati potenziali outliers. In altre parole, i punti che non si adattano bene a nessun gruppo potrebbero essere identificati come valori anomali.

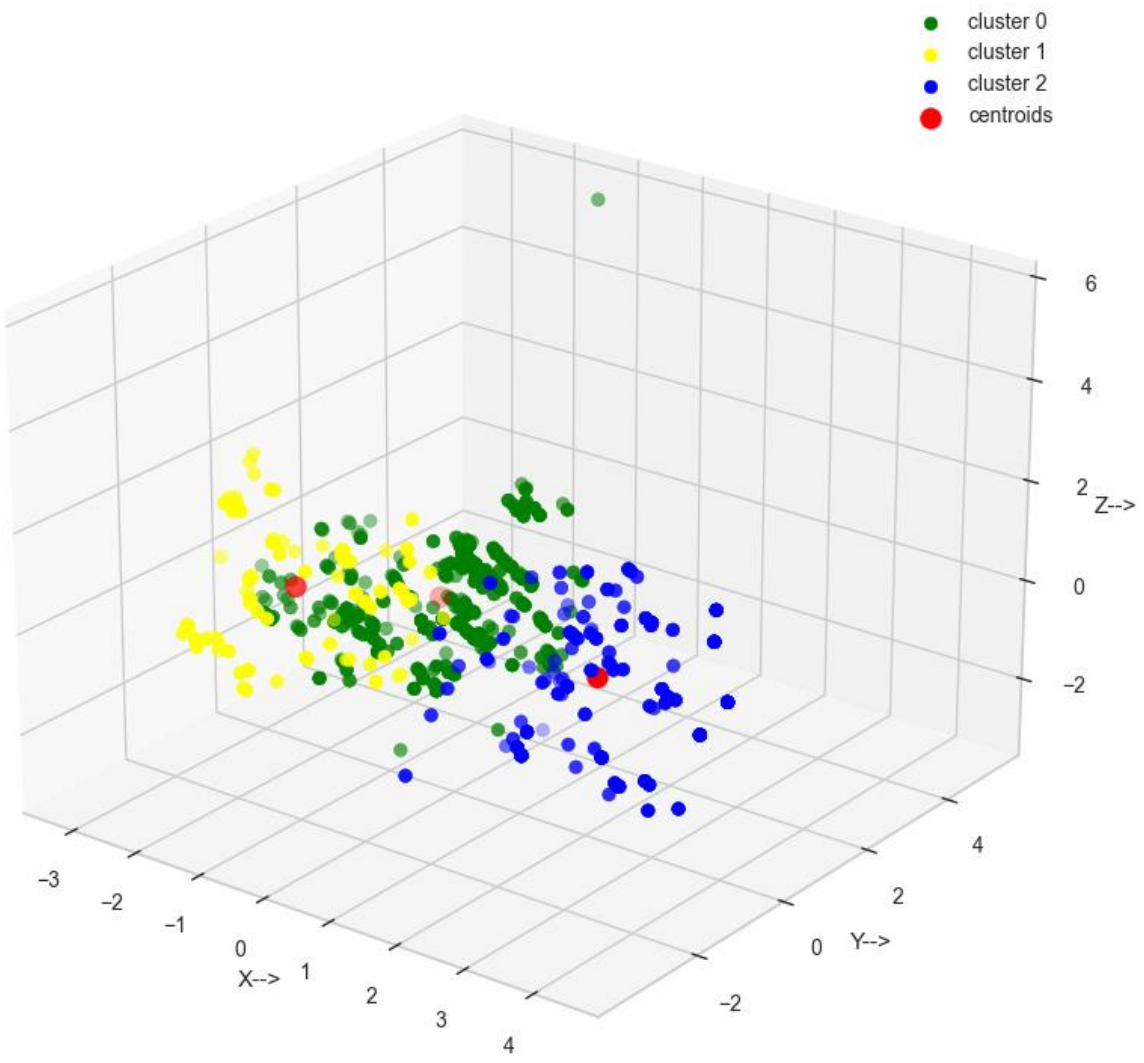
Metodo del gomito e silhouette score





Prima abbiamo usato il metodo del gomito che si basa sull'osservazione della variazione della somma dei quadrati delle distanze tra i punti dati e i centroidi dei cluster al variare del numero di cluster. Quando si traccia questa variazione in un grafico, sembrerà simile a un braccio con un gomito. Il punto in cui la curva inizia a livellarsi rappresenta il numero ottimale di cluster da utilizzare. Nel nostro caso sembrerebbe 4.

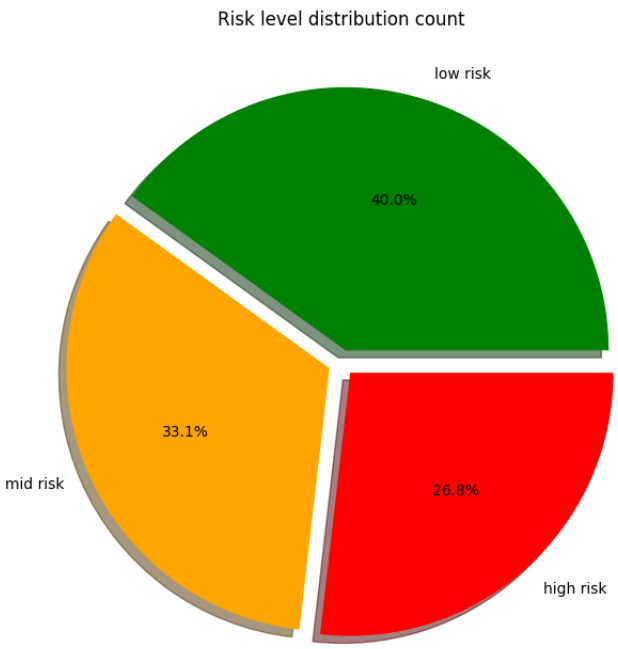
Successivamente abbiamo eseguito il K-Means con diversi valori di K e calcolato il Silhouette Score per ciascun K. Il valore di K che massimizza il Silhouette Score indica il numero ottimale di cluster da utilizzare: un punteggio più alto significa una migliore separazione dei cluster e quindi un K migliore. Il suo valore varia da -1 a 1, nel nostro caso i K miglior sono 3 e 4, con rispettivamente circa 0.406 e 0.375, un punteggio di Silhouette più alto indica un migliore clustering quindi abbiamo preferito scegliere K=3.



Dall'analisi del grafico generato con il K-Means, abbiamo individuato un punto dati che si trova significativamente distante rispetto agli altri punti dati dei cluster (ai centroidi dei cluster).

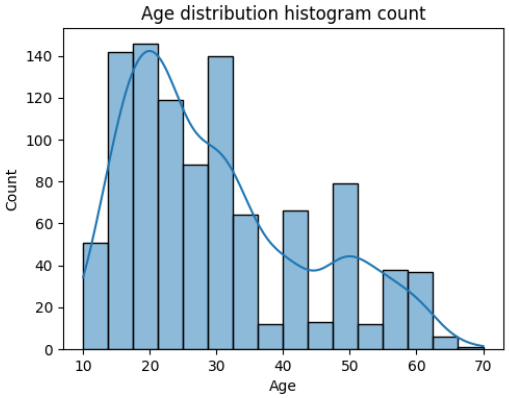
Questa posizione anomala suggerisce che potrebbe trattarsi di un outlier all'interno del dataset. Tuttavia, per capire quale dato specifico corrisponde a un outlier, sarebbe utile osservare e analizzare le caratteristiche individualmente.

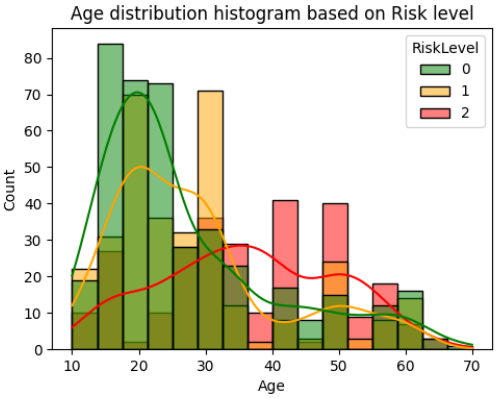
Passiamo quindi all’analisi grafica dei dati:

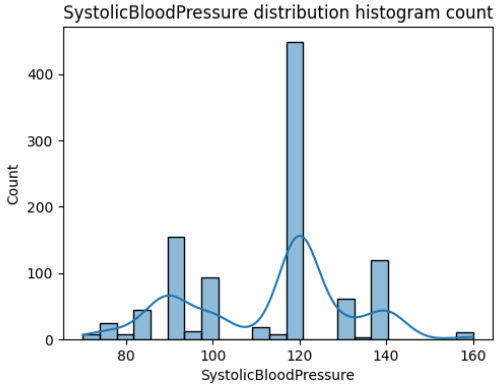


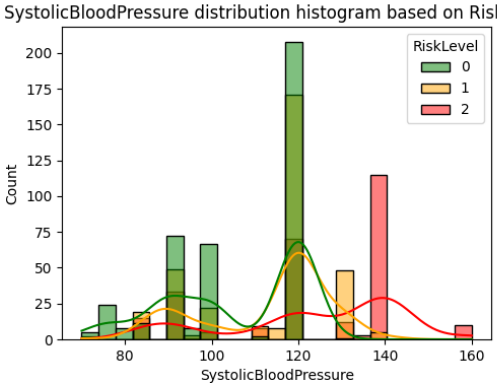
Sapendo che Risk Level è il nostro target, innanzitutto abbiamo cambiato il suo valore nel database da stringhe a valori numerici:

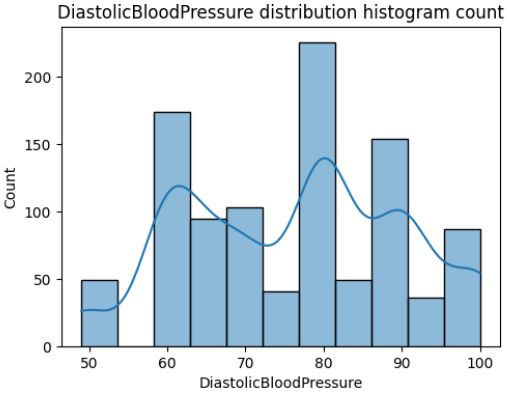
Low risk: 0 – Mid risk: 1 – High risk: 2

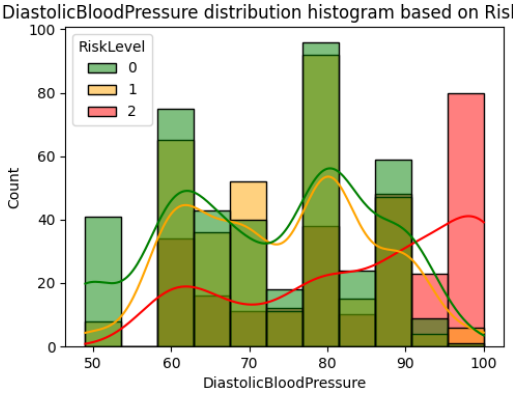

Age distribution histogram count
Count
Age

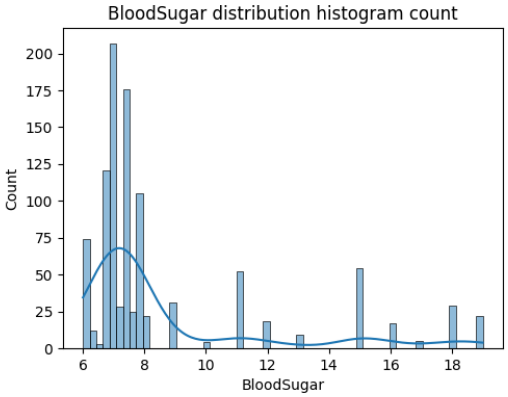

Age distribution histogram based on Risk level
RiskLevel
0
1
2
Count
Age

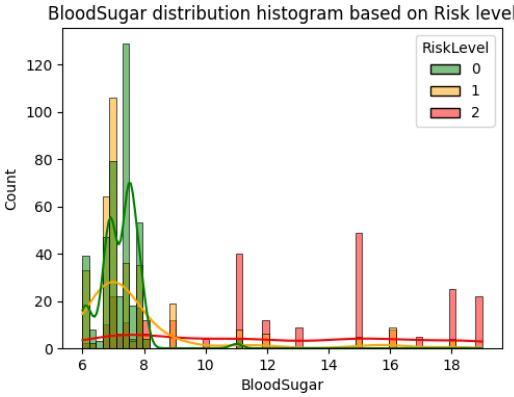

SystolicBloodPressure distribution histogram count
Count
SystolicBloodPressure

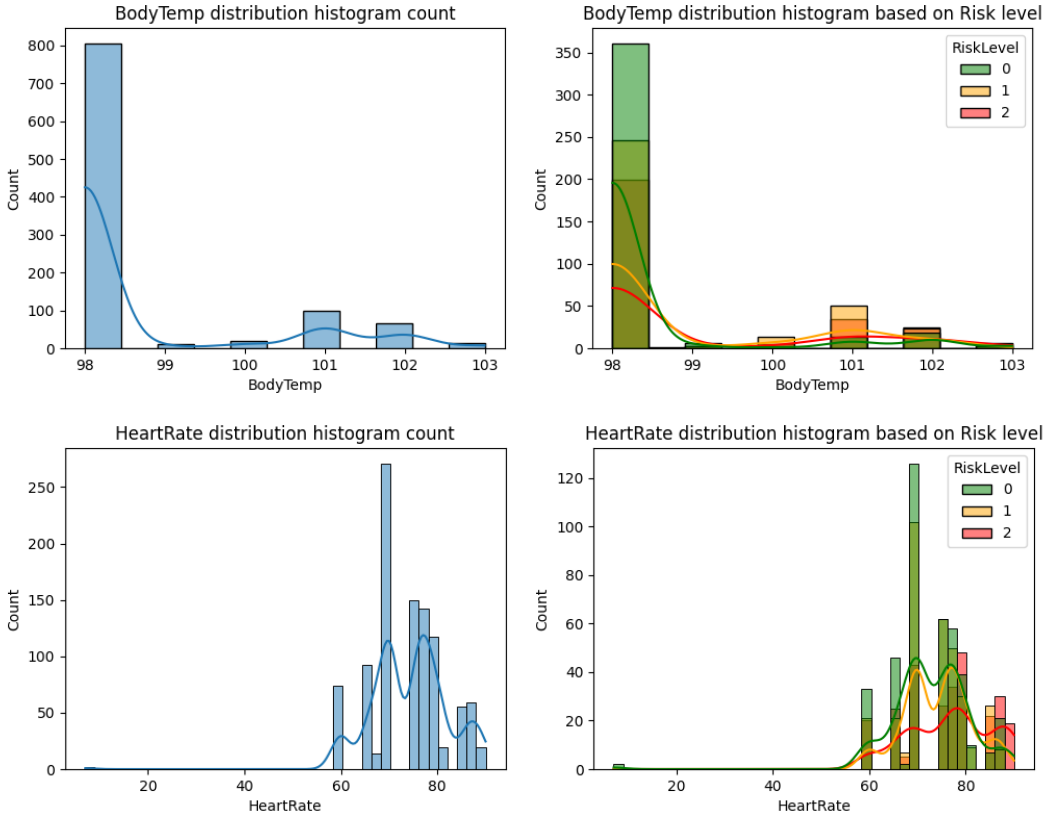

SystolicBloodPressure distribution histogram based on Risk level
RiskLevel
0
1
2
Count
SystolicBloodPressure


DiastolicBloodPressure distribution histogram count
Count
DiastolicBloodPressure


DiastolicBloodPressure distribution histogram based on Risk level
RiskLevel
0
1
2
Count
DiastolicBloodPressure

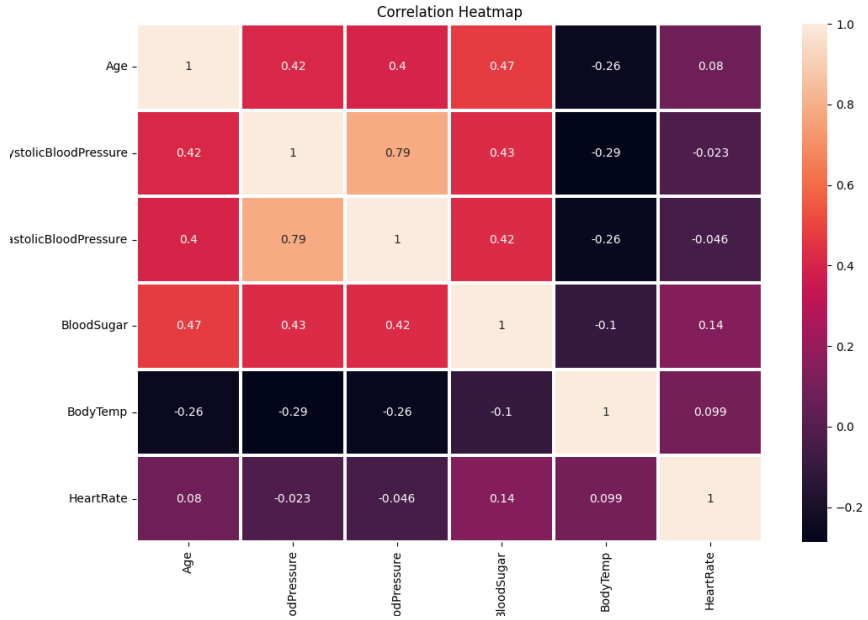

BloodSugar distribution histogram count
Count
BloodSugar

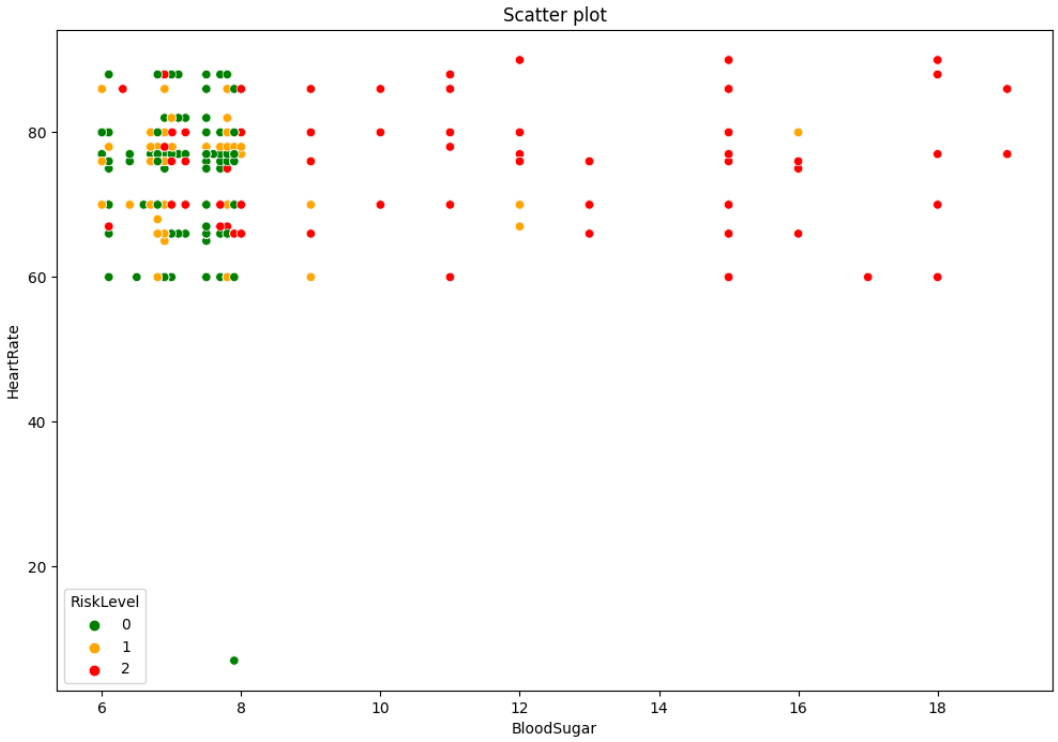

BloodSugar distribution histogram based on Risk level
RiskLevel
0
1
2
Count
BloodSugar



Abbiamo esaminato i grafici di tutte le feature in relazione al target. Dall'analisi di questi dati, abbiamo rilevato un singolo valore anomalo, si tratta di due registrazioni con 7 battiti al minuto, un valore impossibile per un essere umano. Per questa ragione, abbiamo deciso di rimuovere tali due registrazioni dal dataset. Era con ogni probabilità quello trovato prima.

Analisi correlazione dei dati





Dall’analisi della correlazione tra i dati, abbiamo rilevato che la frequenza cardiaca non fornisce alcuna informazione aggiuntiva nel calcolo del RiskLevel. Qui sopra un esempio tra frequenza

cardiaca e glicemia, altri esempi sono disponibili eseguendo l’applicazione visualizzando i grafici dei dati.

Come possiamo osservare, indipendentemente dal numero dei battiti cardiaci, l'unico fattore che risulta essere significativamente associato a un aumento dei rischi è un elevato livello di glicemia nel sangue. Questo si verifica per ogni altro valore, mantenendo costante HeartRate. Pertanto, abbiamo preso la decisione di eliminare completamente l’intera colonna.

ALGORITMI DI APPRENDIMENTO

Prima di iniziare diamo un’occhiata a due algoritmi utilizzati:

StratifiedKFold

La Stratified K-fold Cross-validation ha un funzionamento simile alla classica K-fold Cross-validation, ma presenta una distinzione fondamentale: garantisce che ogni fold abbia una distribuzione delle classi simile a quella dell'intero dataset. La cross-validation permette per diverse porzioni del set di dati di train e test di calcolare lo score per ciascuna iterazione. Questo approccio ci permette di evitare l'overfitting del modello e ci fornisce una visione più completa del suo comportamento. Il risultato finale è ottenuto calcolando la media dei punteggi ottenuti in tutte le diverse iterazioni.

RandomizedSearchCV

La Randomized Search Cross-validation è utile quando hai un numero limitato di risorse computazionali e vuoi trovare rapidamente una buona combinazione di iperparametri senza esplorare ogni possibile combinazione. È stata utilizzata in tutti gli

algoritmi in cui era applicabile al fine di ottenere le migliori prestazioni.

Inoltre, abbiamo creato due nuove differenti versioni del dataset perché alcuni algoritmi performano meglio avendo i dati standardizzati o normalizzati.

Per la standardizzazione è stato usato *StandardScaler()* con la seguente formula:

$$xStandard = (x - m) / u$$

dove xStandard rappresenta il valore di x dopo essere stato standardizzato, m ed u corrispondono rispettivamente alla media e alla deviazione standard dei valori contenuti nella colonna di x nel dataset di addestramento. I dati vengono standardizzati in modo che la media delle caratteristiche sia zero e la deviazione standard sia uno.

Per la normalizzazione è stato usato *MinMaxScaler()* con la seguente formula:

$$xNorm = (x - min) / (max - min)$$

dove xNorm rappresenta il valore di x dopo essere stato normalizzato, mentre min e max corrispondo rispettivamente al minimo e massimo valore della colonna x nel dataset di addestramento. I dati vengono scalati in modo che i valori della caratteristica x si trovino all'interno di un intervallo specifico, solitamente tra 0 e 1.

Ora passiamo ai classificatori (Nel repository sono presenti anche le immagini della matrice di confusione per ogni classificatore e i grafici che rappresentano l'accuracy per ogni iterazione della stratified k-fold):

Support Vector Classification(SVC)

Nella foto seguente troviamo i migliori parametri scelti da "RandomizedSearchCV", i risultati del modello e la media dei risultati ottenuti dalla "StratifiedKFold"

```
Best hyperparameter: {'gamma': 1, 'C': 10}
Result:

              precision    recall  f1-score   support

           0       0.72      0.72      0.72        79
           1       0.69      0.67      0.68        60
           2       0.88      0.91      0.89        64

    accuracy                           0.76       203
   macro avg       0.76      0.76      0.76       203
weighted avg       0.76      0.76      0.76       203

Accuracy: 76.35467980295566 %
Accuracy with K-Fold: 77.63117283950616 %
```

L'algoritmo delle macchine a vettori di supporto (SVM) rappresenta ogni dato come un punto in uno spazio a n dimensioni, dove 'n' è il numero di caratteristiche disponibili, e assegna il valore di ciascuna caratteristica come coordinata di quel punto. Successivamente, l'SVM esegue la classificazione trovando l'iperpiano ottimale, che separa i dati positivi da quelli negativi in questo spazio multidimensionale.

È stato scelto di usare il dataset normalizzato per questo algoritmo perchè essendo un algoritmo distance-based è più opportuno.

Random Forest Classifier

Nell'immagine seguente troviamo i migliori parametri scelti da "RandomizedSearchCV", i risultati del modello e la media dei risultati ottenuti dalla "StratifiedKFold"

```
Best hyperparameter: {'n_estimators': 250}
Result:

              precision    recall  f1-score   support

           0       0.91      0.85      0.88        79
           1       0.84      0.87      0.85        60
           2       0.93      0.97      0.95        64

    accuracy                           0.89       203
   macro avg       0.89      0.89      0.89       203
weighted avg       0.89      0.89      0.89       203

Accuracy: 89.16256157635468 %
Accuracy with K-Fold: 82.08024691358024 %
```

L'algoritmo utilizzato in questo metodo fa parte delle tecniche di *ensemble learning* che sfruttano un insieme di modelli meno precisi, come gli alberi decisionali, con l'obiettivo di crearne uno più accurato. Vengono quindi costruiti un elevato numero di alberi decisionali individuali, ognuno su un diverso sottoinsieme del dataset originale. Ciascun albero nel Random Forest effettua una previsione di classe e la classe che è stata predetta più frequentemente diventa il risultato del modello.

Decision Tree Classifier

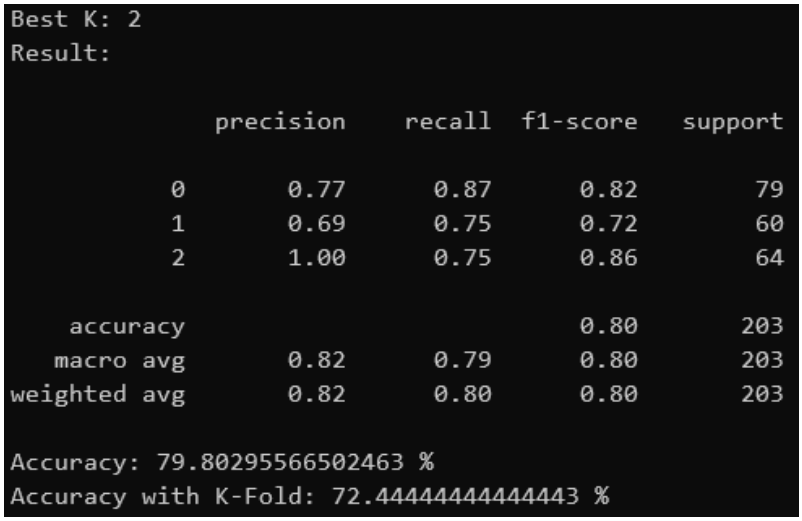
Nella foto seguente troviamo i migliori parametri scelti da “RandomizedSearchCV”, i risultati del modello e la media dei risultati ottenuti dalla “StratifiedKFold”

```
Best hyperparameter: {'criterion': 'log_loss'}
Result:

              precision    recall  f1-score   support

           0       0.88      0.85      0.86        79
           1       0.78      0.83      0.81        60
           2       0.95      0.94      0.94        64

    accuracy                           0.87       203
   macro avg       0.87      0.87      0.87       203
weighted avg       0.87      0.87      0.87       203

Accuracy: 87.192118226601 %
Accuracy with K-Fold: 81.83487654320987 %
```

Il Decision Tree Classifier è un algoritmo di apprendimento automatico che costruisce una struttura ad albero in cui ogni nodo rappresenta una decisione basata su una caratteristica dei dati. Questi nodi conducono a foglie che rappresentano classi o categorie. Durante la classificazione, il modello attraversa l'albero in base alle caratteristiche dei dati e assegna una classe al punto dati basata sulla sequenza di decisioni prese lungo il percorso dell'albero. L'obiettivo è trovare regole di divisione ottimali per separare le classi nel dataset di addestramento.

K-Nearest neighbors Classifier

Nella foto seguente troviamo i migliori parametri scelti da “RandomizedSearchCV”, i risultati del modello e la media dei risultati ottenuti dalla “StratifiedKFold”



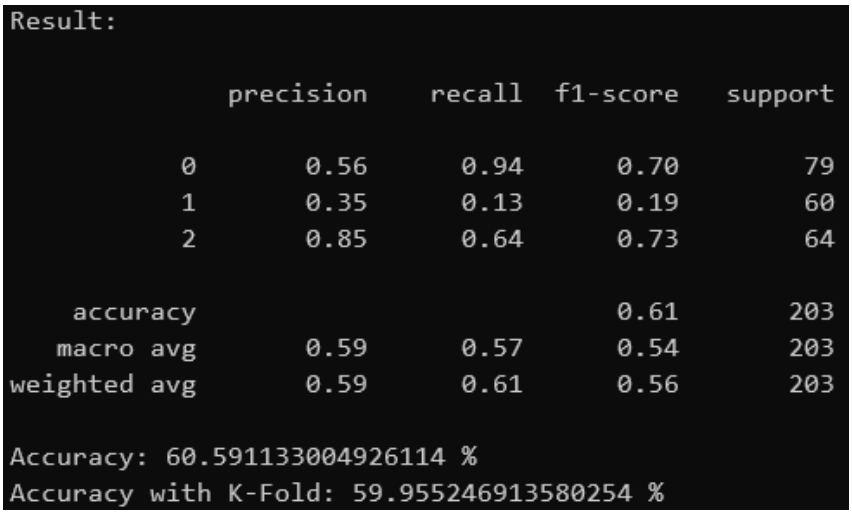
```
Best K: 2
Result:

              precision    recall  f1-score   support

           0       0.77      0.87      0.82        79
           1       0.69      0.75      0.72        60
           2       1.00      0.75      0.86        64

    accuracy                           0.80       203
   macro avg       0.82      0.79      0.80       203
weighted avg       0.82      0.80      0.80       203

Accuracy: 79.80295566502463 %
Accuracy with K-Fold: 72.44444444444443 %
```

Il classificatore K-Nearest Neighbors (K-NN) è un algoritmo di apprendimento automatico che assegna una classe a un punto dati in base alle classi dei suoi "K" punti più vicini nel dataset di addestramento, utilizzando una misura di distanza. È un metodo "instance-based" che non costruisce un modello durante l'addestramento, ma effettua una ricerca ogni volta che deve effettuare una previsione.

Abbiamo testato l’algoritmo con k da 2 a 50, alla fine il migliore è risultato quello con k = 2, inoltre per massimizzarne i risultati abbiamo usato il database normalizzato essendo il K-nn distance-based

Gaussian Naive Bayes

Nella foto seguente troviamo i migliori parametri scelti da “RandomizedSearchCV”, i risultati del modello e la media dei risultati ottenuti dalla “StratifiedKFold”



Result:

	precision	recall	f1-score	support
0	0.56	0.94	0.70	79
1	0.35	0.13	0.19	60
2	0.85	0.64	0.73	64
accuracy			0.61	203
macro avg	0.59	0.57	0.54	203
weighted avg	0.59	0.61	0.56	203

Accuracy: 60.591133004926114 %
Accuracy with K-Fold: 59.955246913580254 %

Il Gaussian Naive Bayes è un algoritmo di classificazione basato sul teorema di Bayes e sull'assunzione di indipendenza condizionale tra le caratteristiche (da cui "naive"). Questo algoritmo assume che le caratteristiche seguano una distribuzione gaussiana (normale) e calcola le probabilità di appartenenza a ciascuna classe per un dato punto dati utilizzando la probabilità a priori delle classi e le probabilità condizionate delle caratteristiche. In altre parole, il Gaussian Naive Bayes utilizza il teorema di Bayes per calcolare la probabilità che un punto dati appartenga a una determinata classe data la sua distribuzione di caratteristiche. Questo rende il Gaussian Naive Bayes un algoritmo di classificazione probabilistico che può gestire dati

numerici continui. È "naive" perché assume l'indipendenza completa tra le caratteristiche, il che potrebbe non essere vero in molti casi reali.

SISTEMA ESPERTO

Per migliorare l'esperienza utente ed il riconoscimento delle complicazioni più frequenti nello specifico, è stato implementato un sistema esperto basato su regole grazie alla libreria Python "experta" (https://experta.readthedocs.io/en/latest/index.html). La base di conoscenza è stata formata sulle informazioni estratte da un'informativa del NICHD (National Institute of Child Health and Human Development) raggiungibile al seguente link: https://www.nichd.nih.gov/health/topics/pregnancy/conditioninfo/complications.

Il sistema è in grado di riconoscere 6 complicazioni sulla base di 14 sintomi o fattori di rischio. Vengono elencati all'utente tutti i fattori a cui può rispondere con "si" o "no" in base alla propria esperienza. Gli verrà poi indicata la complicanza con una breve descrizione ed il trattamento consigliato, a cui corre maggior rischio, anche nel caso non vengano riscontrati tutti i fattori, specificandolo.

Le regole utilizzate al riconoscimento delle complicazioni sono rappresentabili con questa tabella:

FATTORI DI RISCHIO	**COMPLICAZIONI**					
	Mortinatalità	**Travaglio pretermine**	**Preeclampsia**	**Diabete gestazionale**	**Iperemesi gravidica**	**Anemia da carenza di ferro**
Obesità		X	X	X		
Contrazioni addominali	X	X	X			
Pressione alta		X	X			
Bruciore di stomaco		X	X			
Vista sfocata		X	X			
Nausea		X	X		X	
Infezioni		X				
Perdita di sangue	X	X				
Perdita di liquidi	X	X				
Perdita di peso o appetito					X	
Vomito					X	
Sensazione di svenimento					X	X
Respiro corto						X
Stanchezza						X

Per esempio, se l'utente indica che riscontra solo il fattore di rischio "obesità" il sistema gli indicherà il diabete gestazionale come complicazione più probabile.

Il sistema indicherà il fatto (complicazione) che rispetta più regole (fattori di rischio) e nel caso di parità verrà indicata la complicazione che richiede più urgenza ad agire, in ordine decrescente in tabella. Dal momento in cui sono presenti poche regole questa applicazione rimane solo una dimostrazione del potenziale di un sistema esperto che, anche in questa applicazione, rimane facilmente scalabile grazie alle sue caratteristiche ma richiede che la base di conoscenza sia curata e completata dalle informazioni di esperti nel settore di assistenza sanitaria alla maternità.