DIABETECARE

Studente: Samuele Palumbo
Matricola: 676988
Caso di studio Ingegneria della Conoscenza
GitHub:

1) Introduzione

Il diabete è una malattia cronica caratterizzata da un aumento dei livelli di zucchero (glucosio) nel sangue, chiamato iperglicemia. Questo squilibrio si verifica perché il corpo non produce abbastanza insulina o non è in grado di utilizzare l'insulina in modo efficace. L'insulina è un ormone prodotto dal pancreas che regola il livello di glucosio nel sangue, permettendo alle cellule di assorbirlo per produrre energia. Esistono diversi tipi di diabete

- Diabete di tipo 1: È una malattia autoimmune in cui il sistema immunitario attacca e distrugge le cellule del pancreas che producono insulina. Di solito si manifesta in età giovanile e richiede somministrazioni quotidiane di insulina per controllare i livelli di glucosio nel sangue.
- Diabete di tipo 2: È il tipo più comune e si sviluppa quando il corpo diventa resistente all'insulina o non produce abbastanza insulina per mantenere i livelli di glucosio normali. Spesso associato a sovrappeso, sedentarietà e cattive abitudini alimentari, può essere gestito con dieta, esercizio fisico e farmaci.
- Diabete Gestazionale: si verifica durante la gravidanza, quando gli ormoni prodotti dalla placenta interferiscono con l'azione dell'insulina. Solitamente si risolve dopo il parto, ma aumenta il rischio di sviluppare un diabete di tipo 2 in futuro

I sintomi del diabete includono sete eccessiva, minzione frequente, stanchezza, perdita di peso, visione offuscata e lenta guarigione dalle ferite, se non viene trattato può portare a problemi gravi come problemi cardiovascolari, danni ai reni, danno ai nervi e alla retina oltre a varie infezioni.
Il trattamento dipende dal tipo di diabete ma può includere

- Insulina, soprattutto nel caso del diabete di tipo 1
- Farmaci ipoglicemizzanti, soprattutto nel caso di diabete di tipo 2
- Dieta bilanciata e monitoraggio del glucosio
- Attività fisica regolare

2) Requisiti fondamentali

La realizzazione di questo progetto è stata effettuata utilizzando il linguaggio python, poiché particolarmente performante nella manipolazione e l'analisi dei dati.
L'ambiente di sviluppo utilizzato è stato Visual Studio Code, le librerie utilizzate sono state:

- Pandas: libreria utile per la manipolazione e l'analisi dei dati
- Scikit-learn: per la fase di classificazione e le metriche di valutazione
- Numpy: per l'esecuzione delle adeguate operazioni su certi tipi di dato,chat
- Matplotlib, Seaborn: utilizzate per la gestione e la creazione dei grafici
- Os: per selezionare il colore in output delle scritte
- Warnings: per la gestione degli errori
- Pickle: per salvare e caricare i modelli addestrati

3) Dataset

Il dataset utilizzato è stato scaricato dal sito di www.kaggle.com in formato csv.
L'obiettivo è quello di effettuare delle predizioni, attraverso delle misurazioni diagnostiche, sulla possibilità che un paziente sia diabetico o meno.
In questo dataset tutti i soggetti analizzati sono donne di almeno 21 anni.
Nel dataset sono presenti 8 feature continue e 1 feature dicotomica

- Pregnancies: indica il numero di gravidanze avute
- Glucose: indica la concentrazione di glucosio nel plasma a 2 ore da test di tolleranza orale
- BloodPressure: indica la pressione diastolica (mmHg)
- SkinThickness: indica lo spessore della piega cutanea del tricipite (mm)
- Insulin: indica il valore dell'insulina dopo 2 ore (mu U/ml)
- BMI: indica il valore dell'indice di massa corporea
- DiabetesPedigreeFunction: indica la probabilità di ereditare il diabete
- Age: indica l'età
- Outcome: indica se la persona ha il diabete o meno, 1 ha il diabete 0 non ha il diabete

Glucose→ valori di glicemia dopo 2 ore da un carico orale di glucosio, inferiori a 140 mg/dl sono ritenuti normali, tra i 140 e i 199 fanno porre diagnosi di ridotta tolleranza ai carboidrati e infine i valori >200 fanno porre diagnosi di diabete

BloodPressure→ la pressione minima è il valore della pressione arteriosa nel momento in cui il cuore è in fase di rilassamento, la pressione troppo alta è molto pericolosa, però nel caso dei pazienti diabetici anche quella troppo bassa lo è.

L'American Heart Association definisce 'normale' la pressione sistolica inferiore a 120 e quella *diastolica inferiore a 89*.
SkinThickness→ la persona diabetica presenta disturbi e malattie cutanee connesse all'alterato metabolismo del glucosio. Le persone affette da diabete tendono ad avere una pelle più spesa rispetto ai non diabetici

Insulin→ L' insulina è un ormone che permette a tutte le cellule dell'organismo di prelevare il glucosio del sangue per trasformarlo in energia da utilizzare per le proprie necessità, svolgendo di fatto un ruolo insostituibile nel metabolismo. I valori normali di insulina a seguito di somministrazione di 75g di glucosio sono:

- Digiuno 5-25 micr.UI/ml
- Dopo 30 min 41-125 micr.UI/ml
- Dopo 60 min 20-120 micr.UI/ml
- Dopo 90 20-90 micr.UI/ml
- Dopo i 120 min 18-56 micr.UI/ml

BMI→ il calcolo del BMI è un sistema di valutazione del peso, il suo valore ci permette di capire se una persona è normopeso, sottopeso, sovrappeso o obesa.
I valori più indicati di BMI, se riferito all'aspetto metabolico-salutistico, si aggirano intorno a 21-22 (22,5 kg/m 2 nell'uomo e a *21 kg/m 2 nella donna*).

DiabetesPedigreeFunction→ è la funzione che valuta la probabilità di diabete in base alla storia familiare, i valori vanno da 0 a 100

Il dataset successivamente è stato analizzato e modificato per poter essere utilizzato al meglio. Prima di tutto si verifica la presenza di valori nulli, se essi sono presenti bisogna sostituirli, nel dataset sono presenti dei valori nulli.
Dopo aver sostituito gli zeri presenti il risultato del comando print(df.isnull().sum()) è questo:

```
Values with zeroes.
Glucose              5
BloodPressure       35
SkinThickness      227
Insulin            374
BMI                 11
dtype: int64
```

Successivamente ho sostituito i valori nulli con la media delle rispettive colonne, in questo modo non ho valori nulli

```
Filling null values...
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Qui abbiamo il numero di valori nulli dopo la modifica.

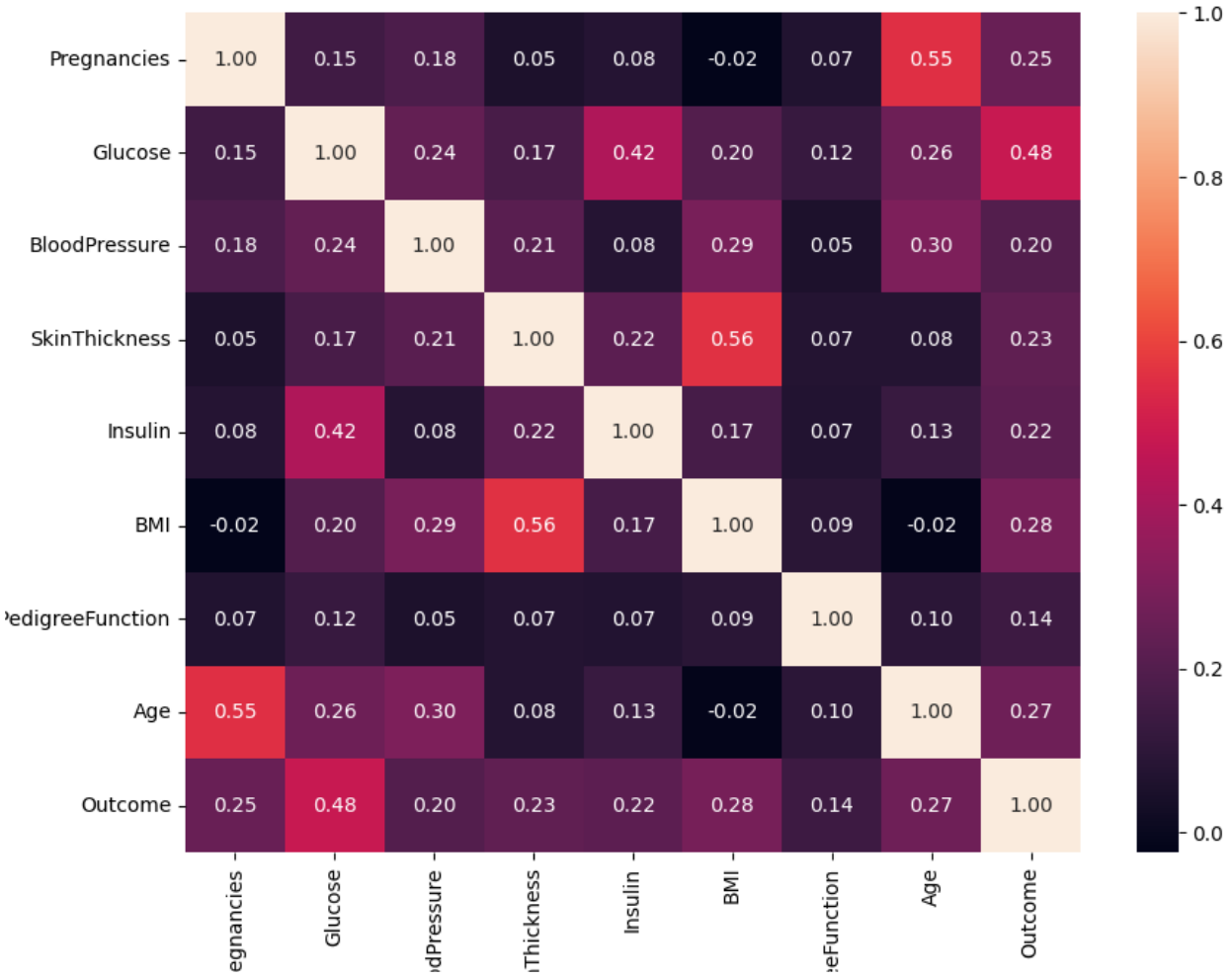
Dopo bisogna effettuare un bilanciamento delle classi, per avere un'ottima efficienza in fase di apprendimento è necessario avere lo stesso numero di esempi per entrambe le classi, quindi sia diabetici che non. Se dovessero essere sbilanciate si potrebbero avere dei problemi nella valutazione della precisione dei modelli addestrati, infatti andrebbero a osservare un numero di valori diverso per ciascuna classe da valutare, rendendo difficoltoso e non corretto l'apprendimento
Prima del bilanciamento questi sono i valori:

```
Controllo del bilanciamento delle classi
Non diabetici:  500 (% 65.10)
Diabetici:  268 (% 34.90)
```

Quindi le classi sono sbilanciate.
Le bilanciamo attraverso l'oversampling, esso è un processo che consiste nel generare nuovi dati simili alla classe meno rappresentata, in modo da colmare il divario tra le due classi. Dopo aver effettuato l'oversampling le classi sono bilanciate

```
Valori dopo oversampling:
Non diabetici:  500 (% 50.00)
Diabetici:  500 (% 50.00)
```

Una volta che le classi sono bilanciate ho cercato dei fattori che potessero caratterizzare un paziente diabetico. Le caratteristiche che possono far pensare ad una possibile causa di diabete sono principalmente i livelli del glucosio e dell'insulina. Per evidenziare il rapporto tra le varie feature generiamo una heatmap



Dalla heatmap possiamo notare che ci sono dei sintomi correlati tra di loro. L'esito di outcome dipende soprattutto dal valore di glucosio, anche se in modo non forte, il quale a sua volta è correlato al valore dell'insulina

4) Apprendimento supervisionato

Prima di tutto ho effettuato una standardizzazione dei dati. La **standardizzazione** è un metodo di riduzione di scala che trasforma una serie di valori in una distribuzione normale standard, con media uguale a zero e devianza standard uguale a uno. La standardizzazione consiste nel sottrarre la media (μ) da ogni valore (x) del vettore e dividere la differenza per la devianza standard (σ).

$$x = \frac{x - \mu}{\sigma}$$

x=vettore
μ=media
σ=devianza standard

È molto importante perché le performance del processo di learning sono generalmente superiori quando l'algoritmo lavora su dati standardizzati.
Successivamente effettuiamo una ottimizzazione degli iperparametri, poiché fondamentale per il corretto funzionamento dei modelli di machine Learning.
Il metodo Grid Search considera diverse combinazioni di iperparametri e sceglie quella che restituisce un punteggio di errore inferiore, inoltre restituisce risultati facilmente riproducibili; quindi, ogni volta che eseguo il Grid Search, otterrò lo stesso risultato.
In GridSearchCV, insieme alla Grid Search, viene eseguita anche la cross-validation

```
#Pipeline DCT
pipe_dct = Pipeline([('scaler', StandardScaler()), ('dtc', DecisionTreeClassifier())])
param_grid = {'dtc__criterion': ['gini', 'entropy'],
              'dtc__max_depth': range(1, 100)}
opt_dct = GridSearchCV(estimator=pipe_dct, param_grid=param_grid, scoring='accuracy')
```

La cross-validation viene utilizzata durante l'addestramento del modello, il tipo più popolare è la k-fold cross-validation.
È un processo iterativo che divide i dati di train in k partizioni. Ogni iterazione conserva una partizione per il test e le restanti k-1 partizioni per il training del modello.
L'iterazione successiva imposterà la partizione successiva come dati di test e il restante k-1 come dati di train e così via. In ogni iterazione, registrerà le prestazioni del modello e alla fine darà la media di tutte. In particolare, ho deciso di effettuare il **Repeated K-fold**, per ottenere una media delle metriche ancora più accurata.
Pertanto, è un processo che richiede tempo.

Mentre come classificatori ho scelto:

- Random Forest
- Decision Tree
- K-Nearest Neighbors
- Logistic Regression
- Multilayer Perceptron
- Gaussian Naive Bayes

Come metriche per la valutazione di questi modelli ho scelto:

- Accuracy
- Precision
- Recall
- F1- score

RANDOM FOREST

L'algoritmo del random forest è basato sull'addestramento di un numero N di decision Tree, ognuno dei quali effettua una classificazione per ogni esempio.
Quando tutti gli alberi (o più precisamente tutta la foresta) hanno classificato l'esempio, si effettua una conta su qual è stata la classe maggiormente stimata e la si assume come predizione della foresta.
Risultati Random Forest:

	Non ottimizzato	Ottimizzato
Accuracy:	0.840	0.860
Precision:	0.850	0.820
Recall:	0.939	0.942
F1Score:	0.892	0.875
Dev. Standard	0.06	

DECISION TREE

Il funzionamento degli alberi di decisione prevede che il valore di una feature obiettivo venga classificato sulla base di una serie di regole di decisione basate sui dati di input a disposizione. Nello specifico, ogni nodo interno dell'albero indica una condizione e i valori derivati dagli esempi di input costituiscono dei sottoalberi. Le foglie dell'albero invece contengono il valore della feature obiettivo.

	Non ottimizzato	Ottimizzato
Accuracy:	0.835	0.805
Precision:	0.847	0.760
Recall:	0.869	0.903
F1Score:	0.810	0.830
Dev. Standard	0.05	

K-NEAREST NEIGHBORS

Il funzionamento dell'algoritmo relativo al classificatore K-NN si basa sulla semplice memorizzazione degli esempi del dataset (comprendendo la/le feature obiettivo), senza che venga appreso un modello.
La classificazione avviene confrontando il nuovo esempio con un insieme formato da k vicini, che "voteranno" per decretare la classe di appartenenza del nuovo esempio.
La votazione può avvenire come calcolo della moda, della media o a seguito dell'interpolazione dei k vicini.

	Non ottimizzato	Ottimizzato
Accuracy:	0.820	0.795
Precision:	0.784	0.740
Recall:	0.947	0.932
F1Score:	0.858	0.825
Dev. Standard	0.04	

LOGISTIC REGRESSION

Il modello di classificazione della regressione logistica si basa su dei pesi di una funzione lineare appiattita dalle sigmoidee, minimizzando un errore su E.
È importante specificare che parte come un modello di regressione, ma successivamente viene appiattito con la log loss.

	Non ottimizzato	Ottimizzato
Accuracy:	0.700	0.720
Precision:	0.808	0.744
Recall:	0.626	0.701
F1Score:	0.705	0.722
Dev. Standard	0.06	

MULTILAYER PERCEPTRON

Una Rete Neurale è un modello di apprendimento che si basa sul funzionamento dei neuroni cerebrali biologici. Ho utilizzato una Rete Neurale di tipo feed-forward che si basa su una gerarchia di funzioni lineari intervallate da funzioni di attivazione. In genere prende in input una serie di feature e le sottopone agli strati nascosti (detti anche feature non osservate) che le mappano secondo una funzione di attivazione e restituiscono in output uno o più feature obiettivo. Formalmente è composta da tre strati o layer:
Un layer di input (in questo caso caso tutte le feature).
Un layer lineare completo, il cui modello matematico è

$$y = \sum_i wi * xi$$

dove y è la classe da predire, x è il numero di feature di input e w sono i pesi da assegnare ad ogni feature di input.
Una funzione di attivazione f.
Per ottenere una classificazione accurata si fa uso della back-propagation che ricalcola ed aggiorna i pesi w.

	Non ottimizzato	Ottimizzato
Accuracy:	0.750	0.760
Precision:	0.803	0.741
Recall:	0.747	0.826
F1Score:	0.774	0.781
Dev. Standard	0.02	

GAUSSIAN NAIVE BAYES

È un algoritmo di classificazione basato sul teorema di bayes

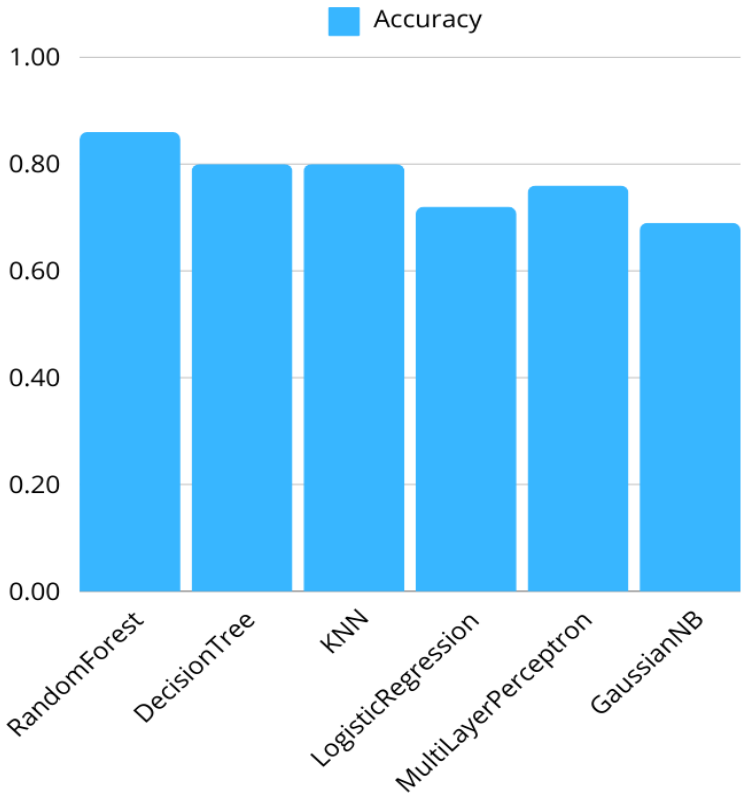
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

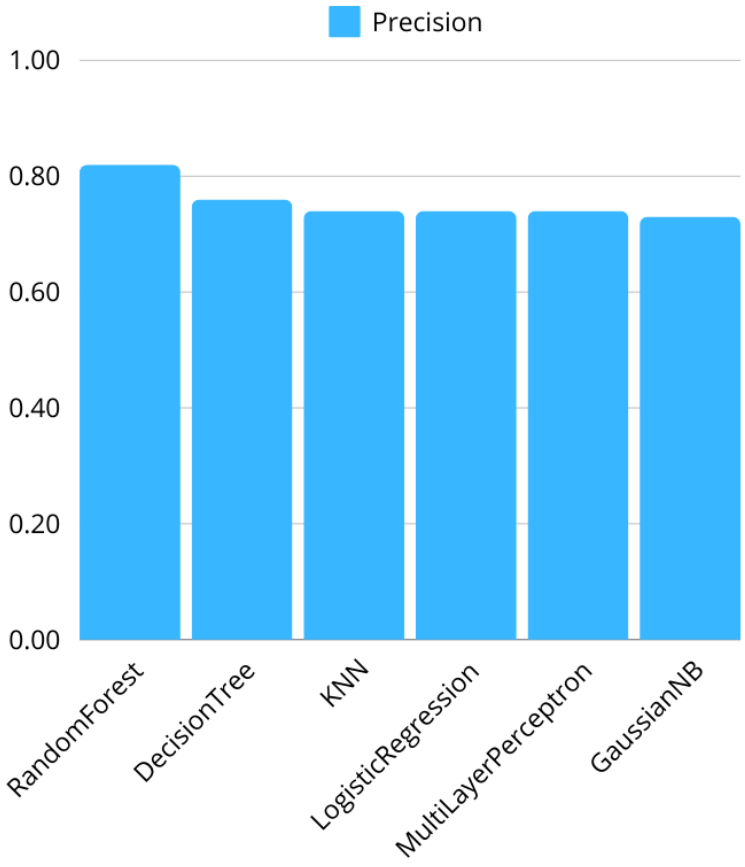
- **P(A|B)** è la probabilità condizionata dell'evento A dato l'evento B (ovvero la probabilità a posteriori della classe dati i predittori).
- **P(B|A)** è la probabilità condizionata dell'evento B dato A
- **P(A)** è la probabilità a priori dell'evento A
- **P(B)** è la probabilità a priori dell'evento B

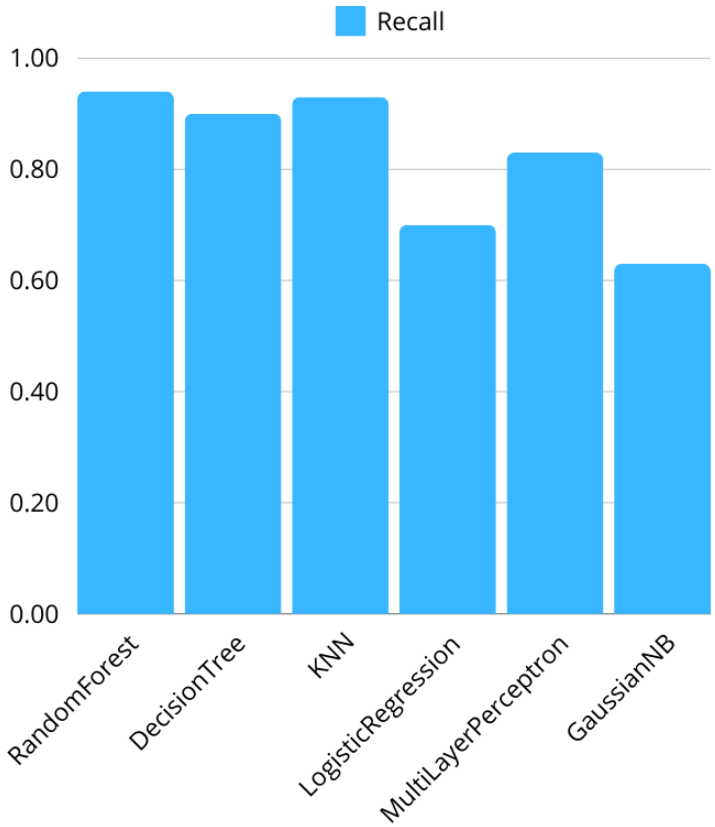
Mentre il gaussian naive bayes è una variante che segue la distribuzione normale gaussiana e supporta dati continui

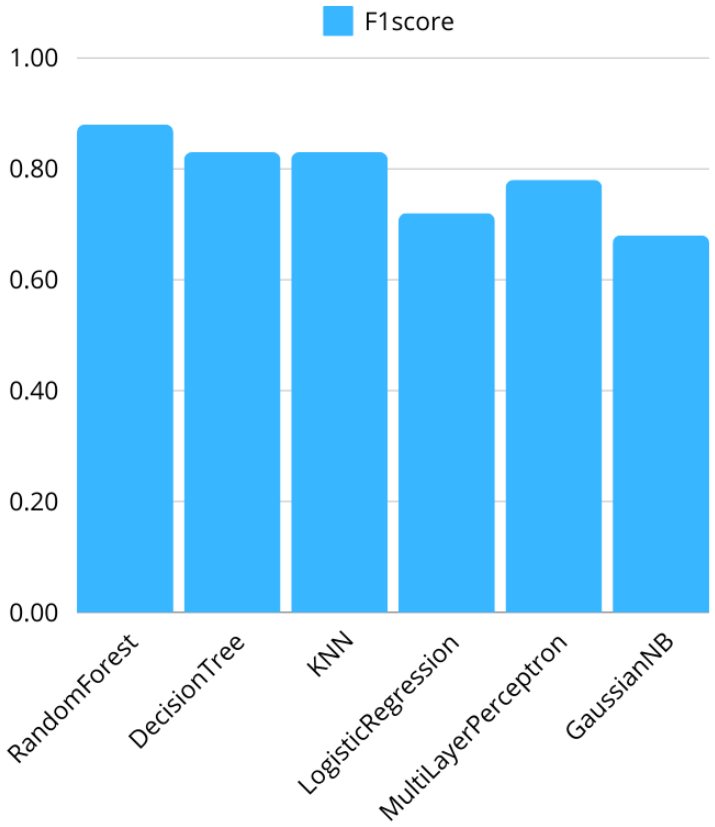
	Non ottimizzato	Ottimizzato
Accuracy:	0.690	0.685
Precision:	0.791	0.725
Recall:	0.626	0.634
F1Score:	0.699	0.676
Dev. Standard	0.07	

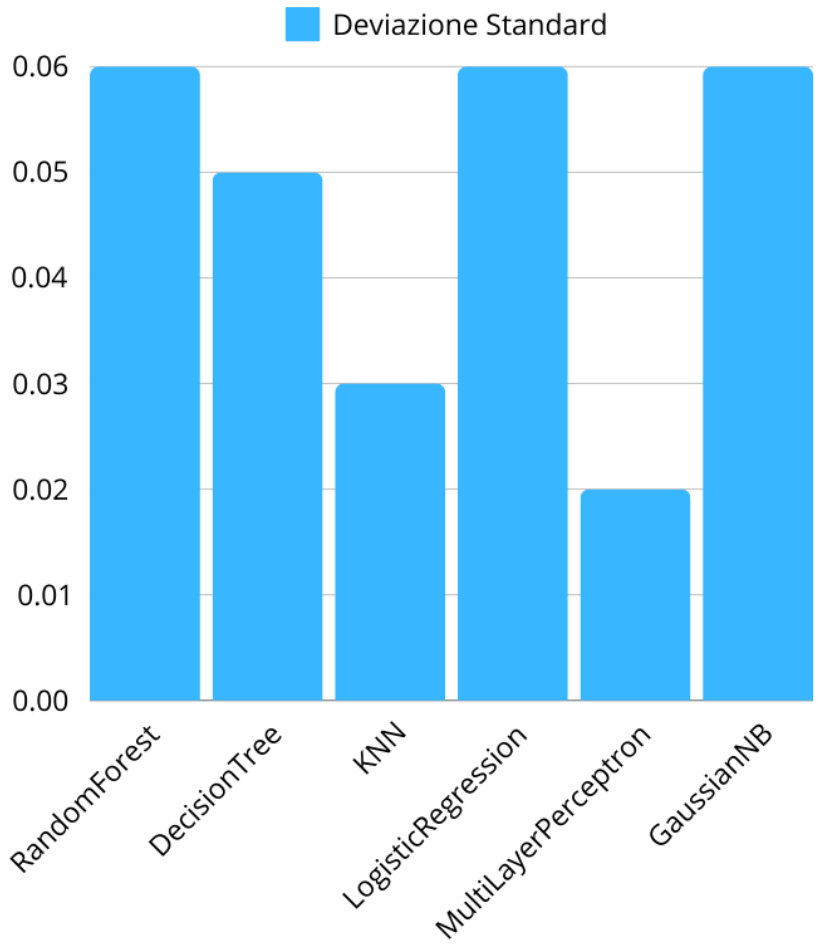
In seguito, abbiamo i seguenti grafici per ogni metrica:






Recall
1.00
0.80
0.60
0.40
0.20
0.00
RandomForest
DecisionTree
KNN
LogisticRegression
MultiLayerPerceptron
GaussianNB


F1score
1.00
0.80
0.60
0.40
0.20
0.00
RandomForest
DecisionTree
KNN
LogisticRegression
MultiLayerPerceptron
GaussianNB



Vedendo i seguenti grafici possiamo vedere che il Random Forest è il migliore

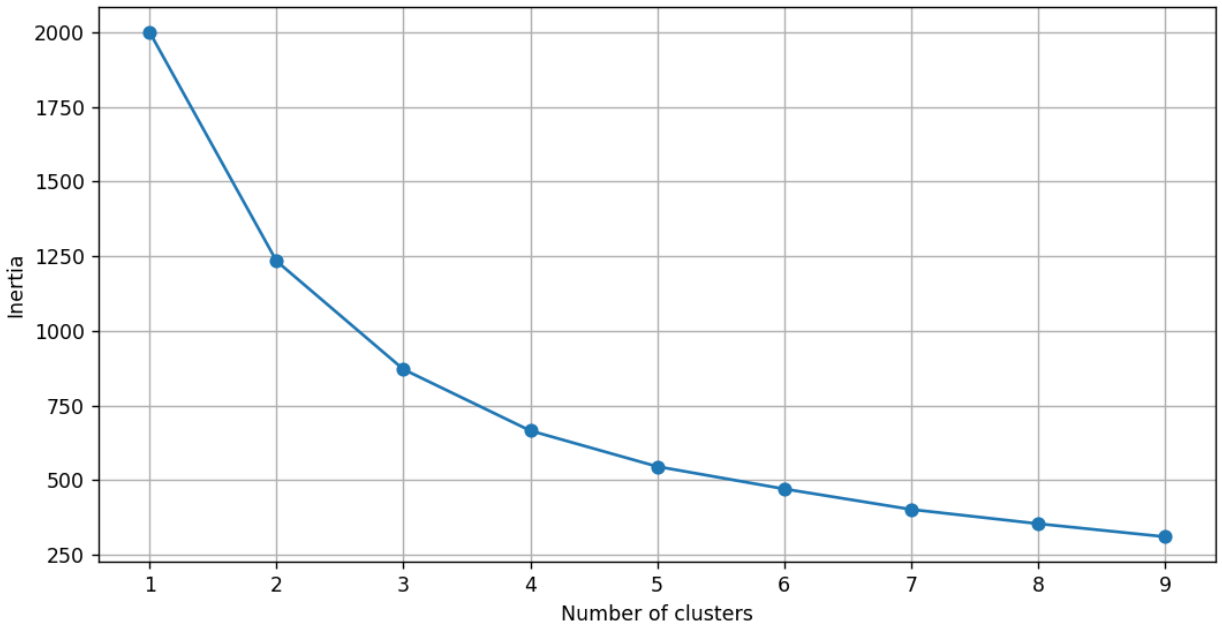
5) APPRENDIMENTO NON SUPERVISIONATO

L'apprendimento non supervisionato si occupa di analizzare e modellare i dati senza utilizzare etichette predefinite per le osservazioni. A differenza dell'apprendimento supervisionato, dove i dati di addestramento sono accompagnati da etichette che indicano la risposta corretta, l'apprendimento non supervisionato lavora su dati non etichettati e cerca di scoprire strutture o pattern nascosti

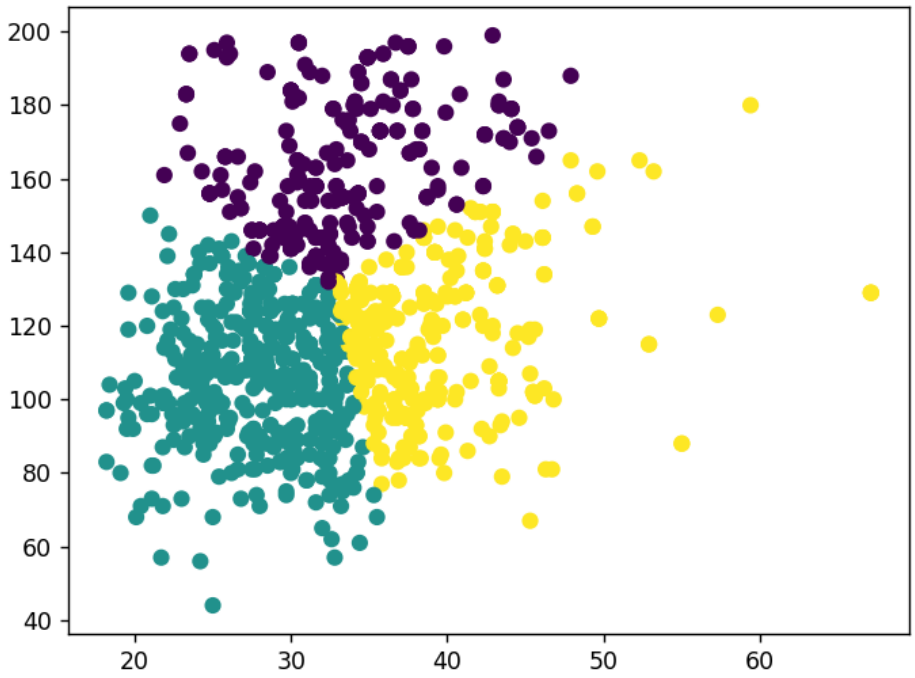
K-MEANS CLUSTERING

Questo algoritmo separa i dati in cluster distinti, che non sono stati etichettati nei dati. I punti dati possono appartenere ad un solo cluster. Un k più grande significa un gruppo più piccolo con più granularità.

In questo caso ho eliminato alcune faeture dal dataset: Pregnancies, BloodPressure, SkinThickness e Outcome.
Ho poi standardizzato i dati, così da avere solo valori nel range di -1 e 1.
Una volta standardizzati i dati è importate scegliere il numero di clusters che si vuole indentificare (k). Successivamente si selezionano casualmente k valori distinti che rappresenteranno i cluster iniziali. Si misura, poi, la distanza tra il primo punto nel dataset e i tre cluster iniziali; assegno quel punto a un cluster, in base a quello con distanza minore, e faccio questa operazione per ogni punto.
In uno spazio a due dimensioni, per ogni cluster si trova il centroide (facendo la media della distanza dei punti assegnati al singolo cluster j).
Una situazione che potrebbe gravare molto sulle performance del nostro classificatore è la giusta scelta del valore k. Ho scelto di sfruttare l'**elbow method**, in modo da ottenere il numero ottimare di cluster da eseguire, ottenendo questo risultato:



Come possiamo vedere il numero di cluster ottimale è 3, quindi il modello è stato costruito in particolare sui dati riguardanti i valori del glucosio e del BMI, i quali sono stati suddivisi in 3 cluster



6) Menù del sistema

Appena avviamo il programma viene effettuata automaticamente l'ottimizzazione del dataset; quindi, abbiamo l'eliminazione dei valori nulli e il bilanciamento delle classi. Dopo questa fase compare il nostro menù:

```
1 -- Apprendimento Supervisionato
2 -- Apprendimento non supervisionato
3 -- Ottimizza i parametri
4 -- Esegui Test
5 -- Predizione
6 -- Esci dal programma
scegli un numero per iniziare ad esplorare il programma:
```

1) Si occupa di effettuare l'apprendimento supervisionato, quindi si occuperà di allenare i classificatori descritti prima, effettuando la Grid Search e la Repeated K-fold Cross-Validation, restituirà le informazioni riguardanti le metriche selezionate per ciascun modello

	model	accuracy	precision	recall	f1score
0	DecisionTree	0.835	0.847458	0.869565	0.858369
1	RandomForest	0.870	0.850394	0.939130	0.892562
2	KNN	0.820	0.784173	0.947826	0.858268
3	LogisticRegression	0.700	0.808989	0.626087	0.705882
4	MultiLayerPerc	0.750	0.803738	0.747826	0.774775
5	GaussianNB	0.690	0.791209	0.626087	0.699029

```
deviazione standard per il  DecisionTree: 0.09300537618869138

deviazione standard per il RandomForest: 0.06041522986797287

deviazione standard per il Knn: 0.050990195135927834

deviazione standard per il LogisticRegression: 0.0406201920231798

deviazione standard per il MultilayerPerceptron: 0.04301162633521314

deviazione standard per il GaussianNB: 0.04636809247747851
```

2) Si occupa di addestrare il k-means per poter poi suddividere i dati presenti nel dataset e quelli forniti in input per le predizioni, nei rispettivi cluster

	Insulin_T	BMI_T	DiabetesPedigreeFunction_T	Age_T	labels
269	-0.067263	-0.785842	-0.723173	-0.539941	1
283	-0.067263	-0.356948	-0.952391	1.095461	1
195	0.569473	0.974102	-0.249456	-0.453867	1
535	-0.067263	0.012788	-0.533686	-0.970310	1
38	-0.067263	0.796629	0.080617	-0.626015	2
269	-0.067263	-0.785842	-0.723173	-0.539941	1
314	-0.067263	0.456472	1.987709	0.751166	2
618	-0.067263	-0.682316	2.461426	1.353683	0
230	-0.067263	1.654417	0.514603	-1.056384	2
261	-0.067263	-0.416106	0.869127	-0.626015	1
399	-0.067263	0.308577	-0.720117	-0.798162	1
337	-0.067263	-0.238632	-0.408380	0.837240	0
394	-0.067263	0.012788	0.997488	-0.281720	1
746	-0.067263	2.438257	-0.362537	-0.626015	2
541	0.335601	-0.061159	0.221204	-0.626015	0
417	-0.067263	0.840997	0.236485	0.234723	2
39	0.534392	0.633945	2.791499	1.870125	2
731	-0.067263	-0.652737	-0.665104	-1.056384	0
683	-0.067263	-0.075949	0.181473	-0.626015	0
129	-0.067263	-0.726684	0.808002	2.386568	0

3) Attraverso la Grid Search verranno restituiti i parametri migliori per gli algoritmi di apprendimento supervisionato

```
 - Esecuzione del decision tree classifier con grid search
       Calcolo degli iperparametri ottimali...
- DTC Best Params: {'dtc__criterion': 'gini', 'dtc__max_depth': 74}
 - Esegui RFC con grid View
       Calcolo degli iperparametri ottimali
- RFC Best Params: {'rfc__bootstrap': False, 'rfc__criterion': 'gini', 'rfc__max_depth': 9, 'rfc__max_features': 'log2', 'rfc__n_estimators': 20}
 - Esegui KNC con grid view
       Calcolo degli iperparametri ottimali ...
- KNN Best Params: {'knn__algorithm': 'kd_tree', 'knn__n_neighbors': 120, 'knn__p': 2, 'knn__weights': 'distance'}
 - Execute LR with Grid View
       Calcolo degli iperparametri ottimali
- LR Best Params: {'logr__C': 0.1, 'logr__penalty': 'l2'}
 - Esegui MLP con grid view
       Calcolo degli iperparametri ottimali
- MLP Best Params: {'mlp__activation': 'relu', 'mlp__hidden_layer_sizes': (16,), 'mlp__solver': 'lbfgs'}
 - Esegui NB con Grid View
       Calcolo degli iperparametri ottimali
- NB Best Params: {'nb__var_smoothing': 0.1}
```

4) In questo caso vengono caricati i classificatori addestrati precedentemente durante la fase di training. Successivamente per ogni classificatore gli viene richiesto di effettuare un predict_proba su un input di test che per semplicità viene caricato automaticamente. I test caricati sono due reali presenti nel dataset, un caso di diabete e uno di non diabete. Una volta calcolate le predizioni di tutti i classificatori, viene fatta una media e viene restituita la percentuale più alta tra quelle delle due classi

```
TEST su un paziente non diabetico:  [1, 89, 66, 23, 94, 28.1, 0.167, 21]

predizione KMeans Cluster:  [2]
questo paziente non ha il diabete .
-Probabilità:  98.36 %

TEST su un paziente diabetico:  [2, 197, 70, 45, 543, 30.5, 0.158, 53]

predizione KMeans Cluster:  [0]
questo paziente ha il diabete .
-Probabilità:  97.11 %
```

5) Permette di effettuare una predizione basata su valori che inserisce l'utente in input. Alcuni valori non sono obbligatori, infatti l'utente può digitare -1 se non ricorda il valore, le uniche feature obbligatorie sono: glucose, Insulin e Age

```
Per favore inserisci i tuoi valori.
se un parametro è rosso è obbligatorio!
Inserisci il numero delle tue gravidanze:
1
Inserisci il tuo livello di glucosio nel sangue:
(>70)75
se non conosci la risposta inserisci '-1'.
Inserisci la tua pressione sanguigna diastolica:
(>40)45
se non conosci la risposta premi '-1'.
Inserisci lo spessore della tua pelle:
(>15)18
Inserisci il tuo livello di insulina:
(>16)19
se non conosci la risposta premi '-1'.
Inserisci la tua BMI:
(>19)22
se non conosci la risposta premi '-1'.
Inserisci la tua funzione di pedigree del diabete
(0<x<100)55
Inserisci la tua età:
26
```

```
valori del paziente:     [[1, 75.0, 45, 18.0, 19.0, 22.0, 55, 26]]
predizione KMeans Cluster:  [2]
questo paziente non ha il diabete .
-Probability:  62.78 %
```