



UNIVERSITÀ  
DEGLI STUDI DI BARI  
ALDO MORO

## DIABETES TEST

Previsioni per il diabete

**Corso di studi:** Ingegneria della conoscenza (2021-2022)

**Studente:** Alessia Viscuso

**Matricola:** 698862

**Repository GitHub:** [https://github.com/Alessia-oo/Progetto\\_diabete](https://github.com/Alessia-oo/Progetto_diabete)

## Sommario

<b>1. Introduzione</b>	3
<b>2. Requisiti funzionali</b>	3
2.1 Librerie utilizzate	3
2.2 Istruzioni per il primo utilizzo	3
<b>3. Dataset</b>	4
3.1 Informazioni sul dataset utilizzato	4
3.2 Valori nulli	6
3.3 Bilanciamento delle classi	7
3.4 Feature correlate	8
<b>4. Apprendimento supervisionato</b>	9
4.1 Dati standardizzati	9
4.2 Ottimizzazione degli iperparametri	9
4.3 K-fold cross validation	10
4.4 Modelli	10
<b>5. Apprendimento non supervisionato</b>	15
5.1 K-Means Clustering	15
<b>6. Interfaccia del sistema</b>	16
6.1 Learning Supervised	16
6.2 Learning Unsupervised	17
6.3 Optimize parameters	18
6.4 Run test	18
6.5 Prediction	19

## 1. Introduzione

Il diabete è una malattia cronica caratterizzata dalla presenza di elevati livelli di glucosio nel sangue (iperglicemia) e dovuta a un'alterata quantità o funzione dell'insulina.

L'insulina è l'ormone, prodotto dal pancreas, che consente al glucosio l'ingresso nelle cellule e il suo conseguente utilizzo come fonte energetica. Quando questo meccanismo è alterato, il glucosio si accumula nel circolo sanguigno.

Il sistema inizialmente caricherà e ottimizzerà i dati del dataset, successivamente addestrerà dei modelli per permettere di fare delle previsioni su dati preimpostati o inseriti dall'utente (non sarà obbligatorio inserirli tutti).

## 2. Requisiti funzionali

La realizzazione del progetto è stata effettuata in Python, in quanto uno dei linguaggi più adatti per la manipolazione e l'analisi dei dati. Come ambiente di sviluppo ho utilizzato PyCharm (Community Edition 2022.2.1).

### 2.1 Librerie utilizzate

Ho utilizzato le seguenti librerie:

- **Pandas:** per la gestione dei dataframe
- **Matplotlib, Seaborn:** per la gestione e la creazione dei grafici
- **Numpy:** per l'esecuzione delle adeguate operazioni su certi tipi di dato
- **Scikit-learn:** per la fase di classificazione e le metriche di valutazione
- **Os:** per poter colorare il testo in output
- **Warnings:** per la gestione dei warning
- **Pickle:** per salvare e caricare i modelli addestrati

### 2.2 Istruzioni per il primo utilizzo

Per avviare correttamente il programma è necessario provvedere all'installazione delle librerie presenti nel file *requirements*.

Pertanto, andare sul terminale presente in PyCharm e digitare il comando

```
pip install -r requirements.txt
```

Successivamente l'avvio del programma non dovrebbe dare problemi.

In caso contrario potrebbe essere necessario modificare il percorso di lettura del .csv e/o dei modelli salvati. Rispettivamente:

Andare nella cartella `loading_data`, nel file `AdjustDataset`, precisamente a riga 15; procedere inserendo il path corretto.

```
13
14     def optimizationData():
15         df = pd.read_csv(r"diabetes.csv")
16         pd.set_option('display.max_columns', None)
17         print("Dataset loaded.")
18
```

Figura 1: Modifica del path

Andare nel file main e modificare i path nelle righe 15-21.

```
dct_filename = 'Progetto_diab\models\dct_model.pkl'
knc_filename = 'Progetto_diab\models\knc_model.pkl'
rfc_filename = 'Progetto_diab\models/rfc_model.pkl'
mlp_filename = 'Progetto_diab\models/mlp_model.pkl'
lr_filename = 'Progetto_diab\models\lr_model.pkl'
nb_filename = 'Progetto_diab\models/nb_model.pkl'
kms_filename = 'Progetto_diab\models\kms_model.pkl'
```

Figura 1(2): Modifica del path

### 3. Dataset

#### 3.1 Informazioni sul dataset utilizzato

Il dataset utilizzato per l'addestramento del sistema è stato scaricato dal sito [www.kaggle.com](https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset) ed è visualizzabile tramite il seguente link:

<https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset>

Questo dataset proviene dall'Istituto Nazionale del Diabete e dei Disturbi Digestivi e dei Reni.

L'obiettivo è quello di effettuare delle predizioni, su delle misurazioni diagnostiche, sulla possibilità che un paziente sia diabetico o meno.

Sono stati imposti diversi vincoli sulla selezione delle istanze, provenienti da un database più ampio. In particolare, tutti i soggetti sono **donne di almeno 21 anni** discendenti di un gruppo di nativi americani denominati "Pima".

Nel dataset sono presenti 8 features continue e 1 feature dicotomica. In particolare:

- **Pregnancies:** indica il numero di gravidanze avute
- **Glucose:** indica la concentrazione di glucosio nel plasma a 2 ore dal 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 (peso in kg/(altezza in m)<sup>2</sup>)
- **DiabetesPedigreeFunction:** indica la probabilità di ereditare il diabete
- **Age:** indica l'età (anni)
- **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 (c.d. curva glicemica) *inferiori a 140 mg/dl sono ritenuti normali*, valori tra 140 e 199 mg/dl fanno porre diagnosi di

ridotta tolleranza ai carboidrati e infine valori superiori o uguali a 200 fanno porre diagnosi di diabete.

### BloodPressure

La pressione diastolica, o pressione minima, è il valore di pressione arteriosa nel momento in cui il cuore è in fase di rilassamento; in altre parole, è la pressione sanguigna tra due battiti cardiaci.

Secondo i ricercatori la pressione troppo alta è pericolosa, ma nei pazienti diabetici (di tipo 2) 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 spesso disturbi e malattie cutanee connesse all'alterato metabolismo del glucosio. Le persone affette da diabete tendono ad avere una pelle più spessa rispetto ai non diabetici e in generale tale incremento di spessore inapparente, ma clinicamente misurabile, non è associato ad alcun sintomo ed è sconosciuto al paziente e al medico.

La pelle umana ha uno spessore che varia da *0,5 a 3 mm*.

### 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 min. 20 – 90 micr.UI/ml
- *dopo 120 min. 18 – 56 micr.UI/ml*

### BMI

BMI è l'acronimo di Body Mass Index (Indice di Massa Corporea).

Il calcolo del BMI è un sistema di valutazione del peso, riferito al rischio di malattia, che richiede due valori noti: statura e peso. Il suo valore permette di stabilire se la 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<sup>2</sup> nell'uomo e a 21 kg/m<sup>2</sup> nella donna).

### DiabetesPedigreeFunction

È la funzione che valuta la probabilità di diabete in base alla storia familiare. I suoi valori variano quindi da 0 a 100.

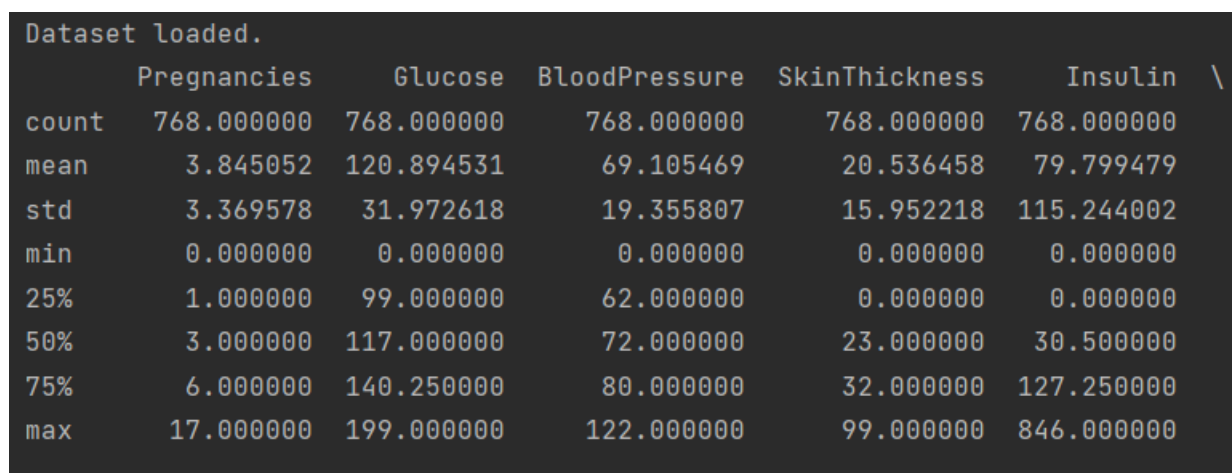
### 3.2 Valori nulli

In questa fase il dataset verrà analizzato, ed eventualmente modificato, per poter essere sfruttato al meglio successivamente.

Una delle cose più importanti da verificare è la presenza, o assenza, di valori nulli. In un primo momento, avendo cercato tali valori attraverso il comando

```
print(df.isnull().sum())
```

sembrava che non ce ne fossero, in quanto accanto ad ogni feature compariva uno 0. Tuttavia, come si può notare in questa figura



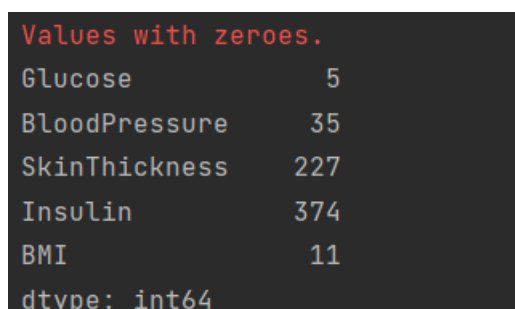
Dataset loaded.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin \
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479
std	3.369578	31.972618	19.355807	15.952218	115.244002
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000
75%	6.000000	140.250000	80.000000	32.000000	127.250000
max	17.000000	199.000000	122.000000	99.000000	846.000000

Figura 2: Describe del dataset

Il valore minimo di molte feature (non sono mostrate tutte in figura per motivi di spazio) è pari a zero, il che è anomalo. Dunque sono presenti dei valori nulli.

Dopo aver sostituito gli zeri presenti in tutte le feature ad eccezione di Pregnancies, DiabetesPedigreeFunction, Age e Outcome il risultato del comando precedente è risultato questo:



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

Figura 3: Valori nulli

Ho provveduto a sostituire questi valori nulli con la media delle rispettive colonne. In questo modo il conteggio dei valori nulli è effettivamente di zero.

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

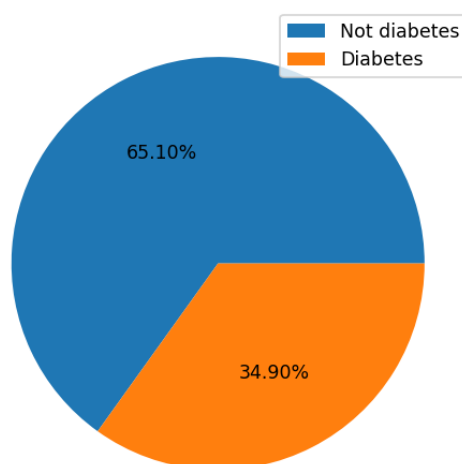
Figura 4: Numero di valori nulli dopo la modifica

### 3.3 Bilanciamento delle classi

Il prossimo passo è quello di visualizzare il bilanciamento delle classi, un problema molto comune.

Infatti è importante avere lo stesso numero di esempi di entrambe le classi per poter avere una maggiore efficienza in fase di apprendimento.

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 più difficoltoso (e inesatto) l'apprendimento.



```
Check for class balance!
Not diabetes:  500 (% 65.10)
Diabetes:      268 (% 34.90)
```

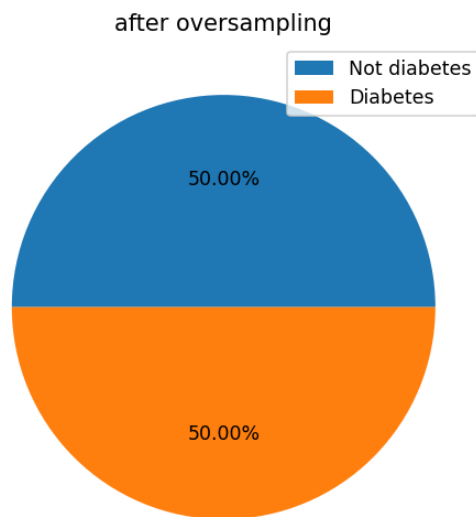
Figura 5: Bilanciamento iniziale

Come si può vedere le classi sono sbilanciate.

Ho deciso di bilanciarle attraverso l'**oversampling**. L'oversampling 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 il resample ho ottenuto le classi bilanciate.



```
Value after oversampling:
Not diabetes: 500 (% 50.00)
Diabetes: 500 (% 50.00)
```

Figura 6: Bilanciamento dopo l'oversampling

### 3.4 Feature correlate

Una volta bilanciate le classi ho cercato dei fattori che potessero caratterizzare un paziente diabetico.

Sulla base delle features del dataset, 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 presente tra le varie features, e verificare quest' ipotesi, ho generato una heatmap.

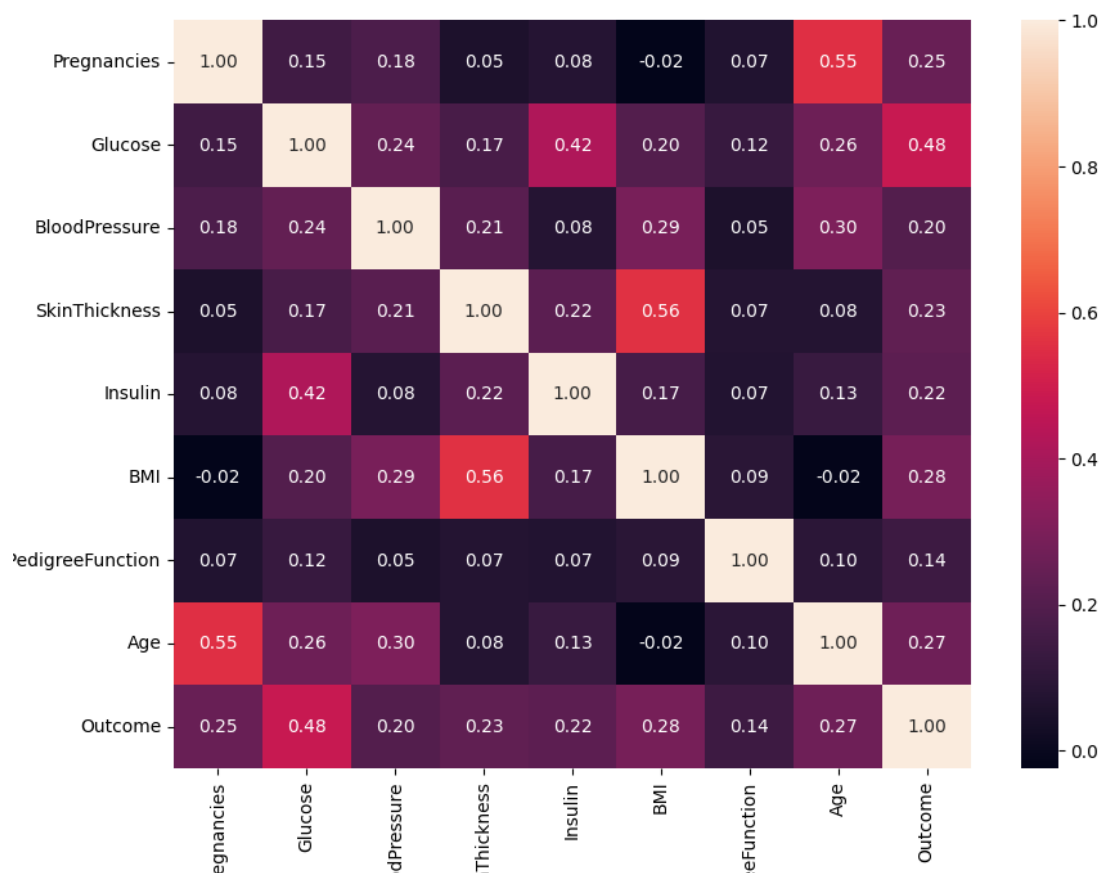


Figura 7: Heatmap



Dalla heatmap si nota che ci sono sintomi mediamente correlati tra di loro. Come teorizzato, l'esito di Outcome dipende soprattutto dal valore del glucosio (anche se non in modo forte), il quale a sua volta è mediamente correlato al valore dell'insulina.

Nelle fasi successive ho addestrato una serie di classificatori determinandone anche i migliori iperparametri, al fine di poter fornire dei risultati più accurati.

## 4. Apprendimento supervisionato

Prima di andare a descrivere i modelli utilizzati parlerò di due funzioni che ho utilizzato per migliorare il loro rendimento finale: StandardScaler() e GridSearchCV().

### 4.1 Dati standardizzati

La riduzione di scala standardizza una serie di valori. 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 ( $\mu$ ) da ogni valore ( $x$ ) del vettore e dividere la differenza per la devianza standard ( $\sigma$ ).

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

$x$ =vettore  
 $\mu$ =media  
 $\sigma$ =devianza standard

Questo è importante, infatti le performance del processo di learning sono generalmente superiori quando l'algoritmo lavora su dati standardizzati.

### 4.2 Ottimizzazione degli iperparametri

L'ottimizzazione degli iperparametri è fondamentale per il corretto funzionamento dei modelli di Machine Learning. Il metodo Grid Search è uno strumento di base per questa ottimizzazione, infatti considera diverse combinazioni di iperparametri e sceglie quella che restituisce un punteggio di errore inferiore.

In **GridSearchCV**, insieme alla Grid Search, viene eseguita anche la cross-validation (omettendo il parametro cv, questo sarà uguale a 5).

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

Figura 8: Esempio di Pipeline con StandardScaler() e GridSearchCV() per il Decision Tree

### 4.3 K-fold cross validation

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.

### 4.4 Modelli

Come classificatori ho scelto:

- Decision Tree
- Random Forest
- 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

#### 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.

Risultati Decision Tree:

	<b>Non ottimizzato</b>	<b>Ottimizzato</b>
<b>Accuracy</b>	0.83	<b>0.85</b>
<b>Precision</b>	0.84	0.82
<b>Recall</b>	0.83	<b>0.88</b>
<b>F1-score</b>	0.84	<b>0.85</b>

**Deviazione standard:** 0.07

### 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
<b>Accuracy</b>	0.88	<b>0.89</b>
<b>Precision</b>	0.87	0.85
<b>Recall</b>	0.88	<b>0.95</b>
<b>F1-score</b>	0.88	<b>0.89</b>

**Deviazione standard:** 0.04

### 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.

Risultati KNN:

	Non ottimizzato	Ottimizzato
<b>Accuracy</b>	0.82	0.80
<b>Precision</b>	0.82	0.71
<b>Recall</b>	0.83	<b>0.95</b>
<b>F1-score</b>	0.82	<b>0.89</b>

**Deviazione standard:** 0.03

### 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.

Risultati Logistic Regression:

	Non ottimizzato	Ottimizzato
<b>Accuracy</b>	0.76	0.76
<b>Precision</b>	0.77	0.77
<b>Recall</b>	0.76	0.73
<b>F1-score</b>	0.76	0.75

**Deviazione standard:** 0.04

## 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 tutte le feature).

Un layer lineare completo, il cui modello matematico è

$$y = \sum_i w_i * x_i$$

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.

Risultati Multilayer Perceptron:

	Non ottimizzato	Ottimizzato
<b>Accuracy</b>	0.80	<b>0.85</b>
<b>Precision</b>	0.80	<b>0.82</b>
<b>Recall</b>	0.79	<b>0.88</b>
<b>F1-score</b>	0.80	<b>0.85</b>

**Deviazione standard:** 0.03

## Gaussian Naive Bayes

Il Naive Bayes è un algoritmo di classificazione basato sul Teorema di Bayes.

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

Dove:

- **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

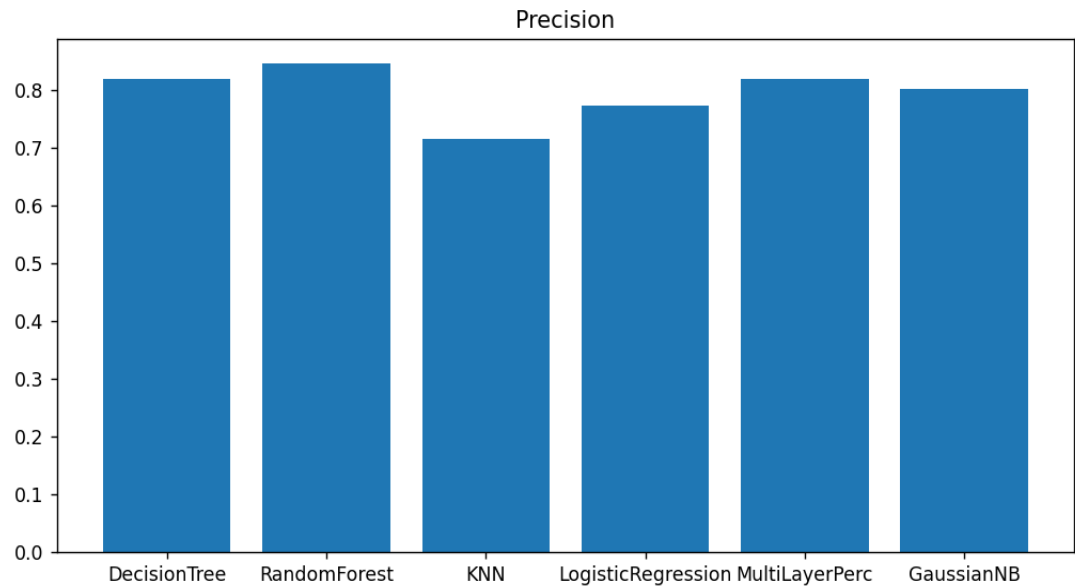
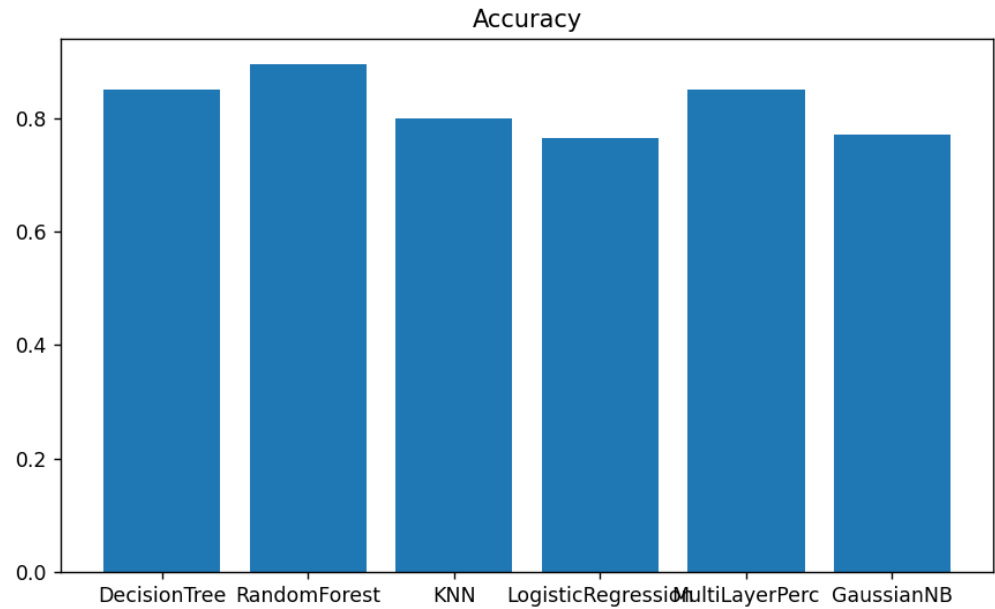
Gaussian Naive Bayes è una variante di Naive Bayes che segue la distribuzione normale gaussiana esupporta dati continui.

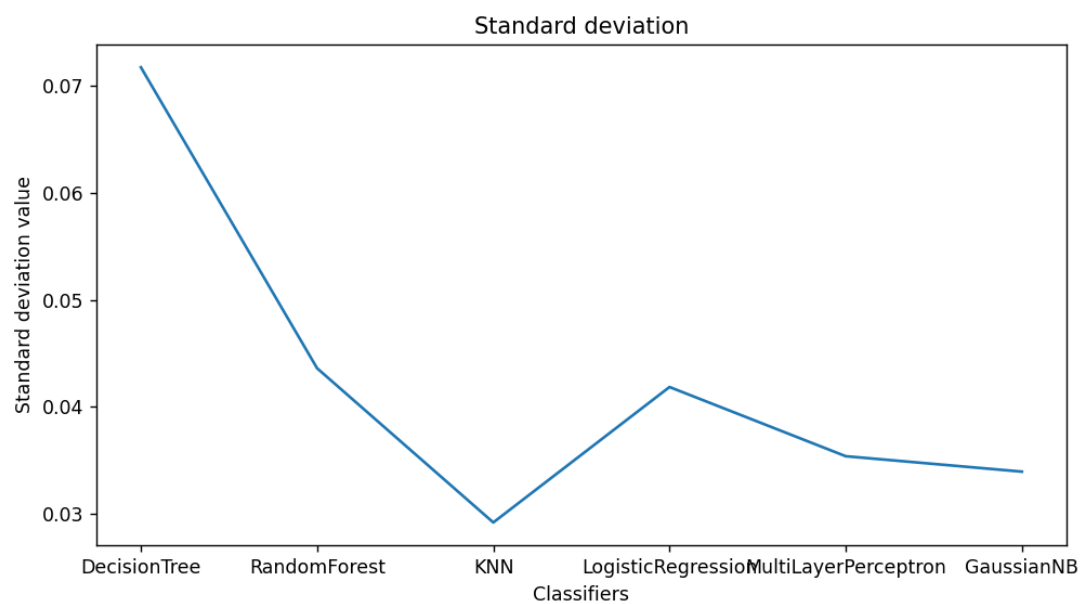
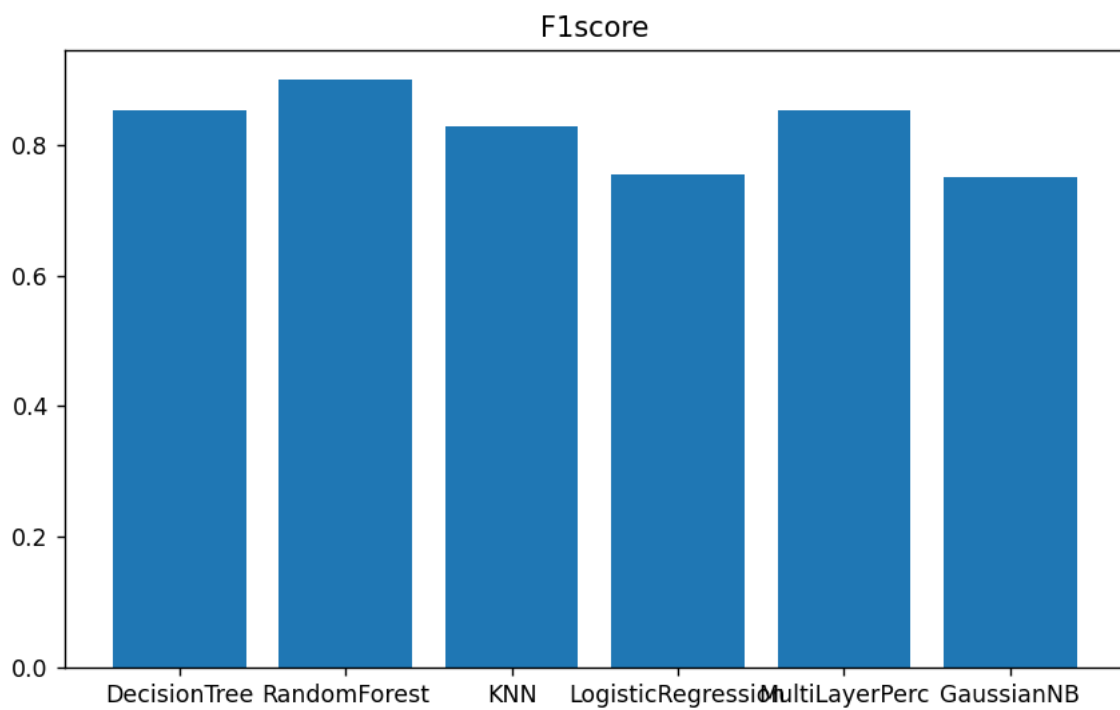
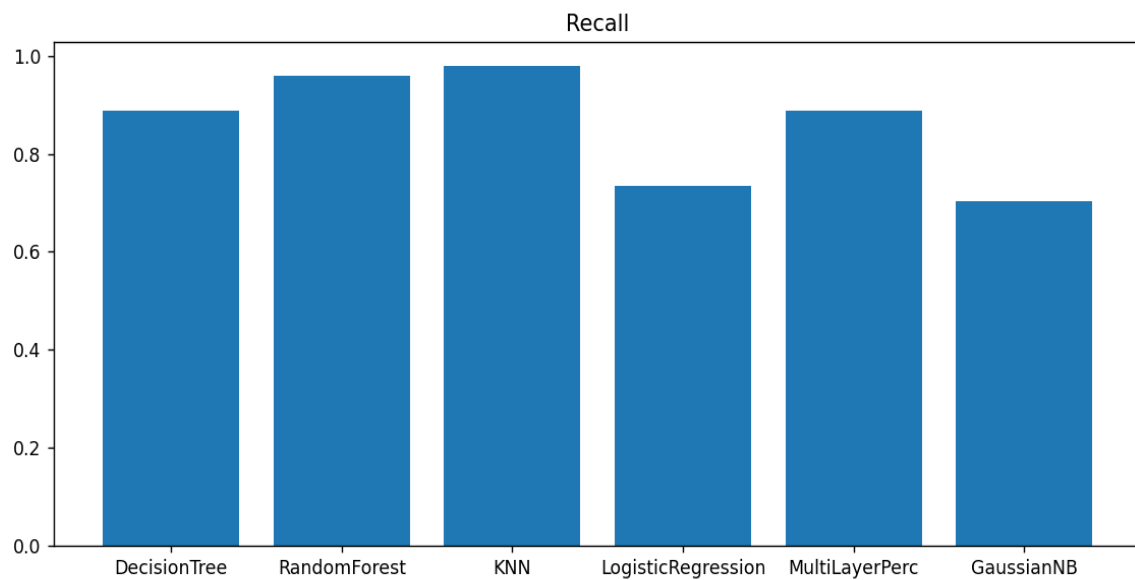
Risultati Naive Bayes:

	Non ottimizzato	Ottimizzato
<b>Accuracy</b>	0.72	<b>0.77</b>
<b>Precision</b>	0.73	<b>0.80</b>
<b>Recall</b>	0.73	0.70
<b>F1-score</b>	0.73	<b>0.75</b>

**Deviazione standard: 0.03**

Ecco i grafici risultanti:





Come si può vedere il modello meglio performante è il **Random Forest**, seguito dal Multilayer Perceptron, mentre il peggiore sembra essere il Logistic Regression.

Ora possiamo utilizzare questi modelli.

## 5. Apprendimento non supervisionato

Per apprendimento non supervisionato si intende un apprendimento in cui non ci sono feature-obiettivo negli esempi di training. L'apprendimento non supervisionato è usato per esplorare dati sconosciuti. Può rivelare schemi che potrebbero essere sfuggiti o esaminare grandi insiemi di dati che sarebbero troppo grandi da affrontare per un umano.

### 5.1 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.

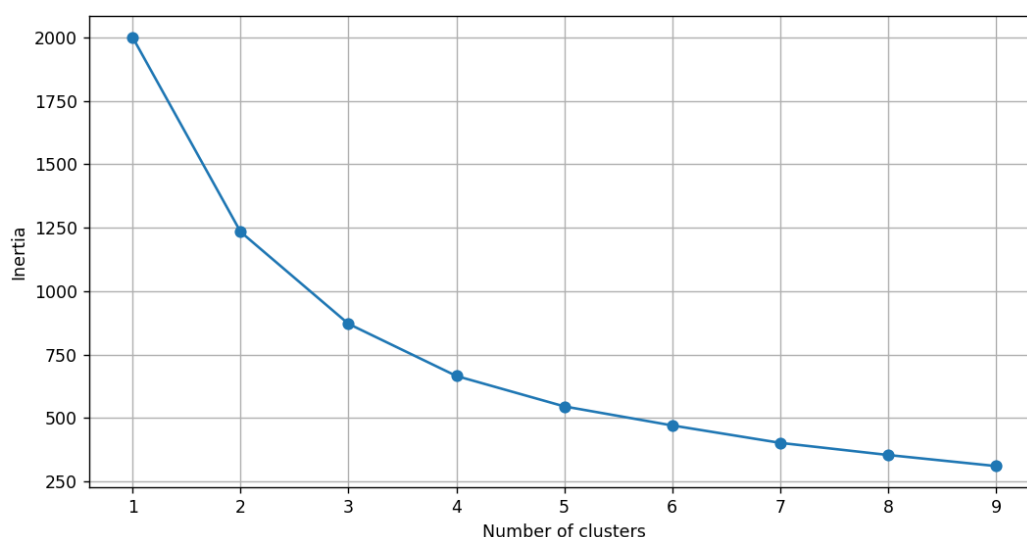
In questo caso ho eliminato alcune feature dal dataset: Pregnancies, BloodPressure, SkinThickness e Outcome.

Ho poi standardizzato i dati, così da avere solo valori nel range di -1 e 1.

Una volta standardizzati i dati è importante scegliere il numero di clusters che si vuole identificare (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 ottimale di cluster da eseguire, ottenendo questo risultato:



Come si può vedere, il numero di cluster ottimale è 3; dunque il modello è stato costruito in particolare sui dati riguardanti i valori del glucosio e del BMI, che sono stati suddivisi in 3 cluster.

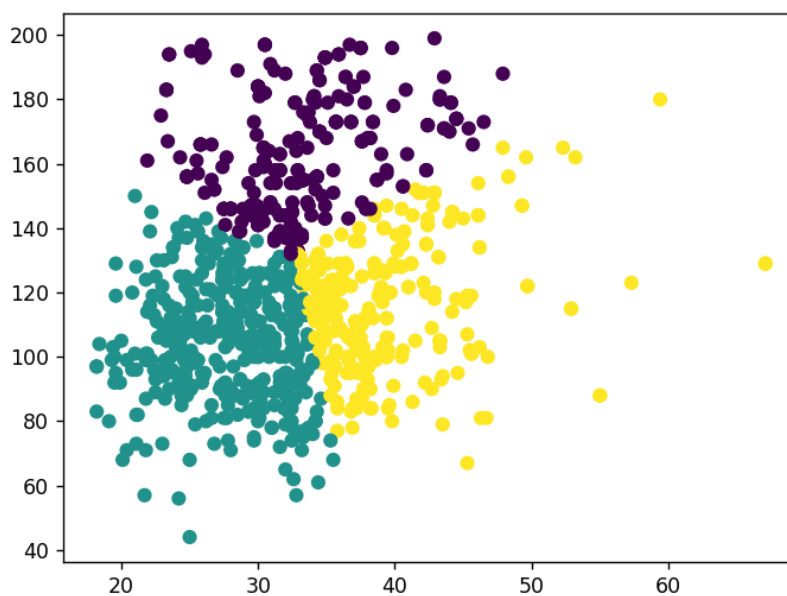


Figura 9: Cluster dei pazienti presenti nel dataset

## 6. Interfaccia del sistema

Una volta avviato il programma verrà automaticamente eseguita l'ottimizzazione del dataset, quindi l'eliminazione dei valori nulli e il bilanciamento delle classi.

Successivamente saranno visualizzate le operazioni a disposizione dell'utente, tramite un menù.

```
Diabete prediction

1 -- Learning Supervised
2 -- Learning Unsupervised
3 -- Optimize parameters
4 -- Run test
5 -- Prediction
6 -- Exit
Enter your choice:
```

Figura 10: Menù del sistema

### 6.1 Learning Supervised

La funzione Learning Supervised si occuperà di allenare i classificatori descritti precedentemente, effettuando la Grid Search e la Repeated K-Fold Cross-Validation, e restituirà le informazioni riguardanti le metriche selezionate per ciascun modello.

	model	accuracy	precision	recall	f1score
0	DecisionTree	0.850	0.820755	0.887755	0.852941
1	RandomForest	0.895	0.846847	0.959184	0.899522
2	KNN	0.800	0.716418	0.979592	0.827586
3	LogisticRegression	0.765	0.774194	0.734694	0.753927
4	MultiLayerPerc	0.850	0.820755	0.887755	0.852941
5	GaussianNB	0.770	0.802326	0.704082	0.750000

Figura 11: Metriche risultanti dei modelli



```

Standard deviation for DecisionTree: 0.07176350047203664

Standard deviation for RandomForest: 0.04358898943540674

Standard deviation for Knn: 0.029154759474226483

Standard deviation for LogisticRegression: 0.04183300132670381

Standard deviation for MultilayerPerceptron: 0.03535533905932736

```

*Figura 12: Standard deviation dei modelli*

## 6.2 Learning Unsupervised

La funzione Learning Unsupervised addestrerà 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	0
38	-0.067263	0.796629	0.080617	-0.626015	0
269	-0.067263	-0.785842	-0.723173	-0.539941	1
314	-0.067263	0.456472	1.987709	0.751166	0
618	-0.067263	-0.682316	2.461426	1.353683	2
230	-0.067263	1.654417	0.514603	-1.056384	0
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	2
394	-0.067263	0.012788	0.997488	-0.281720	1
746	-0.067263	2.438257	-0.362537	-0.626015	0
541	0.335601	-0.061159	0.221204	-0.626015	2
417	-0.067263	0.840997	0.236485	0.234723	0
39	0.534392	0.633945	2.791499	1.870125	0
731	-0.067263	-0.652737	-0.665104	-1.056384	2
683	-0.067263	-0.075949	0.181473	-0.626015	2
129	-0.067263	-0.726684	0.808002	2.386568	2
Model saved!					

*Figura 13: La feature labels indica il cluster di appartenenza di ogni paziente presente nel dataset*

## 6.3 Optimize parameters

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

```
- Execute DCT with Grid View
  Calculating optimal hyperparameters...
- DTC Best Params: {'dtc__criterion': 'gini', 'dtc__max_depth': 56}
- Execute RFC with Grid View
  Calculating optimal hyperparameters...
- RFC Best Params: {'rfc__bootstrap': True, 'rfc__criterion': 'gini', 'rfc__max_depth': 9, 'rfc__max_features': 'sqrt', 'rfc__n_estimators': 90}
- Execute KNC with Grid View
  Calculating optimal hyperparameters...
- KNN Best Params: {'knn__algorithm': 'kd_tree', 'knn__n_neighbors': 120, 'knn__p': 2, 'knn__weights': 'distance'}
- Execute LR with Grid View
  Calculating optimal hyperparameters...
- LR Best Params: {'logr__C': 0.1, 'logr__penalty': 'l2'}
- Execute MPL with Grid View
  Calculating optimal hyperparameters...
- MLP Best Params: {'mlp__activation': 'logistic', 'mlp__hidden_layer_sizes': (20,), 'mlp__solver': 'lbfgs'}
```

Figura 14: Migliori iperparametri selezionati dalla Grid Search

## 6.4 Run test

Con la funzione Run test 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 casi 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, accompagnata da una frase esplicativa.

```
TEST: [1, 89, 66, 23, 94, 28.1, 0.167, 21]

DCT prediction: [0.]
RFC prediction: [0.00423077]
KNN prediction: [0.]
LR prediction: [0.0749744]
MLP prediction: [0.00228169]
NB prediction: [0.01718537]
```

Figura 15: Risultati modelli su un caso reale di non diabete

```
TEST: [2, 197, 70, 45, 543, 30.5, 0.158, 53]

DCT prediction: [1.]
RFC prediction: [0.91171389]
KNN prediction: [1.]
LR prediction: [0.91578357]
MLP prediction: [0.99925663]
NB prediction: [0.99988867]
```

Figura 16: Risultati modelli su un caso reale di diabete

```

TEST on a real non-diabetes patient: [1, 89, 66, 23, 94, 28.1, 0.167, 21]

KMeans prediction Cluster: [2]
This patient doesn't have diabetes .
-Probability: 98.36 %

TEST on a real diabetes patient: [2, 197, 70, 45, 543, 30.5, 0.158, 53]

KMeans prediction Cluster: [1]
This patient has diabetes .
-Probability: 97.11 %

```

*Figura 17: Probabilità su casi reali di non diabete e diabete*

## 6.5 Prediction

Infine, l'ultima funzione permette di effettuare una predizione basata sui valori che l'utente potrà inserire in input. Non è obbligatorio inserire tutte le features richieste, si può infatti inserire '-1' all'occorrenza. Le feature obbligatorie sono le seguenti: Glucose, Insulin e Age.

Ecco alcuni risultati:

```

If you don't know the answer, please digit '-1'.
Insert your skin thickness:
(>15)17
Insert your insulin level:
(>16)30
If you don't know the answer, please digit '-1'.
Insert your BMI:
(>19)25

```

*Figura 18: Parte dei valori inseriti (non-diab)*

```

Patient values: [[2, 80.0, 69, 20.0, 50.0, 25.0, 50, 30]]
KMeans prediction Cluster: [2]
This patient doesn't have diabetes .
-Probability: 61.67 %

```

*Figura 19: Previsione effettuata sui dati inseriti (non-diab)*

```
3
Insert your glucose level in blood:
(>70)150
If you don't know the answer, please digit '-1'.
Insert your diastolic blood pressure:
(>40)96
If you don't know the answer, please digit '-1'.
Insert your skin thickness:
(>15)25
```

*Figura 20: Parte dei valori inseriti (diab)*

```
Patient values:  [[2, 340.0, 130, 25.0, 80.0, 40.0, 50, 40]]
KMeans prediction Cluster:  [0]
This patient has diabetes .
-Probability:  69.99 %
```

*Figura 21: Previsione effettuata sui dati inseriti (diab)*