Analisi e previsione della diagnosi del diabete

Gruppo di lavoro

- Perchiazzi Antonio Maria, 760459, a.perchiazzi4@studenti.uniba.it

repository github

AA 2023-24

Introduzione

Il diabete rappresenta una delle sfide più significative per la salute pubblica a livello globale. Con un numero crescente di persone affette da questa patologia, è fondamentale comprendere le dinamiche che ne influenzano l'insorgenza e la progressione. Questo progetto si propone di utilizzare un dataset ricco di dati su pazienti clinici con l'obiettivo di identificare fattori di rischio e creare modelli per predirne la diagnosi.

Elenco argomenti di interesse

- 2.Apprendimento supervisionato
- 3. Reti Neurali
- 4. Reti Bayesiane

1.EDA

Librerie utilizzate:

```
import pandas as pd
import seaborn as sns                    #visualisation
import matplotlib.pyplot as plt          #visualisation
```

1.1 Dataset

Il dataset utilizzato contiene dati clinici di circa 100'000 pazienti americani, con e senza diabete

```
df = pd.read_csv("../Dataset/diabetes_dataset.csv")

df.info(verbose= True)
```

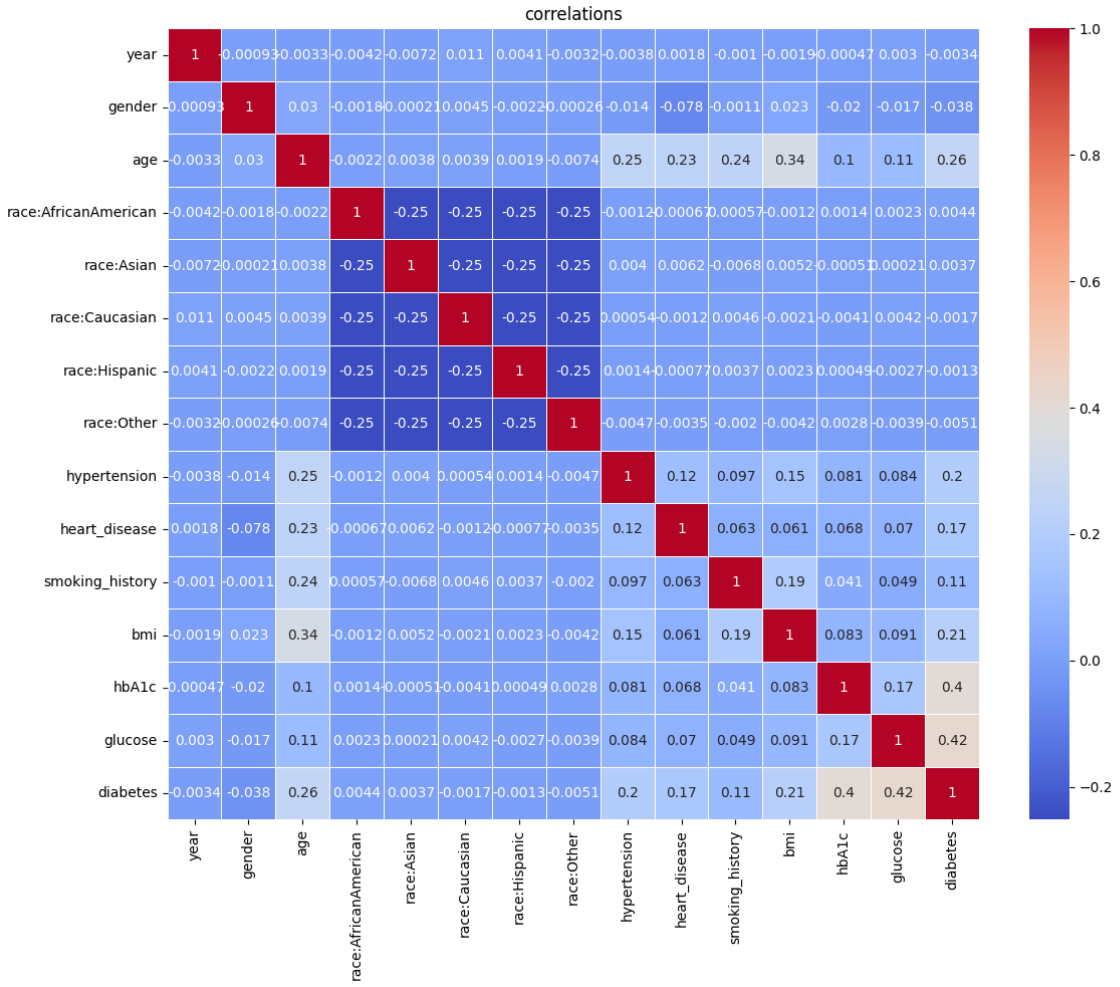
```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   year                 99982 non-null  int64
 1   gender               99982 non-null  int64
 2   age                  99982 non-null  float64
 3   race:AfricanAmerican 99982 non-null  int64
 4   race:Asian           99982 non-null  int64
 5   race:Caucasian       99982 non-null  int64
 6   race:Hispanic        99982 non-null  int64
 7   race:Other           99982 non-null  int64
 8   hypertension         99982 non-null  int64
 9   heart_disease        99982 non-null  int64
 10  smoking_history      99982 non-null  int64
 11  bmi                  99982 non-null  float64
 12  hbA1c                99982 non-null  float64
 13  glucose              99982 non-null  int64
 14  diabetes             99982 non-null  int64
```

Dove:

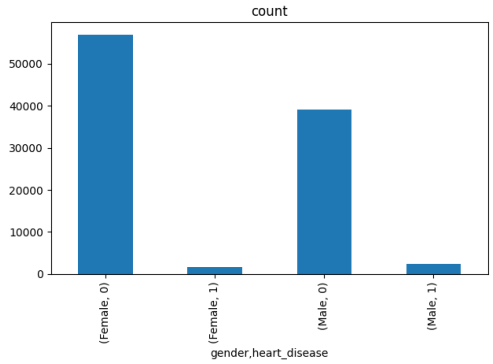
- Year: l'anno in cui sono stati raccolti i dati
- Gender: genere del paziente (0 : maschio, 1 : femmina)
- Age: l'età del paziente
- Race: l'etnia del paziente (0: appartiene, 1: non appartiene)

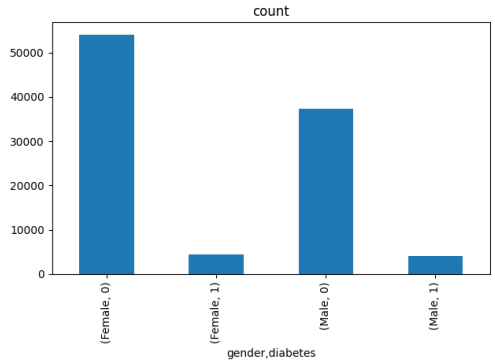
- Hypertension: se il paziente soffra di ipertensione (0: no, 1: sì)
- Heart_disease: se il paziente soffra di malattie al cuore (0: no, 1: sì)
- smoking_history: lo stato di fumatore del paziente (0: no Data, 1: mai fumato, 2: fumato in passato, 3: fumatore)
- bmi: il bmi del paziente
- hbA1c: livelli di emoglobina glicata, media dei livelli di glucosio negli ultimi 2-3 mesi
- glucose: il livello di glucosio in mg/dl
- diabetes: se il paziente abbia o meno il diabete (0: no, 1: sì)

Correlazioni tra i dati:

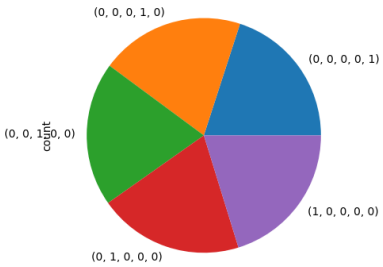


- gender: ha una lieve(ma trascurabile) correlazione sia con hearth_disease che con diabetes, il valore negativo è dovuto al fatto che gli uomini (più probabili di sviluppare entrambe le condizioni) siano rappresentati da 0, mentre la diagnosi positiva per entrambe sia 1, controllando più attentamente si è notato che nonostante il numero di uomini nel dataset sia leggermente inferiore(54'000 contro 37'000) il numero di pazienti con diabete o malattie cardiache non vari di molto fra i sessi.





- Age: il dataset presenta un picco di pazienti intorno agli 80 anni, ha una buona correlazione con tutti gli indicatori successivi, in particolare ha la terza correlazione più alta con la diagnosi di diabete
- Race: il dataset ha una buona suddivisione tra le etnie, non presenta overlap tra esse:



- Hypertension/heart_disease: simili ad age (con cui sono entrambi più correlati)
- Smoking_history/bmi: oltre che alla solita correlazione con age e gli altri indicatori, hanno una buona correlazione tra di loro
- hbA1c: seconda più forte correlazione con la diagnosi di diabete
- glucose: correlazione più forte con la diagnosi di diabete

In conclusione: glucose, hbA1c, age, bmi, smoking_history, heart_disease, hypertension sono le caratteristiche più rilevanti per la diagnosi del diabete e verranno utilizzate come features nell'addestramento dei modelli.

2.Apprendimento supervisionato

Sommario

Il caso di Studio si preoccupa di predire se un paziente abbia il diabete, in particolare utilizzando le features selezionate per predire il valore di 'diabetes'

I modelli con cui si è scelto di sperimentare sono:Naive Bayes, Decision Tree, Random Forest, K-nearest neighbors.

Strumenti utilizzati

```
import matplotlib.pyplot as plt
import pandas as pd

from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
from sklearn.model_selection import cross_val_score, train_test_split

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

from statistics import median
```

Decisioni di Progetto

Abbiamo testato i modelli senza modificarne i parametri

Naive Bayes score:

	precision	recall	f1-score	support
0	0.80	0.88	0.84	2218
1	0.87	0.78	0.82	2155
accuracy			0.83	4373
macro avg	0.83	0.83	0.83	4373
weighted avg	0.83	0.83	0.83	4373

tree score:

	precision	recall	f1-score	support
0	0.87	0.86	0.86	2112
1	0.86	0.87	0.86	2115
accuracy			0.86	4227
macro avg	0.86	0.86	0.86	4227
weighted avg	0.86	0.86	0.86	4227

forest score:

	precision	recall	f1-score	support
0	0.88	0.89	0.89	2112
1	0.89	0.88	0.89	2115
accuracy			0.89	4227
macro avg	0.89	0.89	0.89	4227
weighted avg	0.89	0.89	0.89	4227

KNN score:

	precision	recall	f1-score	support
0	0.88	0.87	0.87	2112
1	0.87	0.88	0.87	2115
accuracy			0.87	4227
macro avg	0.87	0.87	0.87	4227
weighted avg	0.87	0.87	0.87	4227

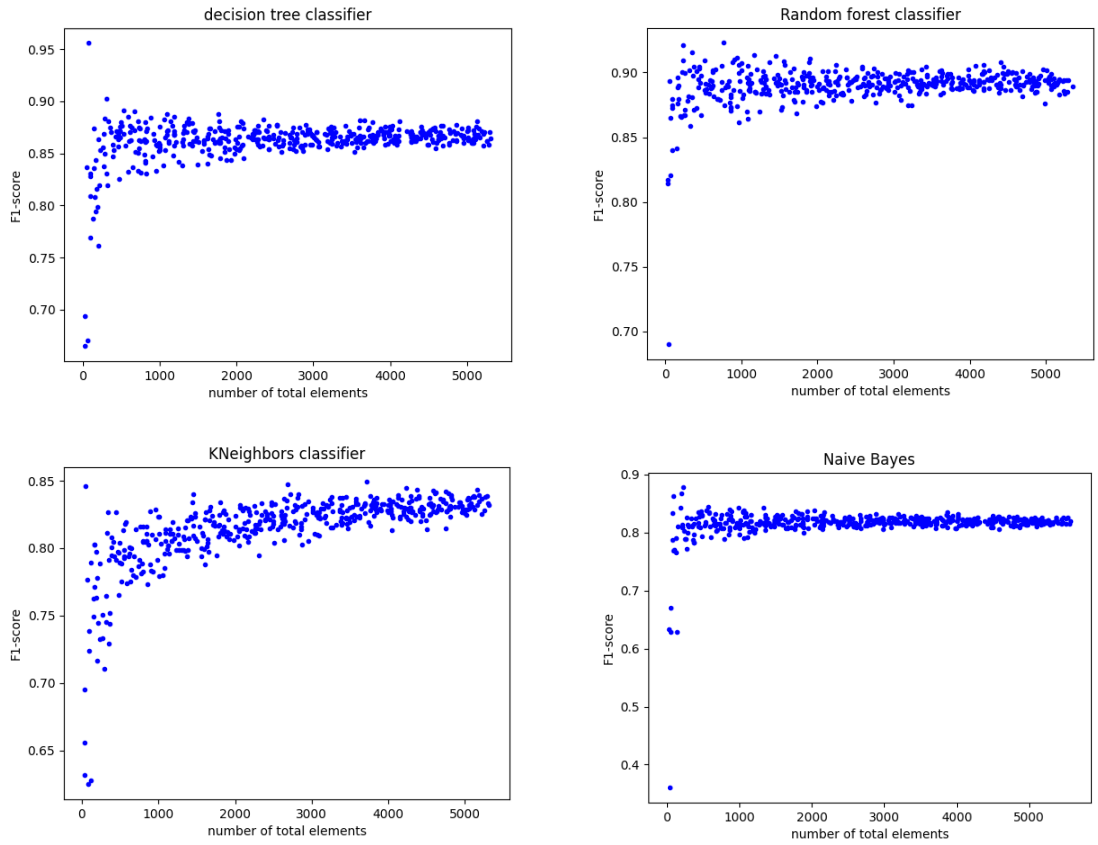
Precision: la misura dei casi positivi identificati correttamente tra tutti i positivi, serve a ridurre il numero di falsi positivi

Recall: la misura di tutti i casi positivi identificati tra tutti i casi positivi, serve a minimizzare i falsi negativi

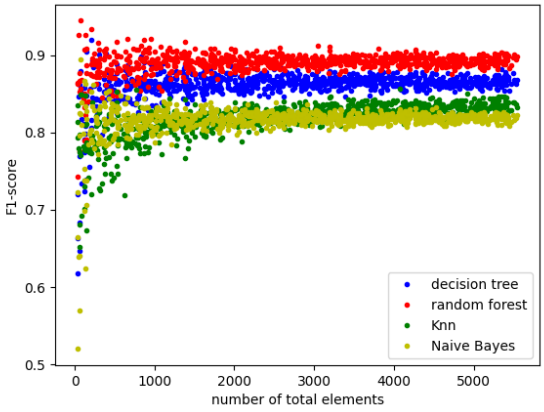
Accuracy: una misura di tutti i casi correttamente classificati, utile quando le classi sono ugualmente importanti ed il dataset è bilanciato

F1-score: media armonica di precision e recall, una misura migliore per i casi classificati incorrettamente, non soffre troppo di classi sbilanciate ed è utilizzato quando falsi positivi e negativi sono cruciali (anche un solo paziente diabetico non identificato, o uno sano a cui viene consigliata una terapia errata è un problema, andremo quindi ad utilizzare questa metrica per giudicare i modelli.)

I risultati elevati potrebbero essere attribuiti alle dimensioni del dataset:



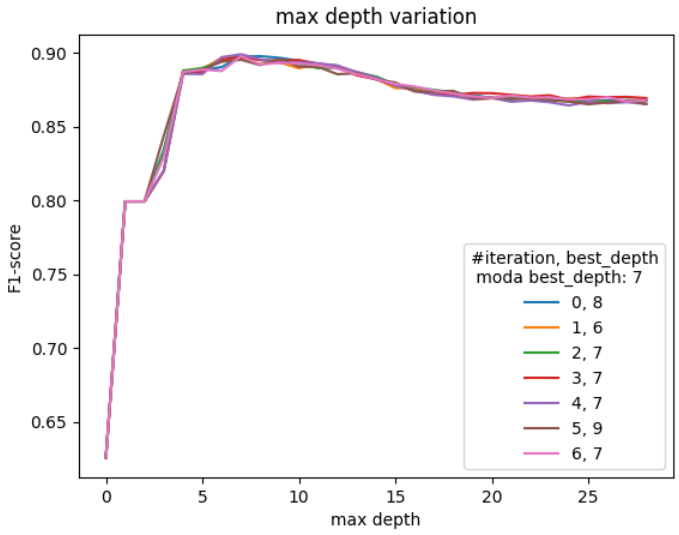
la variazione fra test diversi va a ridursi con l'aumentare degli elementi (test+train), e generalmente vanno a compattarsi oltre i 2500, inoltre:



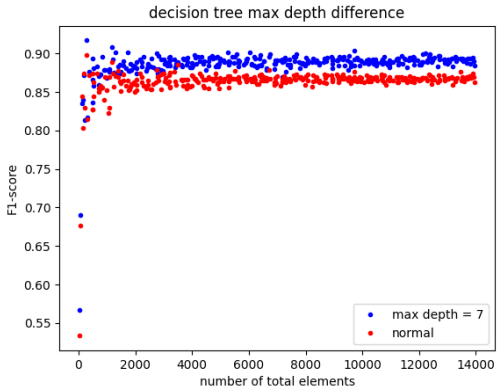
senza modificare i parametri, il modello Random Forest performa meglio degli altri, mentre Naive bayes ha lo score più basso ma una variazione di molto inferiore.

2.1 Decision Tree

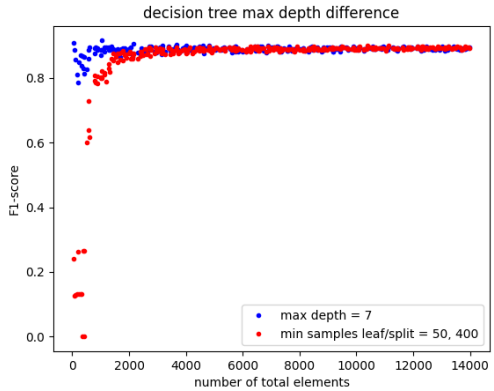
Iniziamo dal valutare come l'F1-score cambi al variare della profondità massima, un valore elevato può portare ad overfitting, mentre uno basso può causare underfitting:



attraverso diverse iterazioni, si nota come lo score più alto sia raggiunto più comunemente con una profondità di 7, è interessante notare come questo sia lo stesso numero di features su cui vengano fatte le predizioni, tuttavia questo sembrerebbe un caso, in quanto diminuendone il numero la profondità ottimale tende invece ad aumentare.



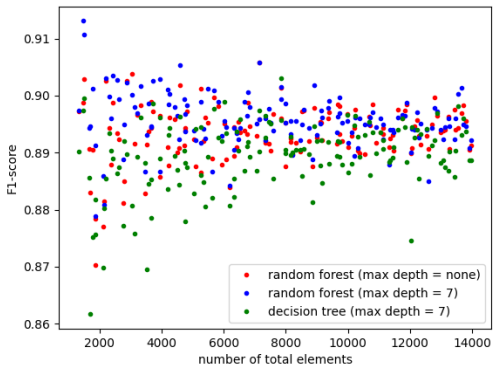
-Le prestazioni del modello aumentano lievemente con l'utilizzo di questo parametro



-Si è sperimentato anche con l'utilizzo dei parametri min_samples_leaf e min_samples_split, che tuttavia hanno portato a risultati migliori del max_depth da solo

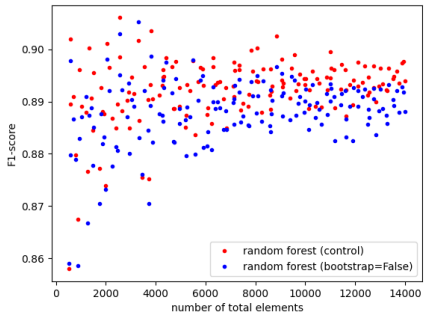
2.2 Random Forest

Iniziamo confrontando il modello con la versione "migliorata" di decision tree:



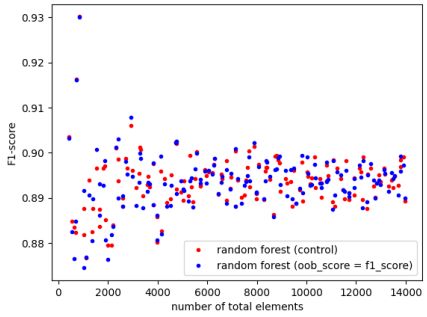
Possiamo notare che nonostante i miglioramenti, random forest sia più performante (seppur di poco), e l'utilizzo di max_depth non influisce in modo notevole sui risultati di random forest,

andiamo quindi a sperimentare su altri parametri:



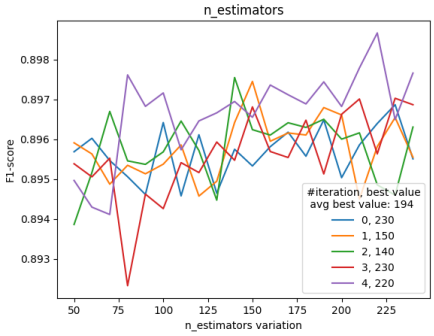
bootstrap: Se vengano utilizzati dei campioni per costruire gli alberi, se falso viene utilizzato l'intero dataset.

Come si vede in figura, portare questo parametro a false riduce leggermente la performance del modello



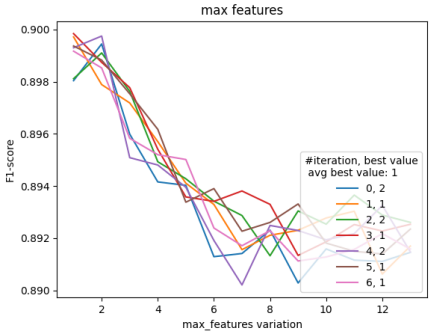
oob_score: Come vengano giudicati gli alberi all'interno della random forest, di default è 'accuracy'.

Utilizzando F1-score non si è notata una variazione nei risultati.



n_estimators: il numero di alberi(estimatori) utilizzati nel modello, di default è 100

Si è notato un lieve miglioramento intorno ai 170 estimatori

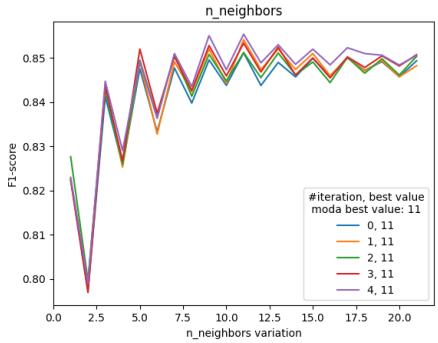


max_features: Il numero massimo di features controllate nei nodi degli alberi, deafult = n_features

risultati diminuiscono all'aumentare del valore, si stabilizzano una volta raggiunto il numero di features,

valore migliore: 1/2

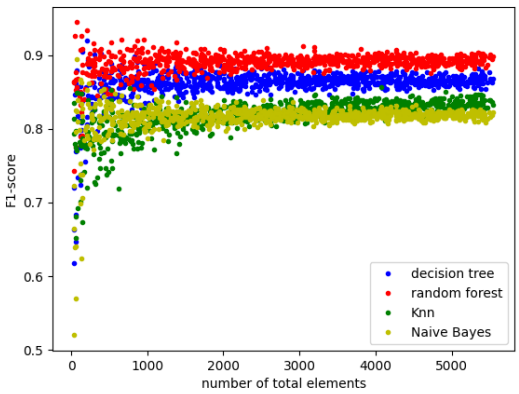
2.3 KNN

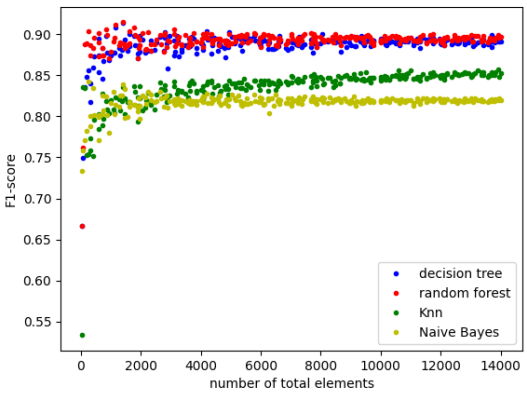


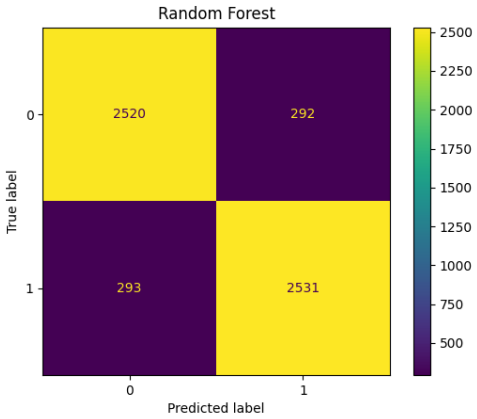
N_neigbors: il numero di vicini presi in considerazione durante la classificazione, default = 5

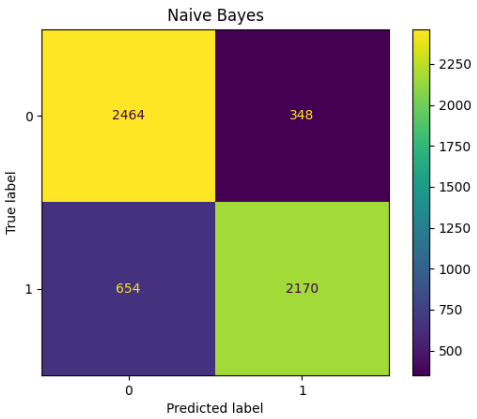
Si nota il calo di qualità quando il valore è pari, il miglior valore individuato è 11

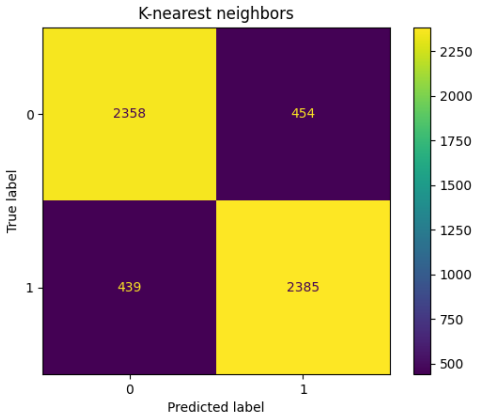
Valutazione

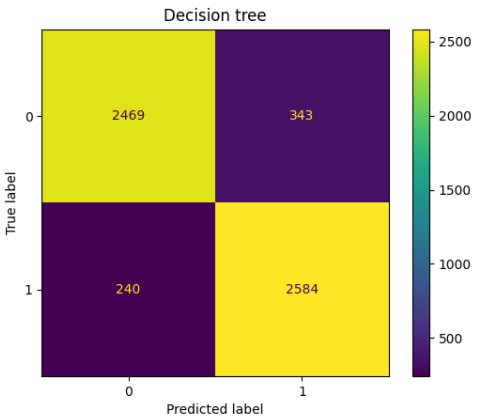












Come si verifica dai grafi, lo score di decision tree è fortemente migliorato, quasi arrivando a raggiungere randomForest, un simile miglioramento si può riscontrare in KNN, specialmente nelle matrici di confusione il numero di falsi positivi e negativi nel random forest, particolarmente importanti in ambito medico.

3. Reti Neurali

Sommario

In questa sezione esaminiamo l'implementazione di una rete neurale per la classificazione dei pazienti con sospetto diabete utilizzando la libreria il MLPClassifier di Scikit-learn.

La rete neurale è composta da più strati, quali un livello di input, uno o più strati nascosti e un livello di output. Ogni elemento in uno strato è connesso a tutti quelli dello strato successivo, e queste connessioni sono pesate, consentendo alla rete di apprendere modelli complessi nei dati. Durante il processo di addestramento, il MLP utilizza il backpropagation per ottimizzare i pesi, minimizzando la differenza tra le previsioni del modello e le classificazioni effettive.

Strumenti utilizzati

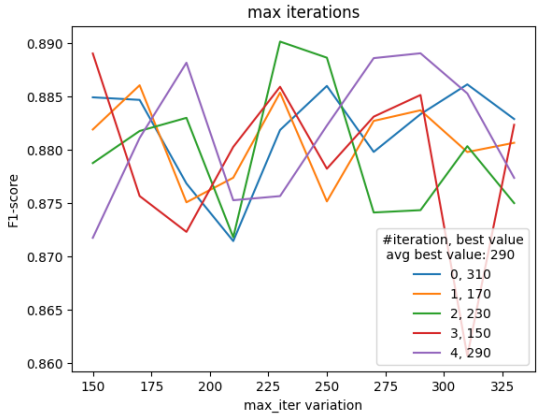
```
import pandas as pd
from imblearn.under_sampling import RandomUnderSampler
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,f1_score
```

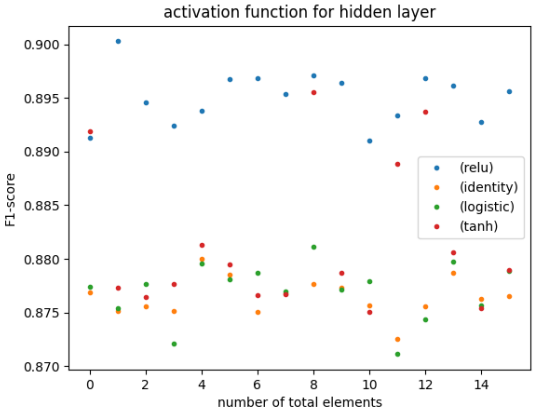
Decisioni di Progetto

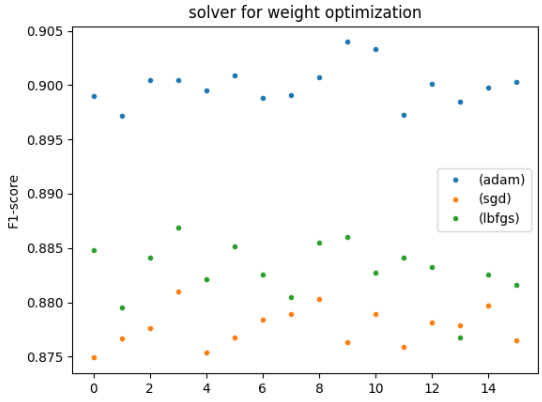
Testando inizialmente il modello senza variarne i parametri otteniamo già risultati promettenti:

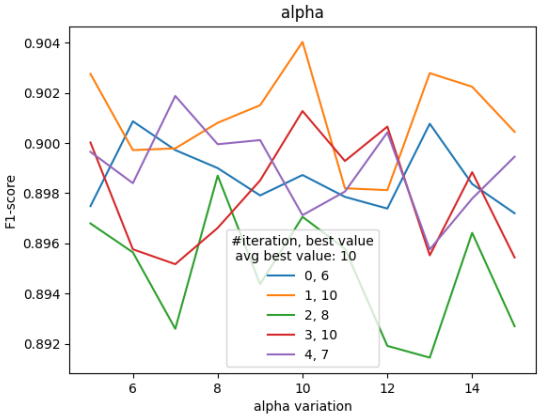
	precision	recall	f1-score	support
0	0.90	0.90	0.90	2812
1	0.90	0.90	0.90	2824
accuracy			0.90	5636
macro avg	0.90	0.90	0.90	5636
weighted avg	0.90	0.90	0.90	5636

Veniamo comunque notificati che lo Stochastic Optimizer abbia raggiunto il suo limite prima che l'ottimizzazione convergesse, andiamo quindi ad osservare i parametri del modello per capire se e di quanto possiamo migliorare il risultato:









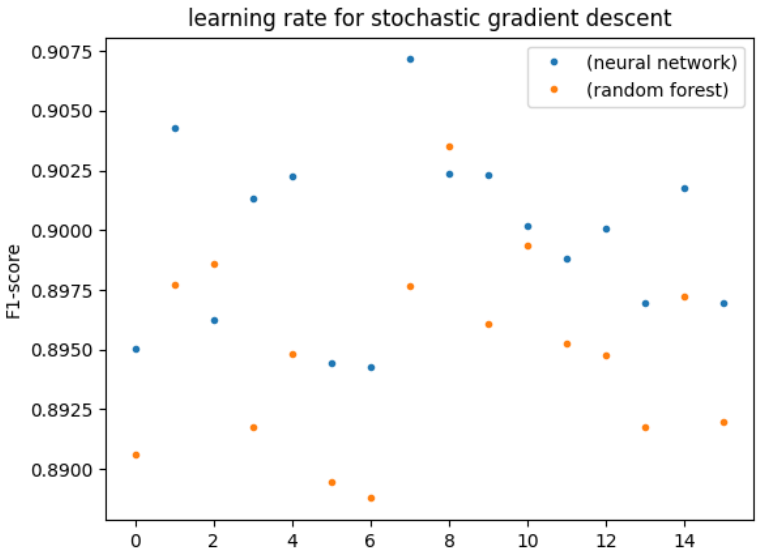
Max_iterations non ha dato risultati concludenti, ma aumentare il numero di iterazioni diminuisce la probabilità che l'ottimizzazione raggiunga il limite prima di convergere, anche se aumenta leggermente il tempo di esecuzione.

Activation indica la funzione di attivazione tra i layer nascosti (come l'output di un layer venga passato al successivo), la funzione relu [f(x) = max (0, x)] da risultati drasticamente migliori delle alternative.

Solver: algoritmo per l'ottimizzazione dei pesi, adam (ottimizzatore basato su sgd da Kingma, Diederik e Jimmy Ba) mostra risultati chiaramente superiori, si annota che secondo la documentazione, mentre adam sia più adatto a grandi dataset, lbfsg invece funzioni meglio con dataset di più piccole dimensioni, in quanto converge meglio e più velocemente.

Alpha: parametro per il controllo dei pesi, aumentarlo può ridurre la varianza mentre ridurlo può diminuire il bias, in questo caso non sembra aver avuto effetto ed il valore migliore rimane 0,0001 (i valori vengono moltiplicati per 0,00001)

Valutazione



Complessivamente, MLPClassifier ha risultati leggermente migliori rispetto alla configurazione migliore di Random Forest

4. Reti Bayesiane

Sommario

Le reti bayesiane sono uno strumento utile per la modellazione delle relazioni tra variabili per poi eseguire inferenze sulle probabilità condizionali ottenute, in questa sezione proviamo a crearne una per stabilire la probabilità che ad un paziente venga diagnosticato il diabete.

Strumenti utilizzati

```
import pandas as pd
from pgmpy.estimators import PC, ExpectationMaximization
from pgmpy.models.BayesianNetwork import BayesianNetwork
from pgmpy.inference import VariableElimination
```

Decisioni di Progetto

```
model = BayesianNetwork([('age','hypertension'),('age','heart_disease'),('age','smoking_history'),('age','bmi'),
                         ('age','hbA1c'),('age','glucose'),('hypertension','heart_disease'),('bmi','hypertension'),
                         ('hypertension','diabetes'),('smoking_history','hypertension'),
                         ('smoking_history','heart_disease'),('bmi','heart_disease'),('bmi','hbA1c'),
                         ('bmi','glucose'),('glucose','hbA1c'),('hbA1c','diabetes'),('glucose','diabetes')])
```

Un lato negativo delle reti bayesiane è la necessità del fornire una struttura, qui abbiamo provato a crearne una basandoci sulle relazioni scoperte nella prima sezione, l'ideale sarebbe chiedere ad un esperto di fornirla in anticipo.

In alternativa è possibile "apprendere la struttura" tramite un algoritmo che testi l'indipendenza condizionata tra le variabili del dataset:

```
est = PC(data = df)
est = est.estimate()
model = BayesianNetwork(est)
return model
```

Lo schema risultate è un grafico diretto aciclico (DAG) i cui archi partono dalle variabili parents() e raggiungono le variabili le cui probabilità ne influenzano

A questo punto alleniamo il modello utilizzando il dataset:

```
model.fit(data=data, estimator=ExpectationMaximization
```

Valutazione

Il modello finale ci permette di fare inferenze inserendo dati di potenziali pazienti e ricevere le probabilità che essi abbiano il diabete:

```
inferenza = VariableElimination(model)
print(inferenza.query(['diabetes'], evidence = {'age':80,'hypertension':0,'heart_disease':0,'bmi':29.16,
                                                'smoking_history':1,'hbA1c':8.8,'glucose':140}))
```

```
+--------------+------------------+
| diabetes     |   phi(diabetes) |
+==============+==================+
| diabetes(0) |           0.0000 |
+--------------+------------------+
| diabetes(1) |           1.0000 |
+--------------+------------------+
```

è anche possibile utilizzare dati parziali:

```
print(inferenza.query(['diabetes'], evidence = {'hypertension':0,'heart_disease':0,
                                                'hbA1c':6.1}))
```

```
+--------------+------------------+
| diabetes     |   phi(diabetes) |
+==============+==================+
| diabetes(0) |           0.7174 |
+--------------+------------------+
| diabetes(1) |           0.2826 |
+--------------+------------------+
```

Complessivamente, nonostante il carico di elaborazione elevato, le reti bayesiane possono essere un forte strumento in ambito clinico per la prevenzione e l'analisi dei rischi nei pazienti.

Riferimenti Bibliografici

- pagina kaggle del dataset
- pagina sci-kit learn (tutti i modelli ad eccezione di reti bayesiane)
- documentazione Bayesian network pgmpy