Studentessa: Cuzzo Francesca

Matricola: 744406

E-Mail istituzionale: f.cuzzo@studenti.uniba.it

URL Repository: https://github.com/phrancescac/ICON

CASO DI STUDIO:

SISTEMA DI PREDIZIONE DEL DIABETE NELLE DONNE

AA 2023-2024

INDICE

INTRODUZIONE

Il diabete è una patologia cronica caratterizzata dall'elevata presenza di zuccheri nel sangue(iperglicemia). La sua manifestazione può derivare dalla carenza di produzione di insulina o dalla sua incapacità nella corretta regolazione del livello di glucosio nel sangue.

Questa condizione, in costante aumento, colpisce prevalentemente le donne, rappresentando per loro un peso significativo rispetto agli uomini. Il diabete assume una considerevole rilevanza nelle donne soprattutto durante l'età fertile e in occasione della gravidanza.

INFORMAZIONI SU DATASET E STRUMENTI UTILIZZATI

Il progetto è stato realizzato con l'ausilio del linguaggio Python e come ambiente di sviluppo è stato utilizzato Visual Studio Code, utilizzando librerie come:

- **Pandas:** per gestire i data frame
- **Matplotlib**, **Seaborn**: per gestire e creare i grafici
- **Numpy**: per eseguire operazioni su determinati tipi di dati
- **Scikit-learn**: per eseguire la classificazione e utilizzare le metriche di valutazione

Il presente progetto si basa su un dataset proveniente dall'Istituto Nazionale del Diabete e delle Malattie Digestive e Renali, presente sul sito Kaggle: *Diabetes Prediction Dataset (kaggle.com)* .

Questo caso di studio mira a fornire una previsione diagnostica sulla presenza del diabete attraverso specifiche misurazioni incluse nel set di dati. È importante sottolineare che i pazienti selezionati per questo studio sono tutte donne di origine indiana, con un'età minima di 21 anni, introducendo così una dimensione culturale e demografica specifica nell'analisi.

ANALISI DEI DATI CONTENUTI NEL DATASET

Il dataset (diabetes.csv) utilizzato è composto da 768 campioni e 9 features:

- **'Pregnancies'** → esprime il numero di gravidanze
- '**Glucose'** → esprime il livello di glucosio nel sangue a 2 ore prima dal test
- **'Blood_Pressure'** → esprime la misurazione della pressione arteriosa
- **'Skin_Thickness'** → esprime lo spessore della pelle
- '**Insulin'** → esprime il livello di insulina nel sangue dopo 2 ore

- **‘BMI’** → esprime l’indice di massa corporea
- **‘Diabetes_Pedigree_Function’** → esprime la probabilità di ereditare il diabete
- **‘Age’** → esprime l’età
- **‘Outcome’** → valore target (1: è diabetica, 0: non è diabetica)

ELENCO ARGOMENTI DI INTERESSE

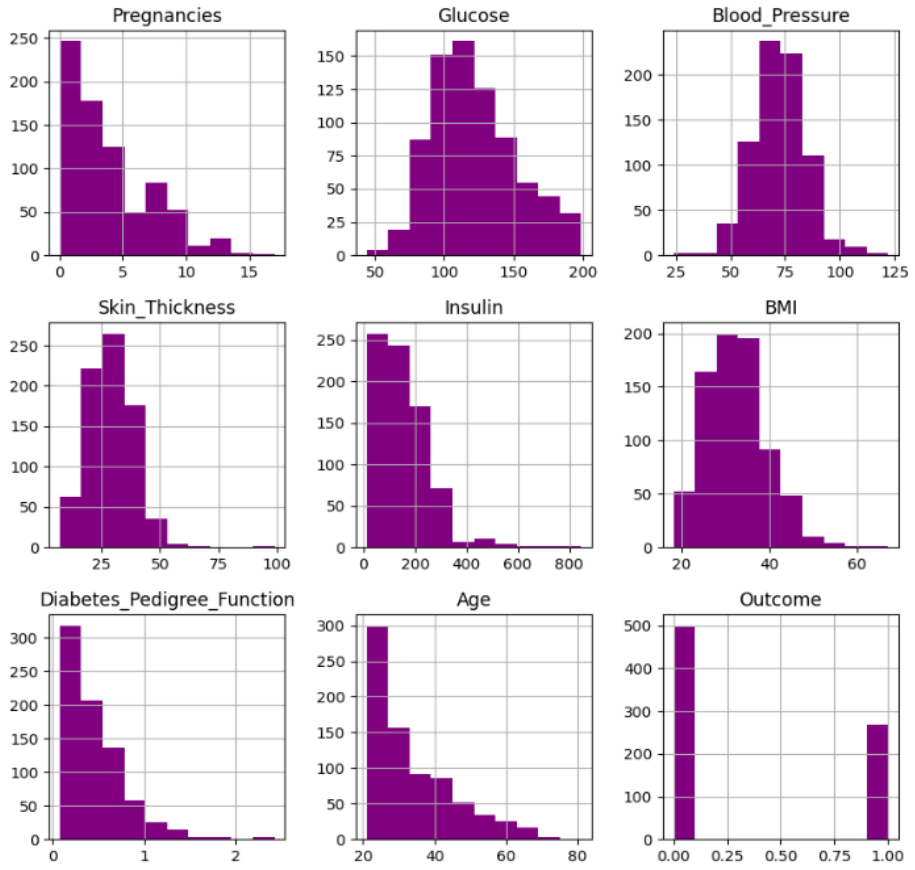
Gli argomenti affrontati per la creazione del progetto includono:

- Apprendimento supervisionato: tecnica in cui il modello viene addestrato su un vasto dataset che comprende informazioni sui pazienti affetti e non dal diabete e il loro stato di salute per fare previsioni su nuovi pazienti.
    - Apprendimento supervisionato con iperparametri: Questo approccio è utilizzato per ottimizzare i parametri del modello al fine di migliorare le prestazioni nella predizione del diabete. Tale ottimizzazione avviene attraverso la ricerca e la regolazione dei parametri del modello al fine di massimizzare l'accuratezza delle previsioni.
- Knowledge Base: creazione di una base di conoscenza per condurre analisi delle informazioni presenti nel dataset, deducendo se il paziente potrebbe aver sviluppato il diabete, utilizzando i parametri forniti.

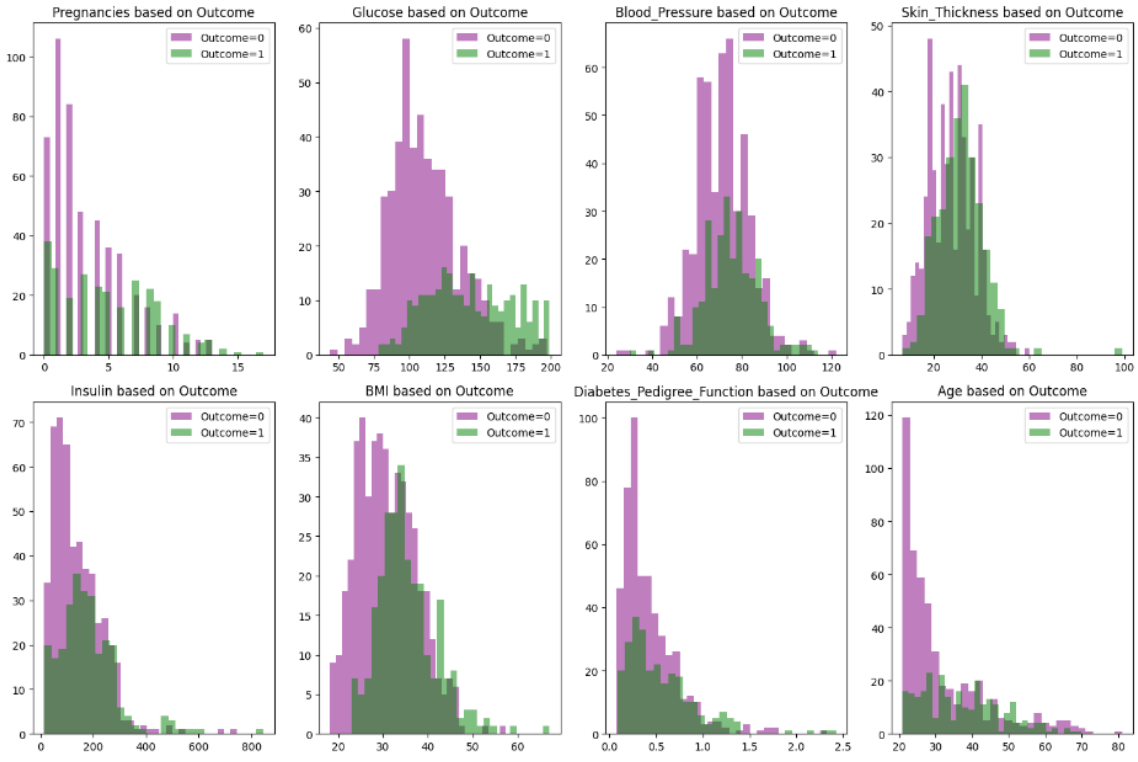
Decisioni progettuali e Strumenti utilizzati

Per una miglior predizione, sono state utilizzate tecniche di analisi esplorativa dei dati (EDA) e modelli predittivi, ripulendo il dataset eliminando eventuali duplicati e i valori nulli presenti all’interno, poiché è stato necessario eliminare tali valori soprattutto per quanto riguarda l’insulina o la pressione sanguigna che non possono essere pari a 0.

Per visualizzare la frequenza con cui determinati valori o intervalli di valori compaiono nel dataset, sono stati utilizzati grafici che mostrano come essi sono distribuiti rispetto alle relative caratteristiche.



Successivamente, sono stati rappresentati graficamente i dati più rilevanti in base al valore target nel dataset, per comprendere meglio i valori che influiscono maggiormente e migliorare così le predizioni.



Dai grafici risulta che il glucosio (Glucose), BMI, e pressione sanguigna (Blood_pressure), sono i valori che maggiormente influiscono sulla predizione del diabete.

KNOWLEDGE BASE

Sommario

Una base di conoscenza (KB) in logica del primo ordine è un insieme di proposizioni, note come assiomi, che sono considerate vere senza necessità di dimostrazione. La KB è fondamentale per rappresentare la conoscenza di un particolare dominio all'interno di un sistema o di una macchina.

La creazione di una base di conoscenza segue una serie di passaggi:

1. **Definizione del dominio:** Inizialmente, è necessario definire il dominio che si desidera rappresentare. In questo caso, la KB ha come dominio il mondo reale e in particolare, condizioni fisiche reali.
2. **Identificazione delle proposizioni atomiche:** Successivamente, vengono identificate le proposizioni atomiche che sono utili per rappresentare il dominio scelto. Queste proposizioni costituiscono gli elementi di base su cui viene costruita la base di conoscenza.
3. **Assiomatizzazione del dominio:** Qui vengono definiti gli assiomi, cioè le proposizioni che saranno considerate vere nell'interpretazione del dominio. Gli assiomi rappresentano le verità fondamentali del dominio e costituiscono la base su cui è fondato il ragionamento logico all'interno del sistema.
4. **Interrogazione del sistema:** Una volta definita la base di conoscenza, è possibile porre domande o query al sistema. Il sistema determina se specifiche proposizioni sono conseguenze logiche della base di conoscenza, verificando se sono vere in tutti i modelli della KB. In altre parole, il sistema valuta se le proposizioni richieste possono essere dedotte logicamente dagli assiomi della KB.

Il sistema, a differenza del progettista, non possiede una comprensione intrinseca del significato dei simboli utilizzati nella base di conoscenza. Il sistema si fonda esclusivamente sugli assiomi definiti e sulla logica inferenziale per determinare se una particolare proposizione è una conseguenza logica della base di conoscenza. È compito del progettista, basandosi sull'interpretazione

del dominio, valutare se il risultato prodotto dal sistema è valido e coerente con la conoscenza del dominio.

Decisioni progettuali e Strumenti utilizzati

Per implementare una base di conoscenza in logica del primo ordine, si è fatto uso del linguaggio di programmazione Prolog con l'ausilio della libreria pyswip. Le caratteristiche selezionate sono state rappresentate come fatti nella base di conoscenza, con la specifica dei loro domini.

- È stata creata una regola Prolog che determina se una paziente è diabetica attraverso una somma di valori. Per determinare la condizione della paziente, sono stati utilizzati parametri medici che sono fondamentali in persone affette da diabete. I parametri analizzati all'interno della regola sono: il numero di gravidanze avute dalla paziente, (anche se non è un elemento fondamentale ai fini dell'analisi del diabete), il livello minimo di glucosio presente nel sangue, il valore minimo della pressione sanguigna, il valore dello spessore della pelle, il valore minimo di insulina, il valore di BMI, la percentuale minima di probabilità di aver ereditato il diabete e infine l'età della paziente, fattore abbastanza importante in quanto si è verificato che la maggior parte delle pazienti hanno il diabete in età avanzata.

  A ciascun parametro vengono assegnati due numeri 1 e 0, dove il primo rappresenta l'esistenza di quel fattore, il secondo rappresenta il contrario. La valutazione finale del sistema consiste nel sommare le condizioni che hanno valore 1 e se la somma è maggiore o uguale ad un minimo di 5, allora comparirà il risultato, rappresentato dalla variabile 'Diabetica', come positivo o negativo in caso contrario.

```
% Regola utile a verificare se la paziente è diabetica
paziente_diabetica(Gravidanze, Glucosio, PressioneSanguigna, SpessorePelle, Insulina, BMI, ProbabilitaDiabete, Eta, Diabetica) :-
    % Condizioni per essere diabetica
    (
        (Gravidanze >= 0 ->  Cond1=1; Cond1=0),% Numero di gravidanze avute dalla paziente, non è un valore necessario al fine della valutazione del diabete
        (Glucosio >= 126 ->  Cond2=1; Cond2=0),% Livello di glucosio nel sangue
        (PressioneSanguigna >= 80 -> Cond3=1; Cond3=0),% Valore della pressione sanguigna
        (SpessorePelle >= 20 ->  Cond4=1; Cond4=0),% Valore di spessore della pelle
        (Insulina >= 25 ->  Cond5=1; Cond5=0),% Livello di insulina
        (BMI >= 30 -> Cond6=1; Cond6=0), % Si consiglia un BMI più alto per indicare il rischio di diabete
        (ProbabilitaDiabete > 0.5 -> Cond7=1; Cond7=0), % Aumento della probabilità di diabete
        (Eta >= 20 -> Cond8=1; Cond8=0) % Età della paziente
    ),

    % Somma delle condizioni
    Somma is Cond1 + Cond2 + Cond3 + Cond4 + Cond5 + Cond6 + Cond7 + Cond8,

    % Se il numero di condizioni soddisfatte supera una soglia, la paziente è considerata diabetica
    (Somma >= 5 -> Diabetica = si; Diabetica = no).
```

Esempio di query con valori di una paziente registrati nel CSV:

```
2 ?- paziente_diabetica(2,100,66,20,90,32.9,0.867,28,Diabetica).
Diabetica = si.
```

- Poiché l'età è un fattore fondamentale per la rilevazione del diabete, è stata creata un'ulteriore regola Prolog che calcola l'età media delle pazienti affette all'interno del dataset. All'interno della regola c'è la funzione 'findall' che serve ad estrarre tutte le età delle pazienti diabetiche e memorizzarle all'interno di una lista che viene chiamata 'ListaEtà'. Viene poi calcolata la lunghezza di questa lista attraverso la funzione 'length' per determinare il numero di pazienti diabetiche.

  La somma di tutte le età viene calcolata e memorizzata all'interno di 'sum_list' e viene calcolata la media delle età dividendo la somma per il numero di persone affette.

```
% Regola per calcolare letà media delle pazienti che hanno il diabete
eta_media(EtàMedia) :-
    findall(Age, (age(Age), Age > 0), ListeEtà),% Estrazione delle età delle pazienti diabetiche
    length(ListeEtà, NumeroPersone),% Conta il numero di persone con diabete
    sum_list(ListeEtà, SommaEtà),% Somma delle età
    EtàMedia is SommaEtà/NumeroPersone.% Calcolo delletà media
```

APPRENDIMENTO SUPERVISIONATO

Sommario

Il presente progetto ha lo scopo di analizzare le informazioni delle pazienti raccolte all'interno del dataset fornito, per valutare e diagnosticare se sono affette da diabete. La valutazione viene indicata della feature 'Outcome', variabile categorica che assume valore 1 se la paziente risulta diabetica e 0 in caso contrario.

Per eseguire la predizione sulla condizione delle pazienti, si sono utilizzati diversi modelli di classificazione. Tenendo conto della quantità limitata dei dati presenti nel dataset, si è preferito scegliere modelli con complessità più semplice rispetto alla regressione lineare o alle reti neurali che richiedono una miglior precisione dovuta ad un dataset più fornito.

I modelli scelti sono: K Nearest Neighbors (KNN), Decisional Tree, Random Forest, Ada Boost, utilizzando la libreria Scikit Learn importando le relative classi e infine il modello XGBoost, utilizzando la libreria XGBoost.

Decisioni progettuali e Strumenti utilizzati

All'interno del file 'classifier_train.py' sono stati addestrati e testati i diversi modelli senza l'utilizzo di particolari parametri, solo per prendere visione dei risultati iniziali.

I risultati ottenuti sono i seguenti:

```
K Nearest Neighbors Classification:
              precision    recall  f1-score   support

           0       0.77      0.72      0.74        50
           1       0.53      0.59      0.56        27

    accuracy                           0.68        77
   macro avg       0.65      0.66      0.65        77
weighted avg       0.68      0.68      0.68        77

F2 Score for KNN Classifier: 0.5797101449275363

Decision Tree Classification:
              precision    recall  f1-score   support

           0       0.69      0.70      0.69        50
           1       0.42      0.41      0.42        27

    accuracy                           0.60        77
   macro avg       0.55      0.55      0.55        77
weighted avg       0.59      0.60      0.60        77

F2 Score for Decision Tree Classifier: 0.41044776119402987
```

```
Random Forest Classification:
              precision    recall  f1-score   support

           0       0.79      0.76      0.78        50
           1       0.59      0.63      0.61        27

    accuracy                           0.71        77
   macro avg       0.69      0.69      0.69        77
weighted avg       0.72      0.71      0.72        77

F2 Score for Random Forest Classifier: 0.6204379562043796

AdaBoost Classification:
              precision    recall  f1-score   support

           0       0.75      0.78      0.76        50
           1       0.56      0.52      0.54        27

    accuracy                           0.69        77
   macro avg       0.66      0.65      0.65        77
weighted avg       0.68      0.69      0.69        77

F2 Score for AdaBoost Classifier: 0.5263157894736842

XGBoost Classification:
              precision    recall  f1-score   support

           0       0.80      0.74      0.77        50
           1       0.58      0.67      0.62        27

    accuracy                           0.71        77
   macro avg       0.69      0.70      0.70        77
weighted avg       0.73      0.71      0.72        77

F2 Score for XGBoost Classifier: 0.6474820143884892
```

Sono state eseguite ulteriori valutazioni delle prestazioni del modello, incluso un 'classification report' che fornisce informazioni come precision e recall.

La precision è una metrica fondamentale per ridurre al minimo i falsi positivi, che si verificano quando il modello erroneamente classifica un caso come

positivo quando in realtà non lo è. Una precision elevata indica che i casi classificati come positivi sono molto probabili di essere effettivamente positivi, riducendo così al minimo i falsi positivi.

La recall è una metrica essenziale per ridurre al minimo i falsi negativi, che si verificano quando il modello classifica erroneamente un caso come negativo quando in realtà è positivo. Una recall elevata indica che il modello è in grado di individuare la maggior parte dei casi positivi senza trascurare troppi falsi negativi.

Queste misure sono particolarmente importanti nella valutazione delle prestazioni del modello, specialmente in problemi di classificazione dove una classe è significativamente più rara dell'altra. Soprattutto in contesti medici è importante ridurre al minimo sia i falsi positivi, che causano preoccupazione nei pazienti, sia i falsi negativi, che causano problemi se si ignora la complessità della situazione.

L'ottimizzazione di queste metriche può essere complessa poiché spesso c'è un compromesso tra precisione e recall. La scelta dipenderà dalle specifiche esigenze del problema e da quanto peso si desidera dare ai falsi positivi rispetto ai falsi negativi.

Dall'analisi iniziale, sembra che il modello ‘XGBoost’ abbia le prestazioni migliori in termini di F2 Score e Precision, seguito da ‘Random Forest’. Successivamente c’è il modello ‘K-Nearest Neighbors’, dopo il modello ‘AdaBoost’, e infine c’è il ‘Decision Tree’ risultando essere il modello con le prestazioni più basse.

Successivamente, è stata condotta un'analisi più approfondita per identificare i parametri ottimali dei modelli KNN, AdaBoost e Decision Tree al fine di migliorarne le prestazioni, essendo i tre modelli con accuracy più basse.

Di seguito i modelli analizzati nel dettaglio:

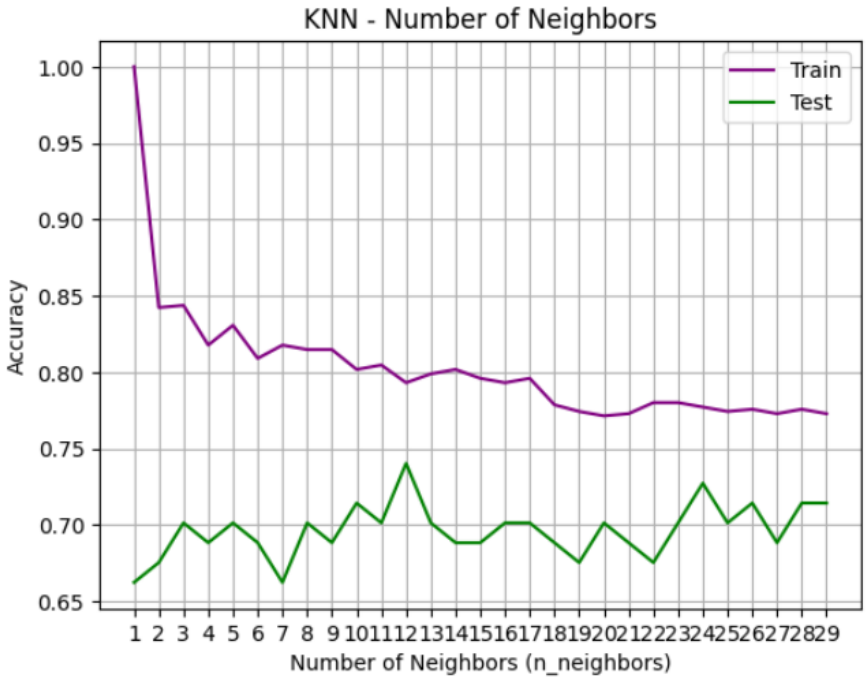
K-NEAREST NEIGHBORS

Il modello **KNN (K-Nearest Neighbors**) esamina le classi dei punti dati che circondano un punto dati target, con l'obiettivo di prevedere a quale classe appartiene il punto più vicino. Per ottenere una ottimizzazione del modello, è fondamentale standardizzare i dati in modo che tutte le variabili abbiano lo stesso peso durante il processo decisionale. L'accuratezza complessiva del

modello KNN tende ad aumentare notevolmente dopo la standardizzazione dei dati.

Tra gli iperparametri considerati e utilizzati all'interno dell'algoritmo KNN c'è 'n_neighbors'.

L'iperparametro **'n_neighbors'** è il più importante del KNN poichè rappresenta il numero di vicini più vicini che saranno presi in considerazione durante la classificazione.



Sono stati scelti 30 numeri di vicini più vicini, e dal grafico è visibile come l'accuratezza dei valori di test rimane più o meno costante presentando alti e bassi, raggiungendo però il suo picco di miglioramento al vicino numero 12. Invece l'accuracy dei valori di train tende a diminuire velocemente fino al vicino numero 12 per poi stabilizzarsi e rimanere costante.

Dunque, i risultati ottenuti da questa analisi più approfondita sono i seguenti:

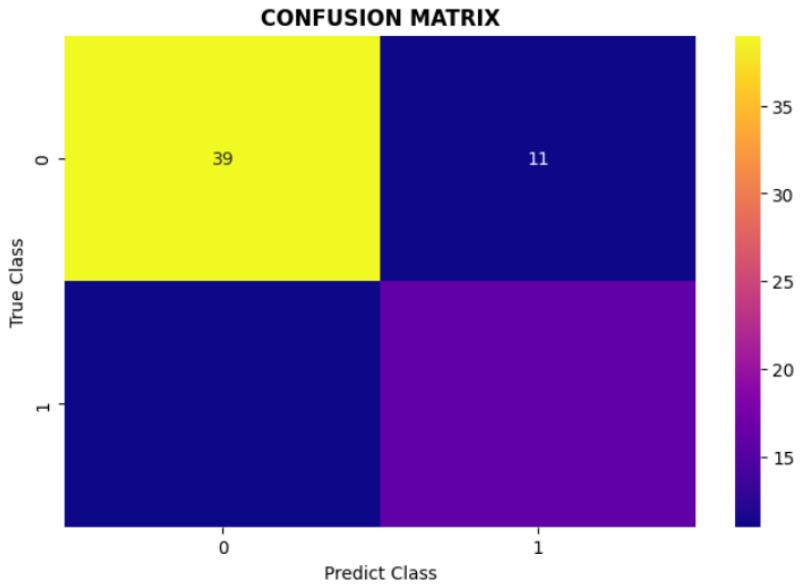
```
Cross-validation accuracy:  [0.73381295 0.76811594 0.68115942 0.73913043 0.76086957]

Mean accuracy:  0.7366176623918257

Standard deviation of accuracy:  0.030561719410100923
Accuracy with k neighbors=12:  0.7402597402597403
Classification report:
              precision    recall  f1-score   support

           0       0.78      0.84      0.81        50
           1       0.65      0.56      0.60        27

    accuracy                           0.74        77
   macro avg       0.71      0.70      0.70        77
weighted avg       0.73      0.74      0.73        77
```



DECISION TREE

Il modello **Decision Tree** ha la struttura di un albero composto da nodi e archi, dove i nodi rappresentano le domande o le condizioni sui dati e gli archi rappresentano le possibili risposte a tali domande. Ogni nodo interno dell'albero rappresenta una caratteristica dei dati, mentre le foglie rappresentano le decisioni o le previsioni.

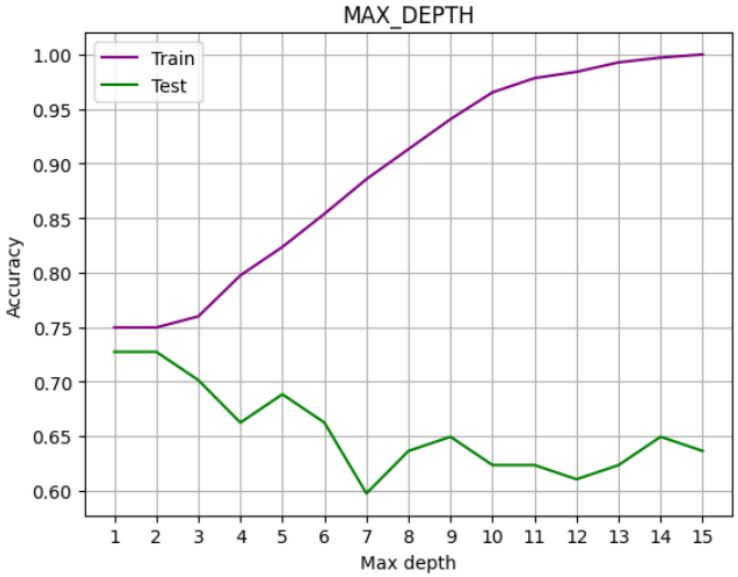
Durante il processo di addestramento, l'algoritmo cerca di suddividere i dati in gruppi omogenei, minimizzando la disomogeneità all'interno di ciascun gruppo. Questo processo viene ripetuto in modo ricorsivo fino a quando non si raggiunge

una condizione di arresto, come ad esempio la profondità massima dell'albero o un numero minimo di campioni in ciascun nodo foglia.

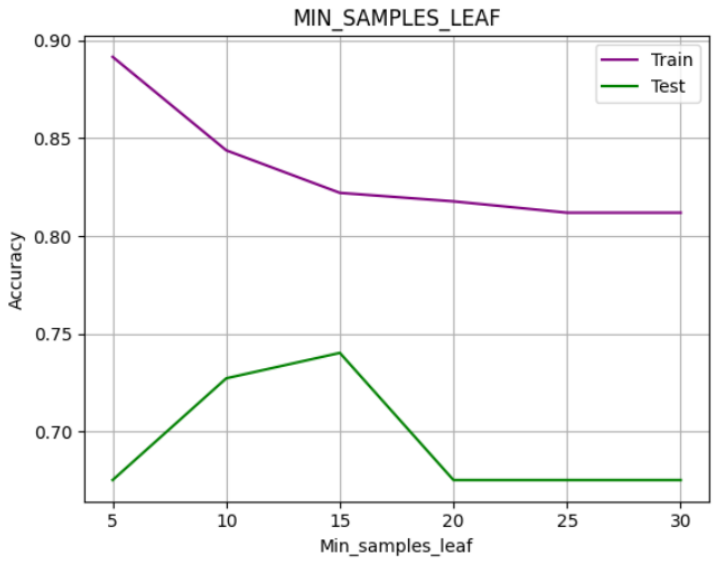
Tra gli iperparametri all'interno dell'algoritmo di Decision Tree sono presenti:

- **max_depth**: iperparametro che rappresenta la massima profondità dell'albero. Determina quante volte l'albero verrà suddiviso in livelli. Una profondità elevata può portare ad un sovradattamento (overfitting), diminuendo la generalizzazione del modello, al contrario una bassa profondità può portare ad un sottoadattamento (underfitting).
- **min_samples_leaf:** iperparametro che rappresenta il numero minimo di campioni che dovrebbero essere presenti nel nodo foglia. Se dopo la divisione il numero di campioni in uno dei nuovi rami diventa inferiore a questo valore, la divisione non viene effettuata.
- **splitter**: è la strategia per eseguire la selezione di un attributo per la divisione ("best" miglior attributo, "random" attributo casuale).

I risultati ottenuti per gli iperparametri max_depth e min_samples_leaf sono i seguenti:



Dal grafico del max_depth, l'accuratezza sui dati di test non supera l'accuratezza sui dati di addestramento, anzi tende a diminuire rispetto a quella sui dati di addestramento che tende ad aumentare.



Dal grafico del min_samples_leaf, si nota come il numero di elementi nel nodo foglia sui dati di test aumenta fino a 15 per poi diminuire e rimanere costante. L'accuratezza sui dati di addestramento tende invece a diminuire.

Un grande numero di valori nel nodo foglia rende il modello non in grado di riaddestrarsi e memorizzare il set di addestramento, come accade con un valore di 5. Dunque, è molto importante scegliere i giusti valori per il numero di valori nel nodo foglia, garantendo al modello di generalizzare correttamente i dati di addestramento al fine di ottenere buone prestazioni sui dati di test.

I valori finali ottenuti in questa analisi approfondita del modello sono i seguenti:

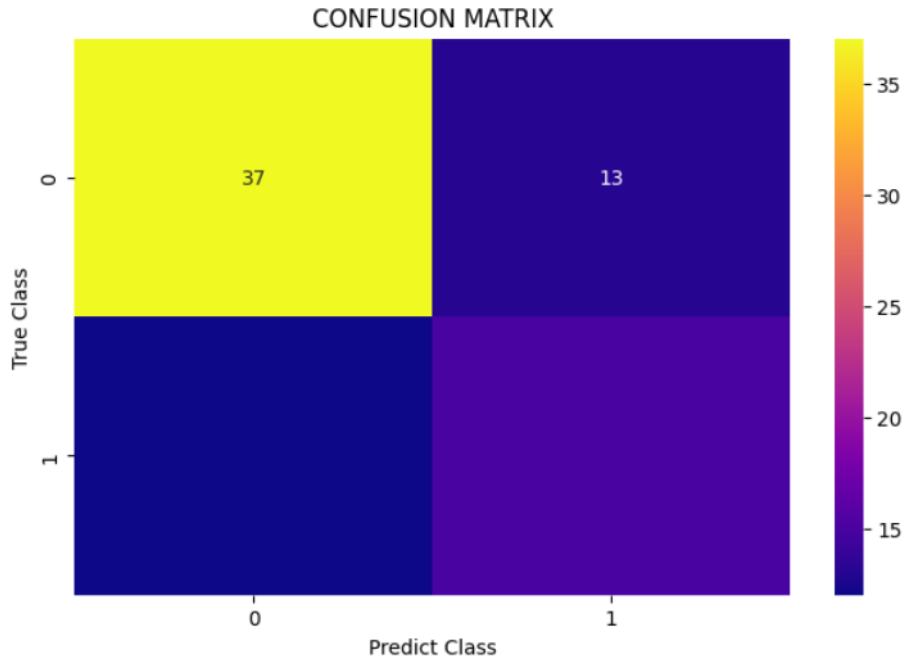
```
Cross-validation accuracy:  [0.77697842 0.79710145 0.72463768 0.76811594 0.78985507]

Mean accuracy:  0.7713377124387446

Standard accuracy:  0.025415558365285744
Accuracy on test set:  0.6753246753246753
Classification report:
              precision    recall  f1-score   support

           0       0.76      0.74      0.75        50
           1       0.54      0.56      0.55        27

    accuracy                           0.68        77
   macro avg       0.65      0.65      0.65        77
weighted avg       0.68      0.68      0.68        77
```

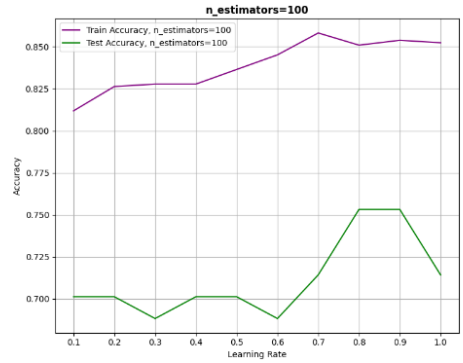


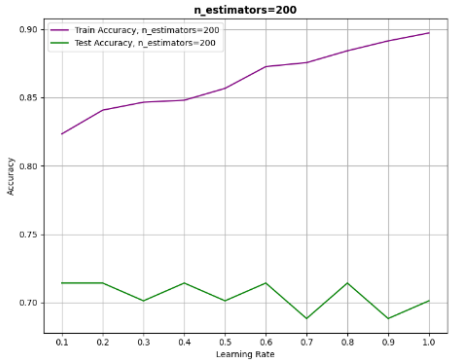
ADABOOST

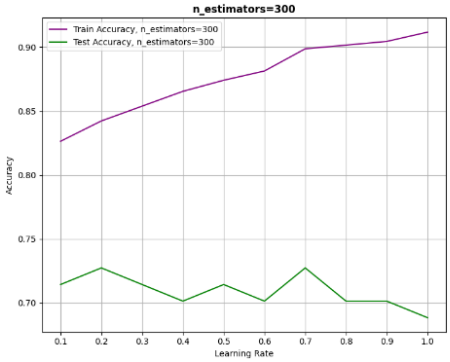
Il modello **AdaBoost** ha la capacità di trasformare una serie di classificatori deboli in un classificatore forte concentrandosi su esempi di addestramento più complessi. Questo processo iterativo di addestramento e pesatura degli esempi, consente all'algoritmo di correggere gli errori commessi dai classificatori deboli precedenti, rendendo il modello finale altamente accurato e robusto.

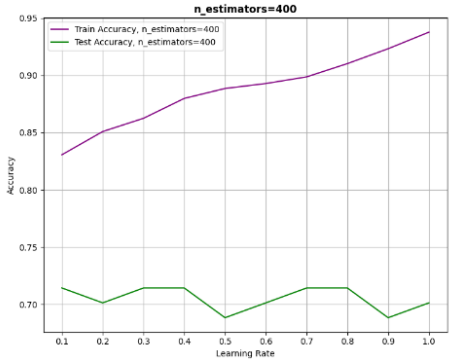
Tra gli iperparametri più importanti utilizzati nell'algoritmo di AdaBoost sono presenti:

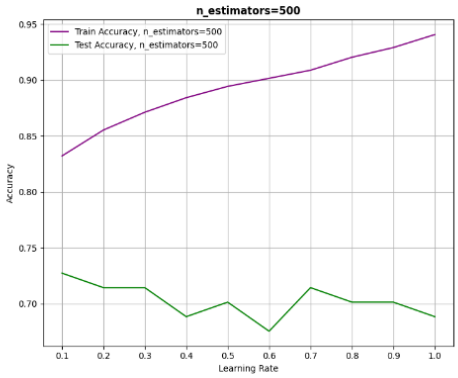
- **n_estimators:** è l'iperparametro principale dell'algoritmo. Rappresenta il numero di stimatori che verranno utilizzati per costruire il modello. La scelta di più stimatori porta ad un miglioramento di prestazioni.
- **learning_rate:** iperparametro che definisce la proporzione dei dati che verranno utilizzati per i dati di test. Nell'algoritmo sviluppato viene utilizzato solo il 10% dei dati per valutare le prestazioni del modello.

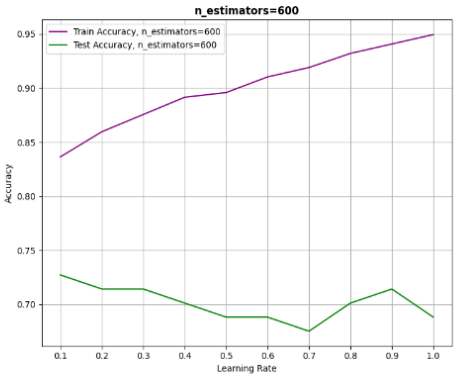

n_estimators=100
Train Accuracy, n_estimators=100
Test Accuracy, n_estimators=100
Accuracy
Learning Rate

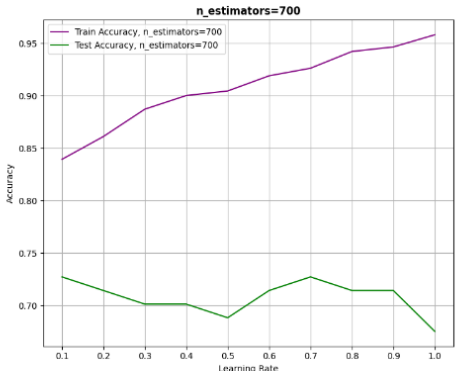

n_estimators=200
Train Accuracy, n_estimators=200
Test Accuracy, n_estimators=200
Accuracy
Learning Rate

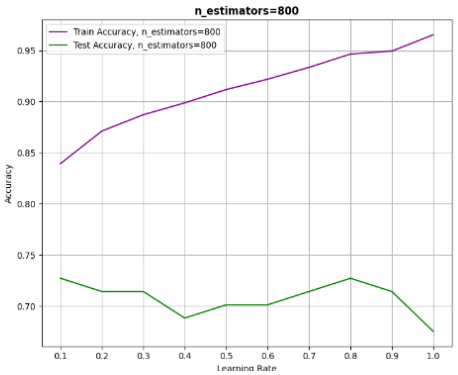

n_estimators=300
Train Accuracy, n_estimators=300
Test Accuracy, n_estimators=300
Accuracy
Learning Rate

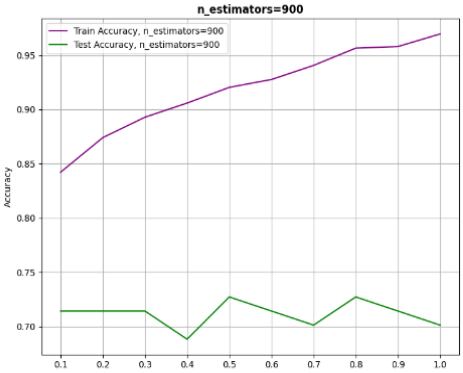

n_estimators=400
Train Accuracy, n_estimators=400
Test Accuracy, n_estimators=400
Accuracy
Learning Rate

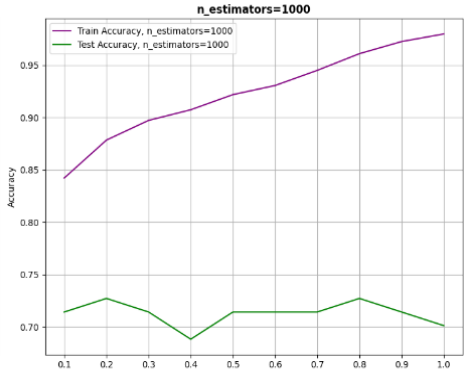

n_estimators=500
Train Accuracy, n_estimators=500
Test Accuracy, n_estimators=500
Accuracy
Learning Rate


n_estimators=600
Train Accuracy, n_estimators=600
Test Accuracy, n_estimators=600
Accuracy
Learning Rate


n_estimators=700
Train Accuracy, n_estimators=700
Test Accuracy, n_estimators=700
Accuracy
Learning Rate


n_estimators=800
Train Accuracy, n_estimators=800
Test Accuracy, n_estimators=800
Accuracy
Learning Rate


n_estimators=900
Train Accuracy, n_estimators=900
Test Accuracy, n_estimators=900
Accuracy
Learning Rate


n_estimators=1000
Train Accuracy, n_estimators=1000
Test Accuracy, n_estimators=1000
Accuracy
Learning Rate

I risultati finali ottenuti sono i seguenti:

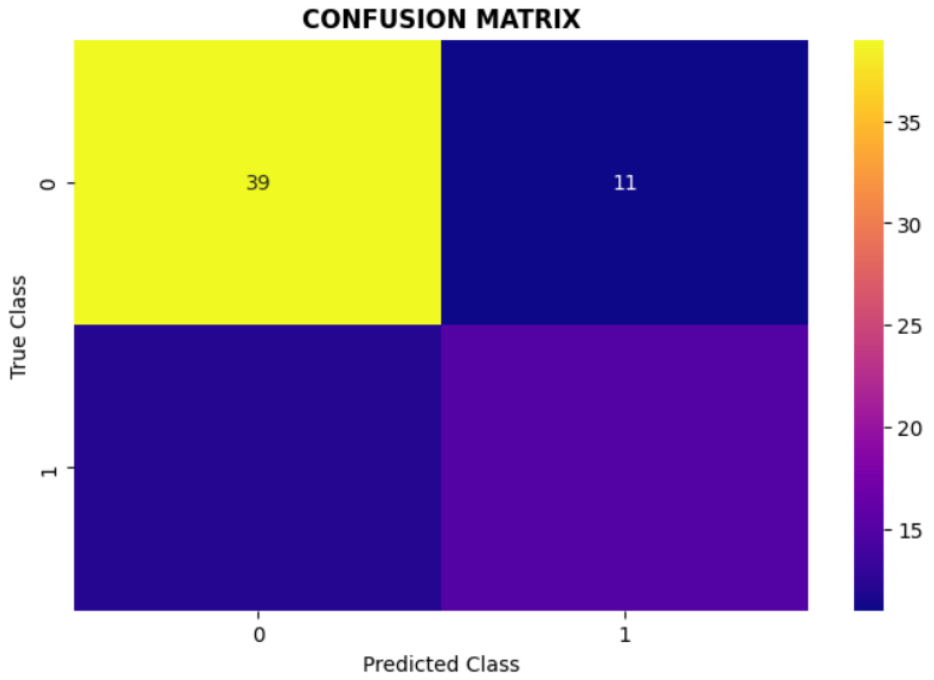
```
Cross-validation accuracy:  [0.77697842 0.79710145 0.73188406 0.76811594 0.78985507]

Mean accuracy:  0.7727869878010635

Standard accuracy:  0.022781347497838122
Accuracy on test set:  0.7012987012987013
Classification report:
              precision    recall  f1-score   support

           0       0.76      0.78      0.77        50
           1       0.58      0.56      0.57        27

    accuracy                           0.70        77
   macro avg       0.67      0.67      0.67        77
weighted avg       0.70      0.70      0.70        77
```



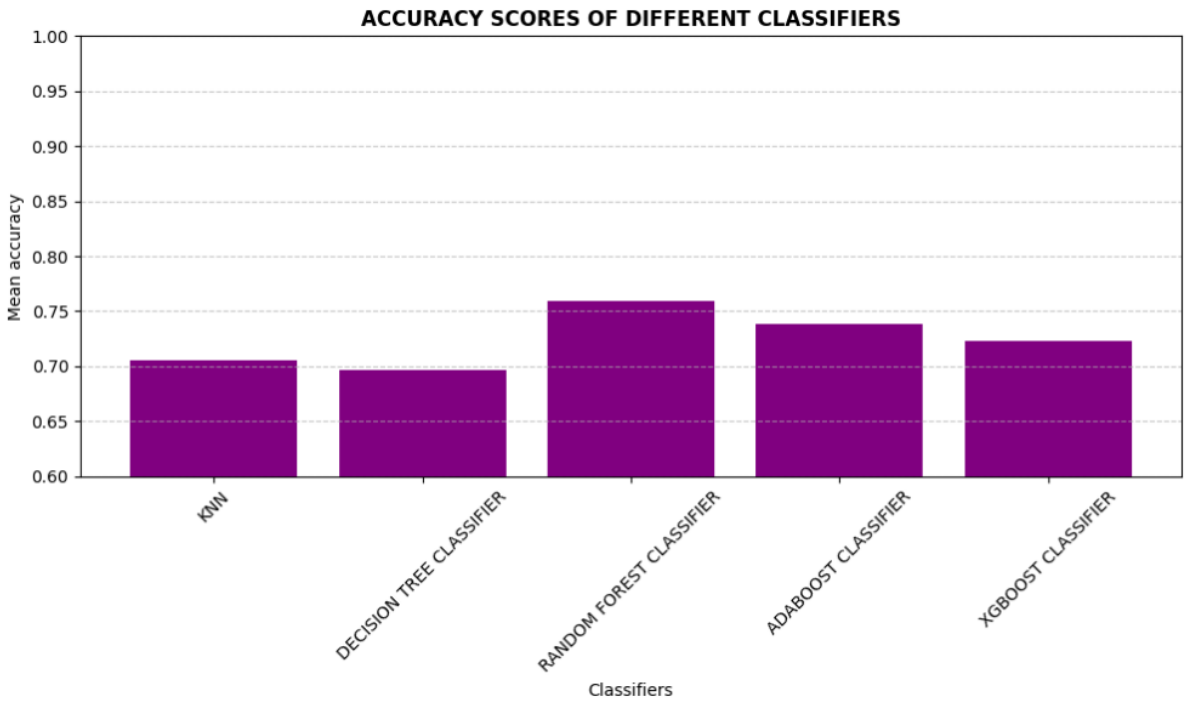
Valutazioni finali

Nonostante sia visibile come l'accuratezza sui dati di test sia molto più bassa rispetto a quella ottenuta nel modello di KNN, la 'mean accuracy' del modello di AdaBoost risulta essere più alta sia rispetto al modello di KNN sia rispetto al modello di Decision Tree.

Questo dimostra che il modello AdaBoost ha prestazioni migliori su diverse partizioni del set di addestramento e dunque, maggiore capacità di generalizzazione.

CONCLUSIONE

Per valutare meglio i modelli di classificazione, si è effettuato un confronto dei diversi algoritmi.



Dai risultati del grafico, è evidente come l'analisi iniziale non si è rivelata completamente accurata.

In effetti, il modello Random Forest risulta essere il migliore a livello di accuratezza, seguito dal modello AdaBoost che si è rivelato avere ottime prestazioni raggiungendo una accuracy di quasi 75%.

Successivamente c'è il modello XGBoost seguito dal modello KNN con una mean accuracy poco più alta del 70%. In conclusione, il modello Decision Tree si è confermato avere una bassa accuratezza rispetto agli altri modelli, e di conseguenza una minore capacità di generalizzazione.