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

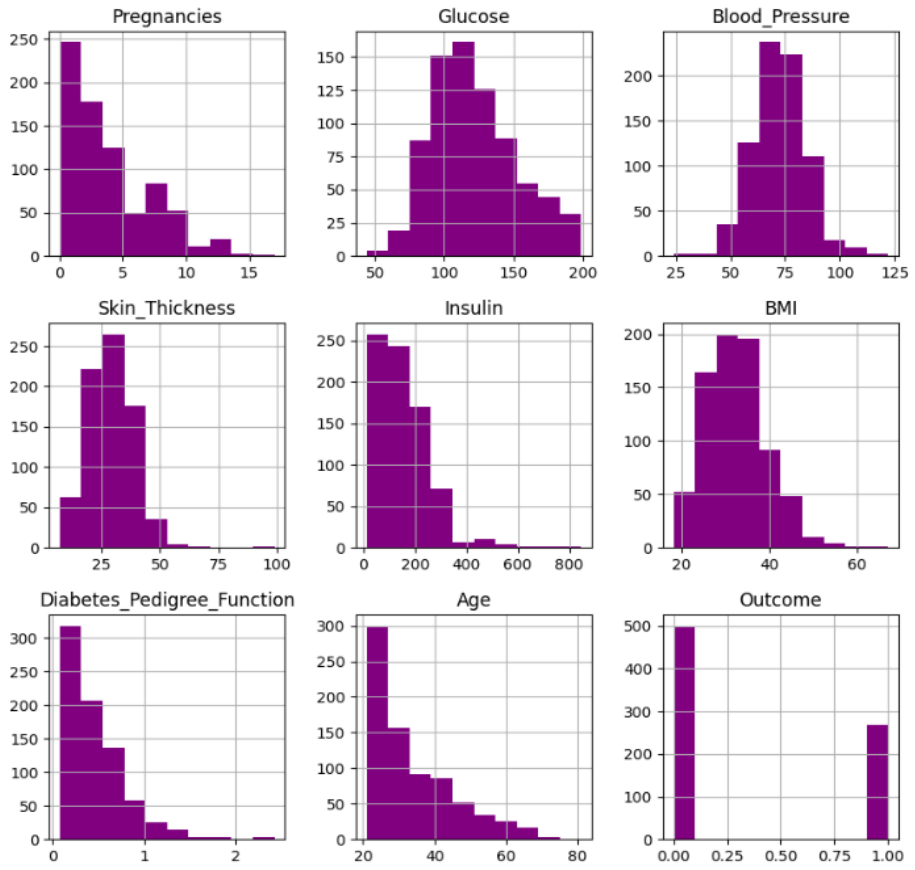
Gli argomenti affrontati per la creazione del progetto includono:

- 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.

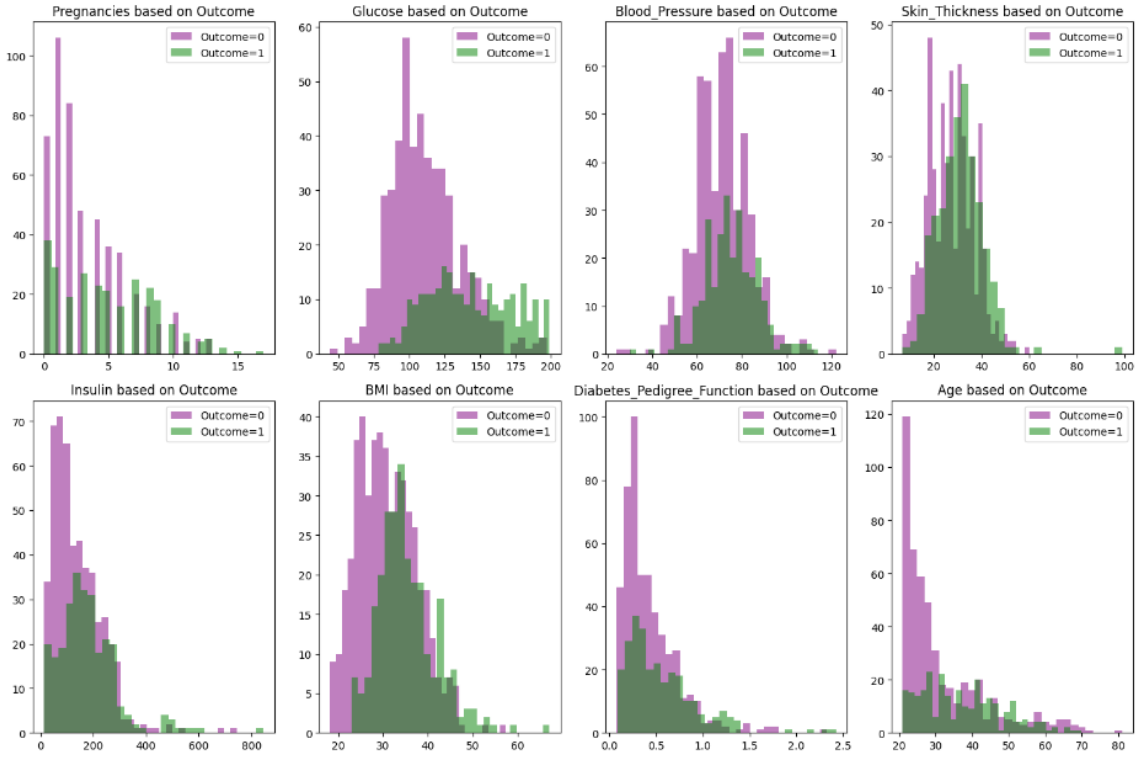
Decisioni progettuali e Strumenti utilizzati

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 insulina (Insuline), 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 (atomi) 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
```

È stata modificata la base di conoscenza con l'aggiunta di tre regole.

- La prima regola aggiunta ha lo scopo di verificare la percentuale di rischio della paziente affetta. Nella regola sono utilizzati solo i valori di Glucosio, Pressione del sangue, insulina, BMI e la probabilità genetica di aver ereditato il diabete, poiché sono i fattori che influiscono maggiormente. Si sono create delle clausole per ciascun fattore dove si specificano i valori che servono per indicare se la percentuale di rischio è alta o meno. Nella regola 'rischio_diabete' viene usata una somma ponderata dei diversi fattori e ad ognuno è assegnato un peso diverso, utile per avere una somma bilanciata.

```
% Regola per calcolare il rischio di diabete
rischio_diabete(Glucosio, PressioneSanguigna, Insulina, BMI, ProbabilitaDiabete, Eta, Rischio) :-
    (
    fattore_genetico(ProbabilitaDiabete, FattoreGenetico),
    fattore_glucosio(Glucosio, FattoreGlucosio),
    fattore_bmi(BMI, FattoreBMI),
    fattore_eta(Eta, FattoreEta),
    fattore_pressione(PressioneSanguigna, FattorePressione),
    fattore_insulina(Insulina, FattoreInsulina),

    % Somma ponderata dei vari fattori, con pesi che sommano a 1.0
    Rischio is (FattoreGenetico * 0.25) + (FattoreInsulina * 0.25) +
               (FattoreGlucosio * 0.2) + (FattoreBMI * 0.15) +
               (FattoreEta * 0.1) + (FattorePressione * 0.05)
    ).

% Clausola per il fattore genetico
fattore_genetico(ProbabilitaDiabete, FattoreGenetico) :-
    (
        ProbabilitaDiabete > 1.0 -> FattoreGenetico = 1.0; % Alto rischio genetico
        ProbabilitaDiabete > 0.5 -> FattoreGenetico = 0.7; % Rischio moderato
        FattoreGenetico = 0.4 % Basso rischio genetico
    ).

% Clausola per il fattore glucosio
fattore_glucosio(Glucosio, 0.9) :- Glucosio >= 126.
fattore_glucosio(Glucosio, 0.5) :- Glucosio >= 100, Glucosio < 126.
fattore_glucosio(_, 0.1).

% Clausola per il fattore insulina
fattore_insulina(Insulina, 0.9 ) :- Insulina >= 25. % alto rischio
fattore_insulina(Insulina, 0.5) :- Insulina >=20, Insulina <25. % medio rischio
fattore_insulina(_, 0.2).
```

```
% Clausola per il fattore BMI
fattore_bmi(BMI, 0.9) :- BMI >= 30.
fattore_bmi(BMI, 0.5) :- BMI >= 25, BMI < 30.
fattore_bmi(_, 0.2).

% Clausola per il fattore età
fattore_eta(Eta, 0.9) :- Eta >= 60.
fattore_eta(Eta, 0.7) :- Eta >= 40, Eta < 60.
fattore_eta(Eta, 0.4) :- Eta >= 20, Eta < 40.
fattore_eta(_, 0.2).

% Clausola per il fattore pressione sanguigna
fattore_pressione(PressioneSanguigna, 0.7) :- PressioneSanguigna >= 80.
fattore_pressione(_, 0.2).
```

Esempio di query:

```
?- rischio_diabete(145, 32, 45, 0.8, 50, Rischio).
% Rischio = 0.71.
```

- Un’altra regola aggiunta è quella che serve a determinare che tipo di diabete potrebbe avere la paziente. Questa regola si divide in 4, poiché per ciascun tipo di diabete sono definiti i diversi valori dei parametri che caratterizzano ciascuna tipologia.

Il tipo 1 si verifica nelle pazienti che hanno una bassa insulina, hanno giovane età, BMI inferiore a 30, pressione sanguigna normale ma alto livello di glucosio che è simbolo di diabete. Il tipo 2 si verifica quando le pazienti hanno insulina normale o alta, età avanzata, BMI superiore a 30 (indice di obesità o sovrappeso), pressione alta e un livello di glucosio elevato. Nel livello di prediabete si classificano le pazienti che hanno glucosio moderato, BMI superiore a 25 e pressione sanguigna moderata; quindi, si trovano in una situazione di pre-diabete. Infine, l'ultimo tipo è quello indeterminato, dove si hanno pazienti con bassa insulina ma avendo un'età avanzata, non è possibile determinare a che tipologia appartengono oppure possiedono una forma particolare di diabete.

```
% Regola per identificare il tipo di diabete
tipo_diabete(Insulina, Eta, BMI, PressioneSanguigna, Glucosio, tipo_1) :-
    Insulina < 25,                  % Insulina bassa
    Glucosio >= 126,                % Livelli elevati di glucosio
    Eta < 40,                       % Età sotto i 40 anni
    BMI < 30,                       % BMI generalmente inferiore (meno comune lobesità in Tipo 1)
    PressioneSanguigna < 80, !.     % Pressione sanguigna non alta (non associata a Tipo 1)

tipo_diabete(Insulina, Eta, BMI, PressioneSanguigna, Glucosio, tipo_2) :-
    Insulina >= 25,                 % Insulina normale o alta
    Glucosio >= 126,                % Livelli elevati di glucosio
    Eta >= 40,                      % Età pari o superiore a 40 anni
    BMI >= 30,                      % Spesso associato a obesità o sovrappeso
    PressioneSanguigna >= 80, !.    % Pressione sanguigna alta o moderata

tipo_diabete(Insulina, Eta, BMI, PressioneSanguigna, Glucosio, prediabete) :-
    Glucosio >= 100,                % Livelli di glucosio elevati ma non diabetici
    Glucosio < 126,
    BMI >= 25,                      % Sovrappeso ma non necessariamente obesità
    PressioneSanguigna >= 80, !.    % Pressione sanguigna moderata

tipo_diabete(Insulina, Eta, BMI, PressioneSanguigna, Glucosio, indeterminato) :-
    Insulina < 25,                  % Insulina bassa
    Glucosio >= 126,                % Livelli elevati di glucosio
    Eta >= 40, !.                   % Età avanzata ma bassa insulina: caso meno comune, indeterminato
```

Esempio di query:

```
?- tipo_diabete(20, 25, 22, 75, 130, Tipo).
Tipo = tipo_1.

?- tipo_diabete(30, 50, 32, 85, 140, Tipo).
Tipo = tipo_2.

?- tipo_diabete(15, 55, 28, 78, 130, Tipo).
Tipo = indeterminato.

?- tipo_diabete(28, 45, 27, 82, 110, Tipo).
Tipo = prediabete.
```

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

Mean of AdaBoost predictions: 0.38961038961038963
Standard Deviation of AdaBoost predictions: 0.4876619053381452
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

Mean of AdaBoost predictions: 0.3246753246753247
Standard Deviation of AdaBoost predictions: 0.46825341239792057
```

```
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

Mean of AdaBoost predictions: 0.37662337662337664
Standard Deviation of AdaBoost predictions: 0.4845391715890295
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

Mean of AdaBoost predictions: 0.3246753246753247
Standard Deviation of AdaBoost predictions: 0.46825341239792057
```

```
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

Mean of AdaBoost predictions: 0.4025974025974026
Standard Deviation of AdaBoost predictions: 0.49042097632465464
```

Sono state condotte valutazioni preliminari delle prestazioni dei modelli, includendo metriche chiave come **precision** e **recall** all'interno di un *classification report*. Inoltre, sono state calcolate la **media** e la **deviazione standard** per analizzare la variabilità delle prestazioni dei modelli nei vari esperimenti.

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'utilizzo della media e della deviazione standard per queste metriche permette di valutare la stabilità delle prestazioni dei modelli e identificare le configurazioni che, in media, offrono i risultati più consistenti.

La media (o valore medio) è una misura della prestazione complessiva del modello, calcolata come la somma delle prestazioni in diversi esperimenti divisa per il numero di esperimenti eseguiti. La media permette di comprendere il livello generale delle prestazioni del modello, offrendo una rappresentazione totale dell'accuratezza, della precision, della recall, o di altre metriche. Una media più alta suggerisce che il modello tende ad avere buone prestazioni nella maggior parte delle esecuzioni.

La deviazione standard misura la variabilità o la dispersione delle prestazioni del modello rispetto alla media. Una deviazione standard bassa indica che il modello ha prestazioni costanti su più esecuzioni, il che è desiderabile in molti contesti. Al contrario, una deviazione standard elevata suggerisce che il modello ha prestazioni più variabili, il che potrebbe indicare un comportamento imprevedibile o una sensibilità ai dati. In sintesi, la deviazione standard è essenziale per valutare l'affidabilità di un modello: anche se la media delle performance è elevata, una deviazione standard troppo alta potrebbe segnalare una scarsa stabilità.

Confronto delle prestazioni dei modelli

Dall'analisi iniziale, il modello ‘XGBoost’ ha dimostrato le prestazioni migliori in termini di F2 Score, oltre a buoni valori di media e deviazione standard. A seguire

il modello ‘Random Forest’ che presenta un’alta accuracy e un buon valore di F2 Score.

Il modello ‘K-Nearest Neighbors’, pur avendo un’accuracy più bassa rispetto a ‘Random Forest’, presenta alti valori di media e deviazione standard. Successivamente il modello ‘AdaBoost’ presenta dei risultati accettabili senza raggiungere l’efficacia dei precedenti modelli.

Infine, il ‘Decision Tree’ risulta essere il modello con le prestazioni più basse.

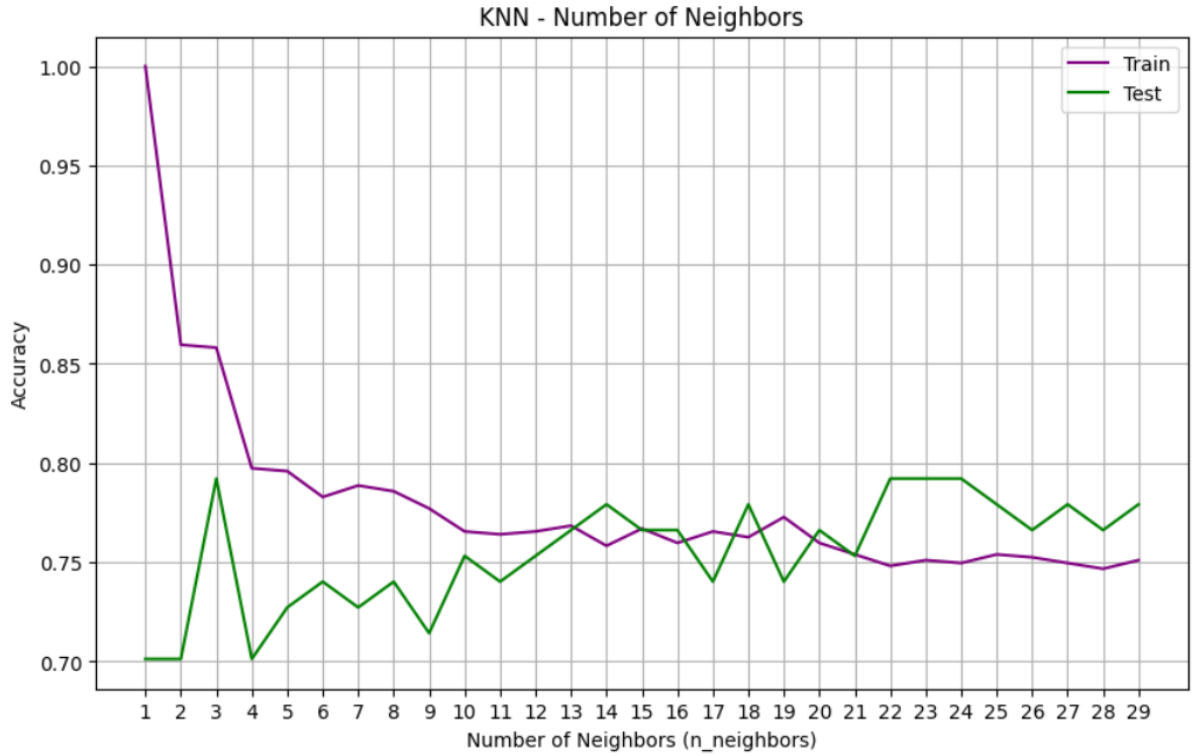
Per migliorare le prestazioni degli ultimi tre modelli, è stata condotta un'analisi più approfondita per identificare i parametri ottimali dei modelli KNN, AdaBoost e Decision Tree, essendo quelli con accuracy più basse.

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 un’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.

Uno degli iperparametri più importanti del KNN è il numero di vicini, indicato con **n_neighbors**, che definisce quanti vicini prossimi verranno considerati per effettuare la classificazione.



Sono stati scelti 30 valori diversi per l'iperparametro **'n_neighbors'**. Il comportamento del modello rispetto a questo parametro è stato analizzato sia per il set di training sia per il set di test.

L'accuracy dei valori del test set mostra un andamento vario, raggiungendo il picco massimo quando n_neighbors=2, dove il modello ottiene le migliori prestazioni considerando solo i due vicini più prossimi.

Per quanto riguarda l'accuracy dei valori di training set, tende a diminuire rapidamente con l'aumentare dei valori di n_neighbors. Dal vicino numero 2 in poi, l'accuratezza del training set cala drasticamente e continua a stabilizzarsi intorno a n_neighbors = 19, dove si mantiene costante fino a n_neighbors = 29.

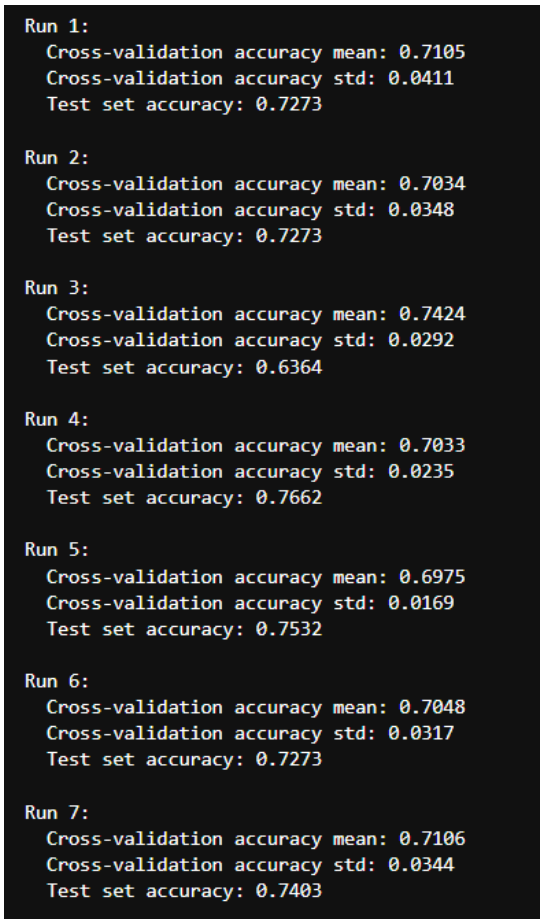
Questo comportamento evidenzia come il numero di vicini possa influenzare il rapporto tra overfitting e underfitting. Valori troppo bassi di n_neighbors possono portare il modello a sovradattare i dati (overfitting), mentre valori troppo alti possono portare a prestazioni subottimali sul test set (underfitting).

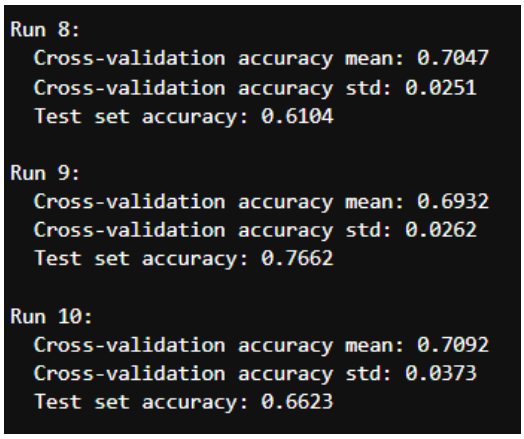
Per avere una valutazione più approfondita del modello, è stata utilizzata una cross-validation stratificata in 5 'k_folds' su 10 run differenti, assicurando che la

distribuzione delle classi venga mantenuta in ogni fold ottenendo una valutazione bilanciata. Successivamente si sono create due liste, cv_means e cv_stds, per poter memorizzare rispettivamente i valori ottenuti dalla media e quelli ottenuti dalla deviazione standard delle accuracy di ciascun run.

Per ogni run viene eseguito un ciclo con i valori dei 'neighbors' calcolando le accuratezze sia su valori di training che su valori di test set e poi memorizzarle nelle rispettive liste (accuracy_train_knn, accuracy_test_knn).

Dai 10 run eseguiti, si ottengono le seguenti accuratezze sia per la cross-validation sia per il test set:

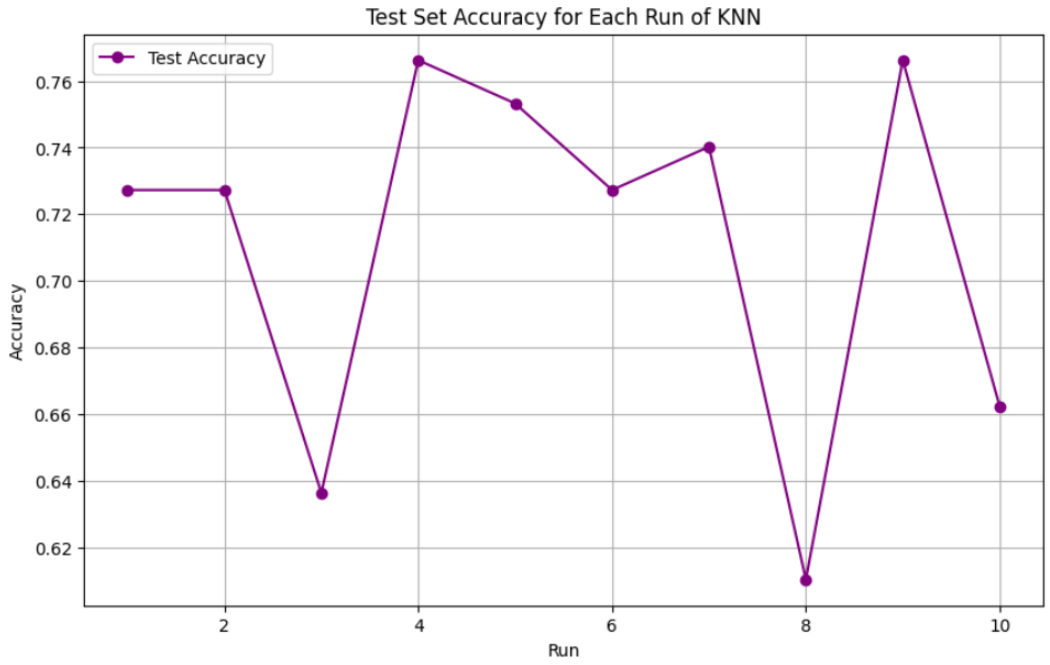


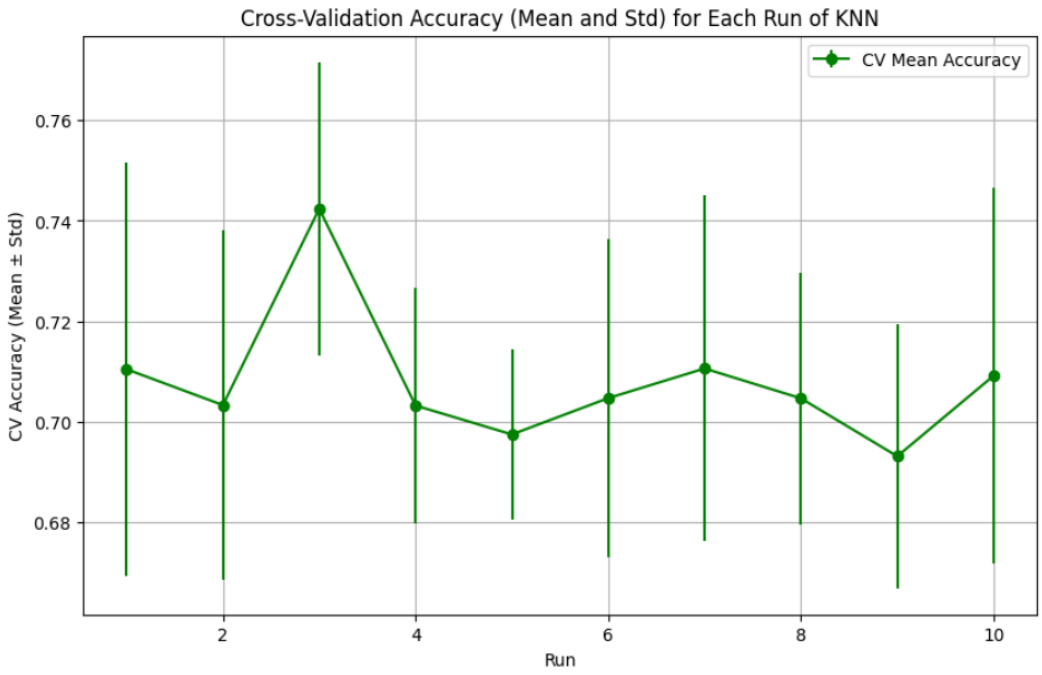


Nei run 4 e 9, il modello ha ottenuto le migliori performance in termini di accuratezza sul test set e cross validation. Questo indica che, per quelle specifiche suddivisioni dei dati in training/test (avvenute in ciascun run), il modello KNN ha imparato meglio le caratteristiche dei dati, riuscendo a predire correttamente gli esiti con una maggiore precisione. Eseguire più run e ottenere valutazioni su suddivisioni diverse serve a evitare che i risultati siano casuali. Se il modello avesse ottenuto buoni risultati solo in un run e molto peggiori in altri, si otterrebbe un caso di overfitting o di instabilità. Tuttavia, poiché i run 4 e 9

hanno prodotto i migliori risultati ma gli altri run hanno comunque mantenuto performance competitive, si può dire che il modello è stabile e capace di generalizzare bene.

Per visualizzare l'andamento dei risultati, sono stati creati altri due grafici uno sui valori del test set per ogni run e l'altro su quelli della cross validation con le rispettive medie e deviazioni standard.





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.

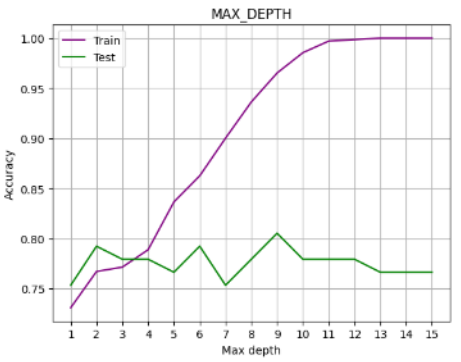
Durante il processo di addestramento, l'algoritmo cerca di suddividere i dati in gruppi omogenei, minimizzando la disomogeneità all'interno di ciascun gruppo. Questo processo viene ripetuto in modo ricorsivo fino a quando non si raggiunge una condizione di arresto, come ad esempio la profondità massima dell'albero o un numero minimo di campioni in ciascun nodo foglia.

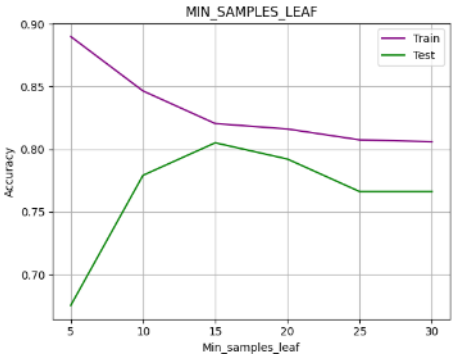
Tra gli iperparametri all'interno dell'algoritmo di Decision Tree sono presenti:

- **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**: non è un iperparametro ma è la strategia per eseguire la selezione di un attributo per la divisione ("best" miglior attributo, "random" attributo casuale).

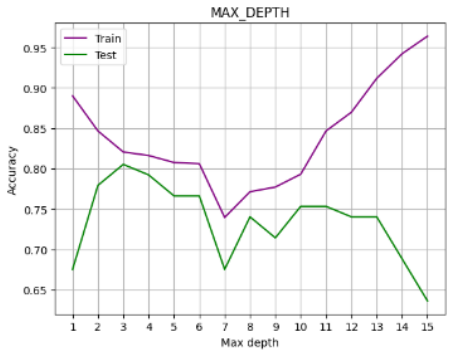
Anche per questo modello di classificazione è stata eseguita una valutazione più approfondita attraverso l'esecuzione di 10 run differenti e per ognuno viene calcolata una cross validation stratificata in 5 folds. Per ciascuno di questi run, viene eseguito un ciclo sui valori di 'max_depth' e su 'min_samples_leaf' entrambi rappresentati con dei grafici che mostrano le performance del modello rispetto agli iperparamentri citati.

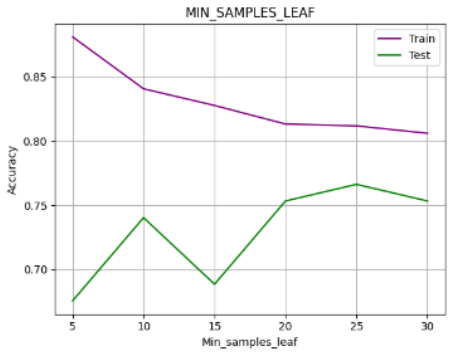
Run 1:



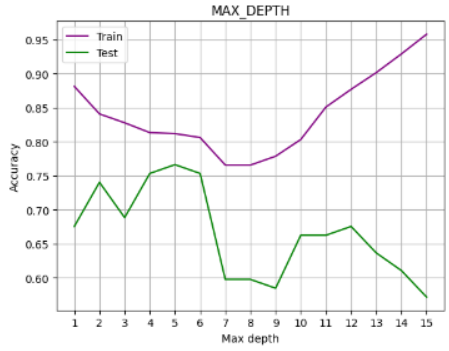


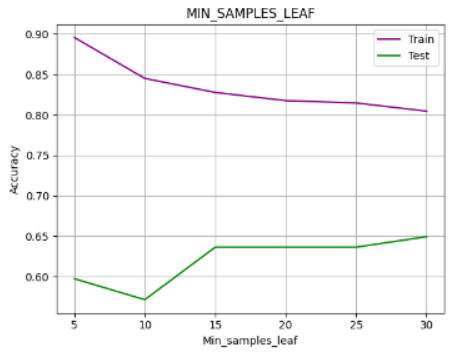
Run2:



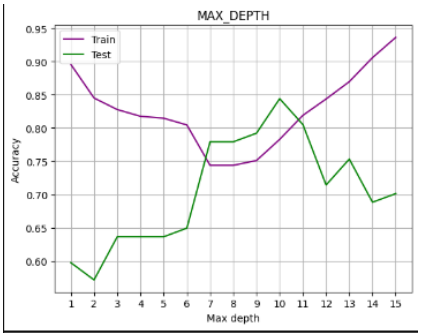


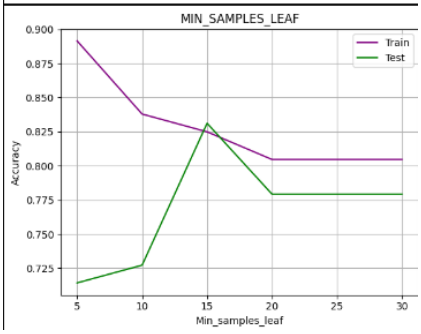
Run 3:





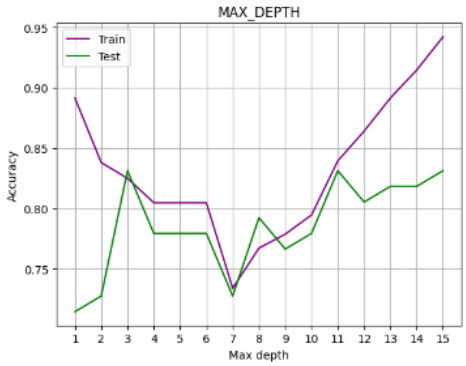
Run 4:

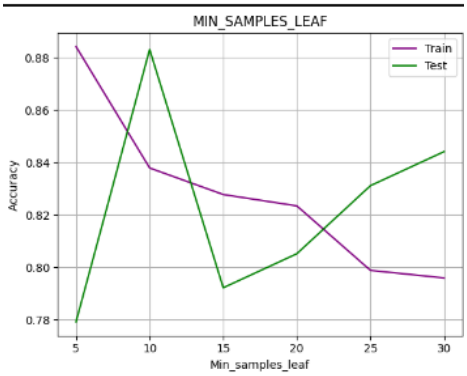




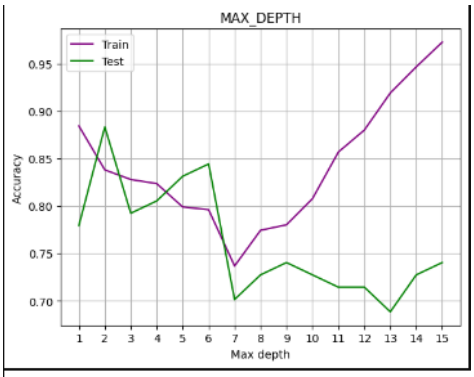
Run 5:

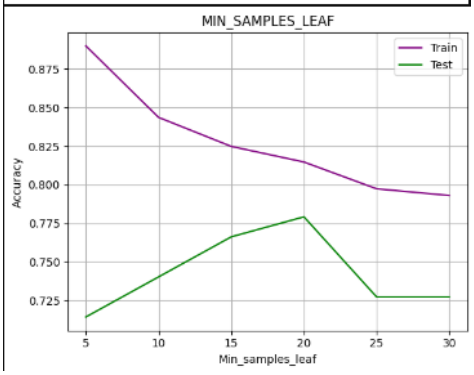
Run 6:



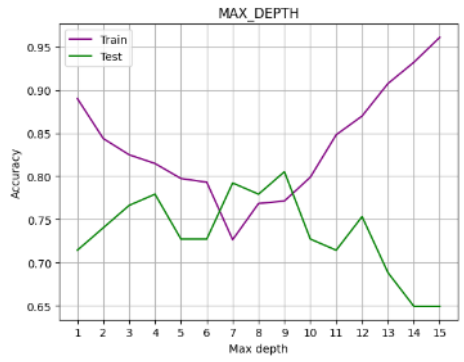


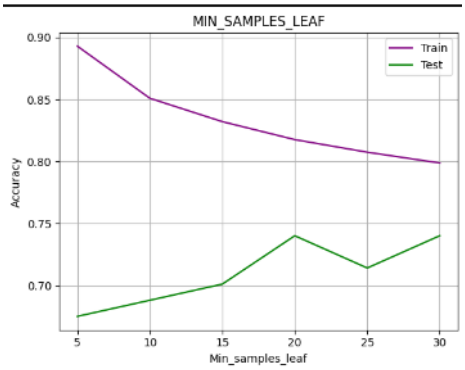
Run 7:



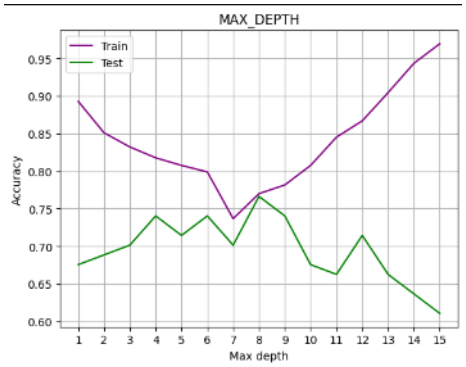


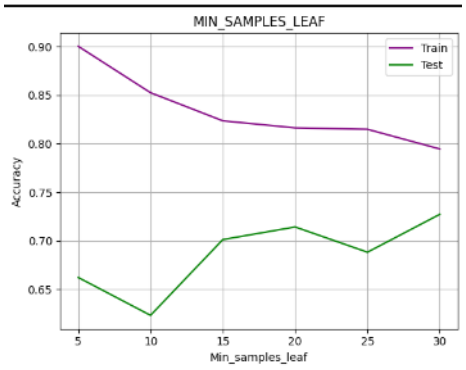
Run 8:



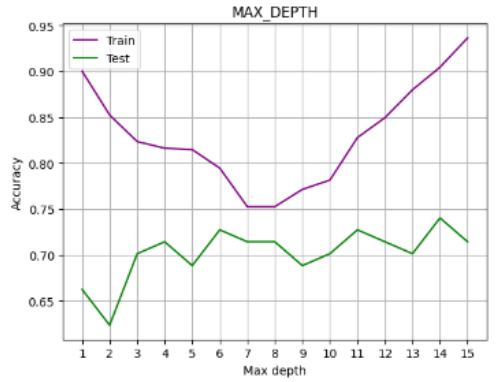


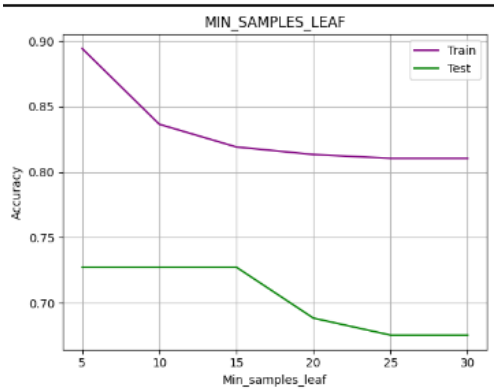
Run 9:

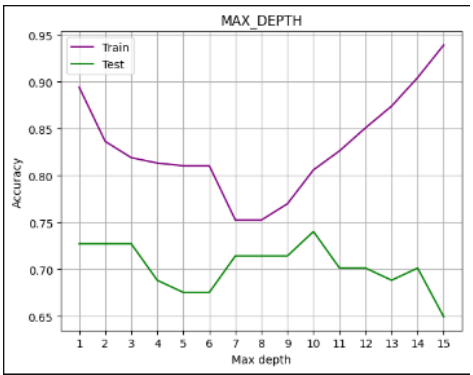


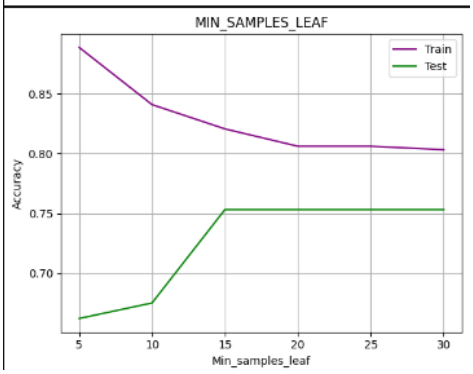


Run 10:









Nei run 4 e 6 l'accuratezza sui valori di test set raggiunge il suo picco per un valore di max_depth rispettivamente a 10 e 2, indicando che questa è la profondità ottimale per l'albero di decisione in quei run. Dopo questo punto, la performance potrebbe iniziare a calare a causa di overfitting.

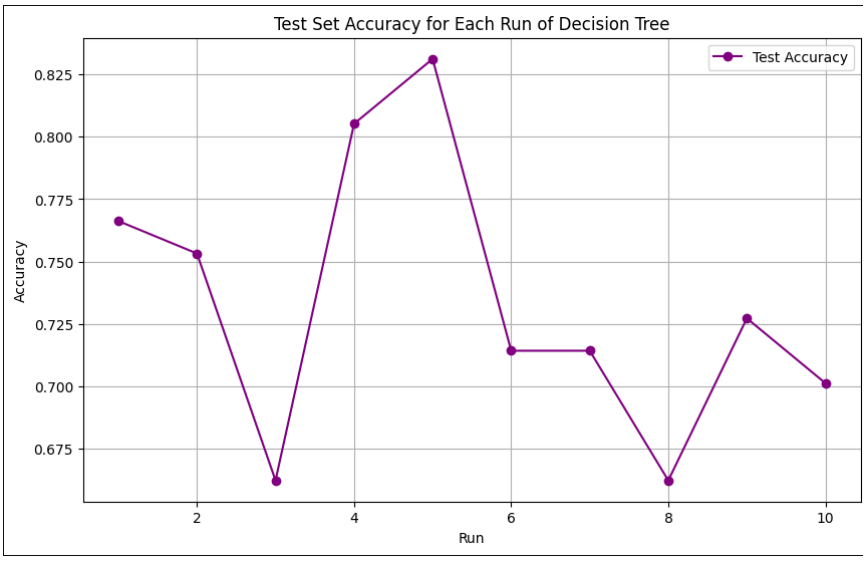
Nei run 1, 4 e 5, i grafici mostrano chiaramente che esiste un compromesso ottimale tra la complessità dell'albero (controllata da max_depth e min_samples_leaf) e le sue performance di generalizzazione.

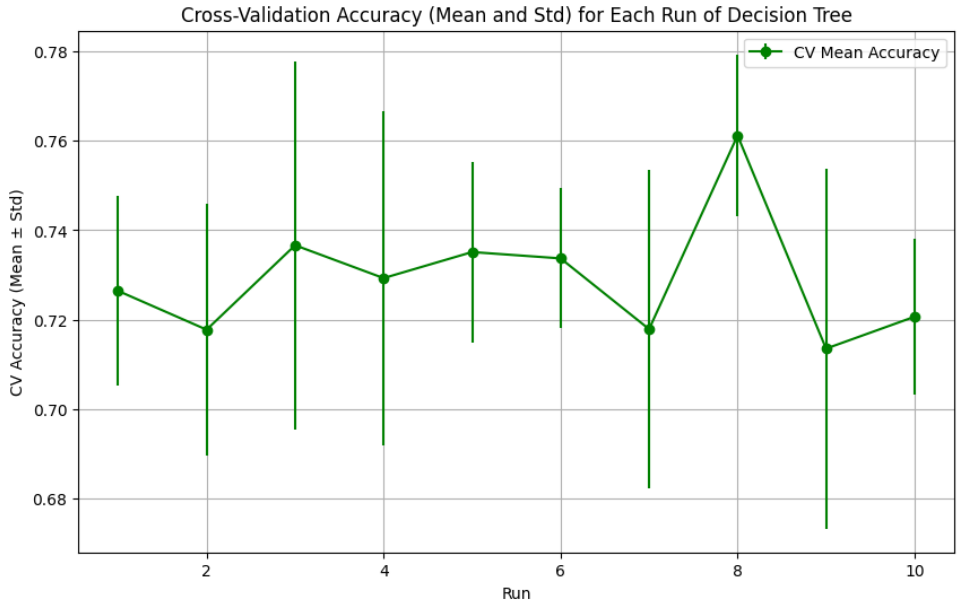
Nei migliori run, il valore di max_depth raggiunge un punto ottimale prima che l'overfitting cominci a manifestarsi, e il valore di min_samples_leaf mantiene le foglie abbastanza grandi da garantire che il modello non si adatti troppo ai dati di training.

```
Run 1:
  Cross-validation accuracy mean: 0.7264
  Cross-validation accuracy std: 0.0212
  Test set accuracy: 0.7662

Run 2:
  Cross-validation accuracy mean: 0.7178
  Cross-validation accuracy std: 0.0282
  Test set accuracy: 0.7532

Run 3:
  Cross-validation accuracy mean: 0.7366
  Cross-validation accuracy std: 0.0411
  Test set accuracy: 0.6623

Run 4:
  Cross-validation accuracy mean: 0.7293
  Cross-validation accuracy std: 0.0373
  Test set accuracy: 0.8052

Run 5:
  Cross-validation accuracy mean: 0.7351
  Cross-validation accuracy std: 0.0202
  Test set accuracy: 0.8312

Run 6:
  Cross-validation accuracy mean: 0.7337
  Cross-validation accuracy std: 0.0156
  Test set accuracy: 0.7143

Run 7:
  Cross-validation accuracy mean: 0.7179
  Cross-validation accuracy std: 0.0355
  Test set accuracy: 0.7143
```

```
Run 8:
  Cross-validation accuracy mean: 0.7612
  Cross-validation accuracy std: 0.0181
  Test set accuracy: 0.6623

Run 9:
  Cross-validation accuracy mean: 0.7135
  Cross-validation accuracy std: 0.0403
  Test set accuracy: 0.7273

Run 10:
  Cross-validation accuracy mean: 0.7207
  Cross-validation accuracy std: 0.0173
  Test set accuracy: 0.7013
```

Da questi risultati è infatti confermato che i migliori run siano 1,4,5 definiti dai valori di accuracy calcolati tramite cross validation e accuracy del test set, ma il picco di cross validation per il valore medio più alto si trova nel run 8. I grafici seguenti mostrano l'andamento delle accuratezze del modello secondo questi fattori dimostrando come anche questo modello si sia adattato bene ai dati.



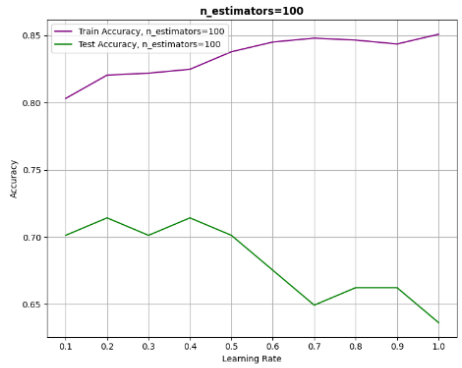


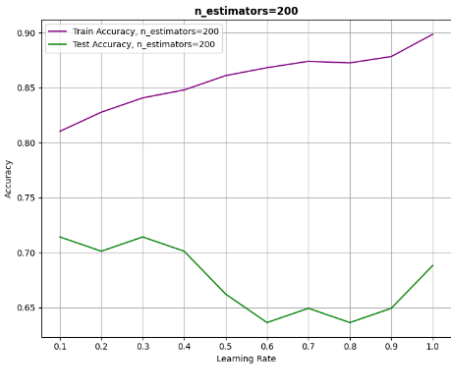
ADABOOST

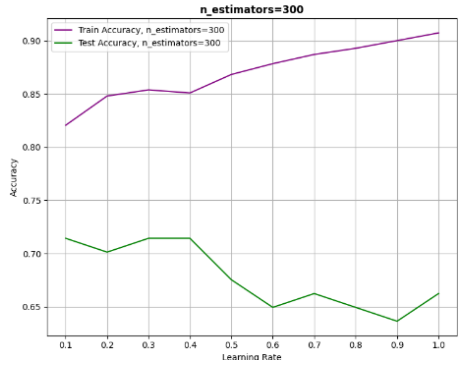
Il modello **AdaBoost** ha la capacità di trasformare una serie di classificatori deboli in un classificatore forte concentrandosi su esempi di addestramento più complessi. Questo processo iterativo di addestramento e pesatura degli esempi, consente all'algoritmo di correggere gli errori commessi dai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

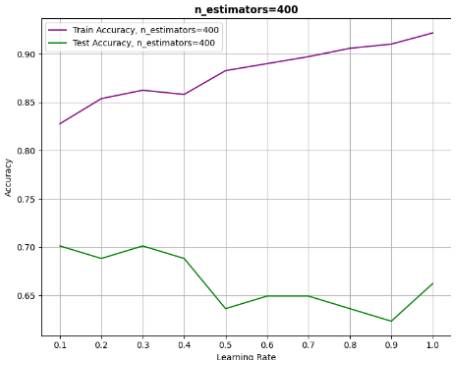
Tra gli iperparametri più importanti utilizzati nell'algoritmo di AdaBoost sono presenti:

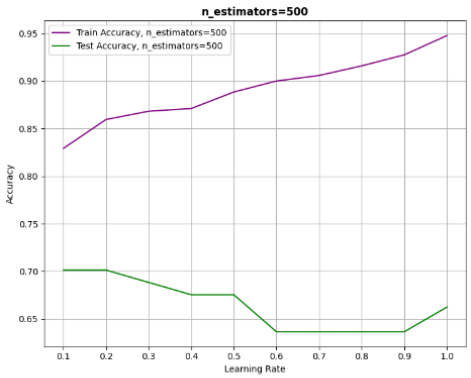
- **n_estimators:** è l'iperparametro principale dell'algoritmo. Rappresenta il numero di stimatori che verranno utilizzati per costruire il modello. La scelta di più stimatori porta ad un miglioramento di prestazioni.
- **learning_rate:** iperparametro che definisce la proporzione dei dati che verranno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

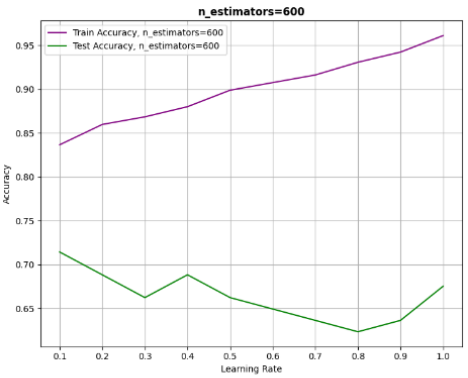

n_estimators=100
Train Accuracy, n_estimators=100
Test Accuracy, n_estimators=100
Accuracy
Learning Rate

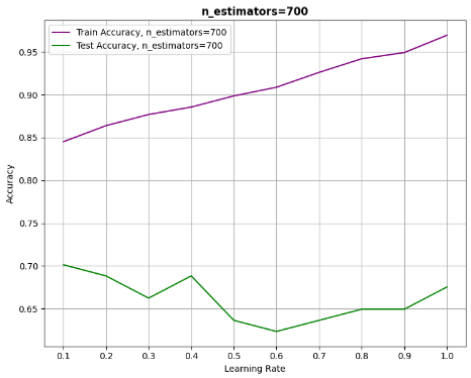

n_estimators=200
Train Accuracy, n_estimators=200
Test Accuracy, n_estimators=200
Accuracy
Learning Rate

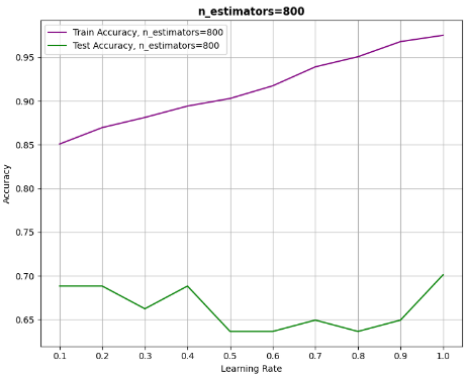

n_estimators=300
Train Accuracy, n_estimators=300
Test Accuracy, n_estimators=300
Accuracy
Learning Rate

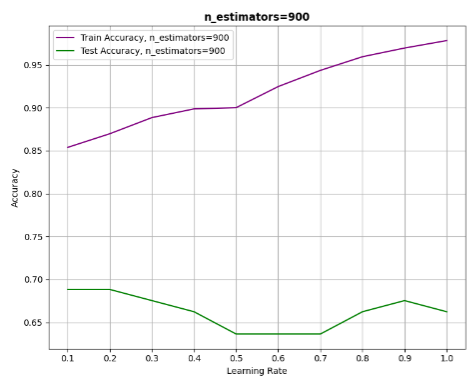

n_estimators=400
Train Accuracy, n_estimators=400
Test Accuracy, n_estimators=400
Accuracy
Learning Rate

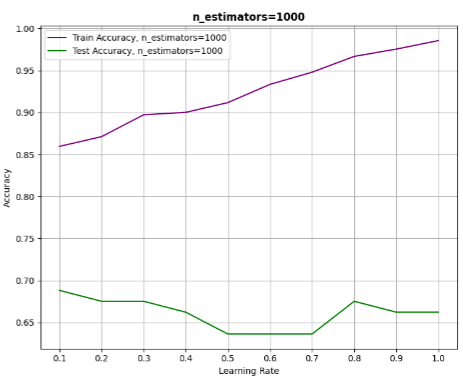

n_estimators=500
Train Accuracy, n_estimators=500
Test Accuracy, n_estimators=500
Accuracy
Learning Rate


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

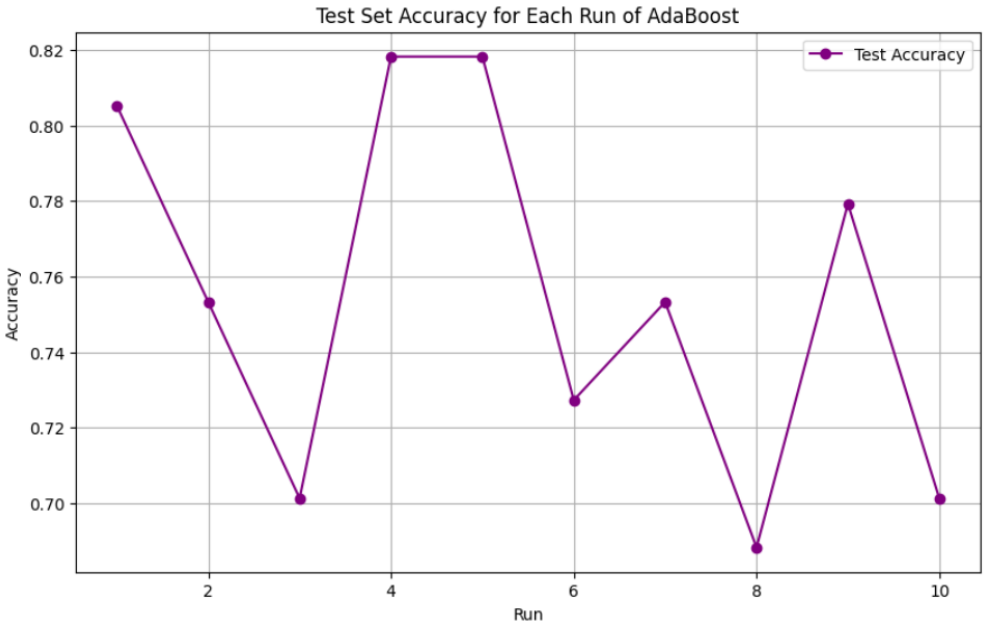
Anche per il modello AdaBoost viene eseguita una valutazione con l'utilizzo di una cross validation stratificata in 5 folds che viene eseguita per ciascuno dei 10 run differenti. I risultati ottenuti sono i seguenti:

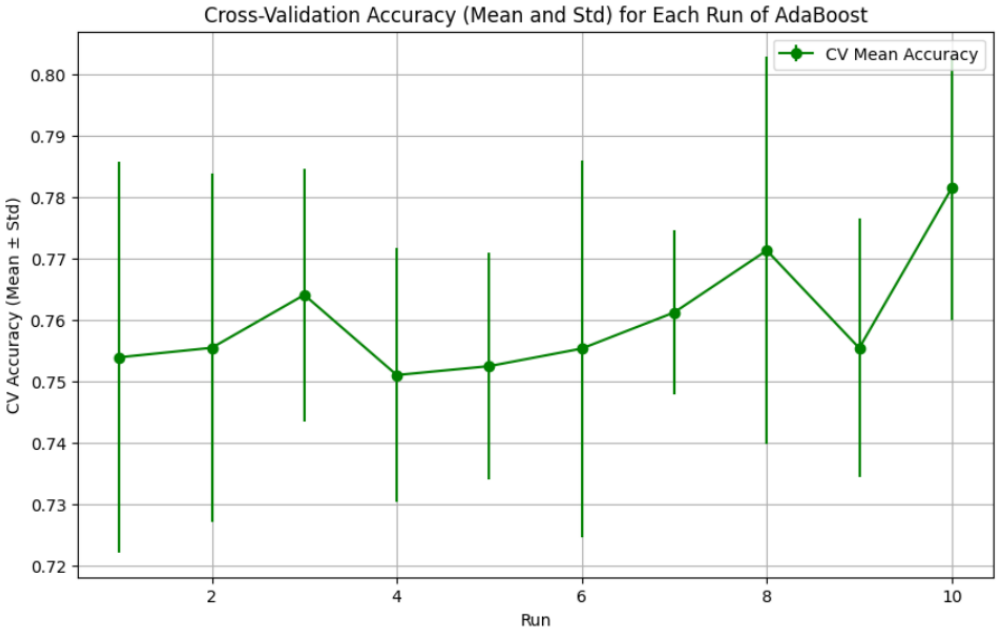
```
Run 1:
  Cross-validation accuracy mean: 0.7539
  Cross-validation accuracy std: 0.0317
  Test set accuracy: 0.8052

Run 2:
  Cross-validation accuracy mean: 0.7555
  Cross-validation accuracy std: 0.0282
  Test set accuracy: 0.7532

Run 3:
  Cross-validation accuracy mean: 0.7641
  Cross-validation accuracy std: 0.0206
  Test set accuracy: 0.7013

Run 4:
  Cross-validation accuracy mean: 0.7510
  Cross-validation accuracy std: 0.0207
  Test set accuracy: 0.8182

Run 5:
  Cross-validation accuracy mean: 0.7525
  Cross-validation accuracy std: 0.0184
  Test set accuracy: 0.8182

Run 6:
  Cross-validation accuracy mean: 0.7554
  Cross-validation accuracy std: 0.0307
  Test set accuracy: 0.7273

Run 7:
  Cross-validation accuracy mean: 0.7612
  Cross-validation accuracy std: 0.0134
  Test set accuracy: 0.7532
```

```
Run 8:
  Cross-validation accuracy mean: 0.7713
  Cross-validation accuracy std: 0.0315
  Test set accuracy: 0.6883

Run 9:
  Cross-validation accuracy mean: 0.7554
  Cross-validation accuracy std: 0.0211
  Test set accuracy: 0.7792

Run 10:
  Cross-validation accuracy mean: 0.7815
  Cross-validation accuracy std: 0.0214
  Test set accuracy: 0.7013
```

I migliori run che presentano un'alta accuracy dei dati del test set sono 1,4,5, ma i run che presentano una bassa accuracy della cv di deviazione standard e alta media sono i run 3,7 e 10. Questo significa che, se le accuratezze sono abbastanza stabili e alte, il modello è robusto e generalizza bene. Una deviazione standard bassa indica che il modello è abbastanza consistente attraverso diverse suddivisioni dei dati, mentre deviazioni alte possono segnalare instabilità.

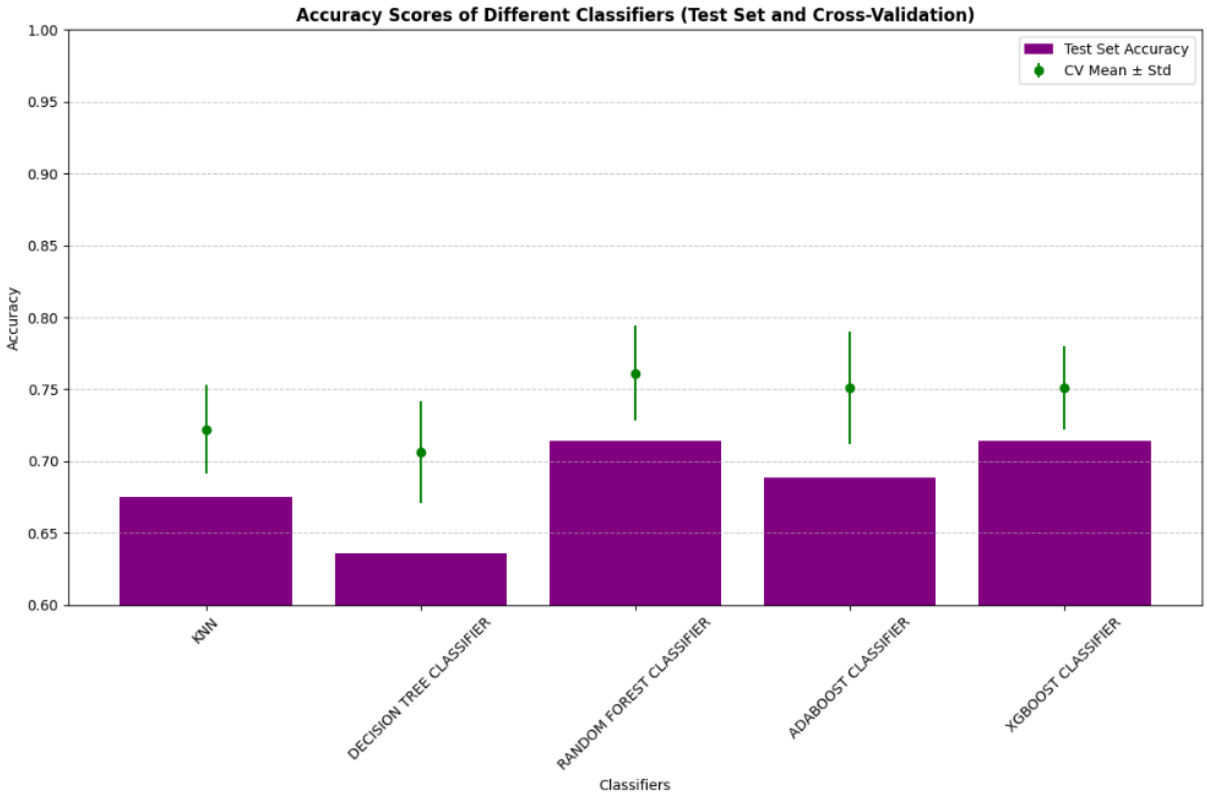
Tutti questi risultati sono visibili nei seguenti grafici:


Test Set Accuracy for Each Run of AdaBoost
Test Accuracy
Accuracy
Run
0.82
0.80
0.78
0.76
0.74
0.72
0.70
2
4
6
8
10


Cross-Validation Accuracy (Mean and Std) for Each Run of AdaBoost
CV Mean Accuracy
CV Accuracy (Mean ± Std)
Run
0.80
0.79
0.78
0.77
0.76
0.75
0.74
0.73
0.72
2
4
6
8
10

CONCLUSIONE

In conclusione, è stato eseguito un confronto tra i cinque classificatori per poter verificare se le valutazioni iniziali si sono rivelate veritiere o meno.




```
Classifier: KNN
  Test Set Accuracy: 0.6753
  CV Mean Accuracy: 0.7222
  CV Std Dev: 0.0309
---------------------------------------
Classifier: DECISION TREE CLASSIFIER
  Test Set Accuracy: 0.6364
  CV Mean Accuracy: 0.7062
  CV Std Dev: 0.0355
---------------------------------------
Classifier: RANDOM FOREST CLASSIFIER
  Test Set Accuracy: 0.7143
  CV Mean Accuracy: 0.7612
  CV Std Dev: 0.0330
---------------------------------------
Classifier: ADABOOST CLASSIFIER
  Test Set Accuracy: 0.6883
  CV Mean Accuracy: 0.7511
  CV Std Dev: 0.0391
---------------------------------------
Classifier: XGBOOST CLASSIFIER
  Test Set Accuracy: 0.7143
  CV Mean Accuracy: 0.7511
  CV Std Dev: 0.0288
---------------------------------------
```

Dai risultati del grafico, è evidente come ci siano dei cambiamenti abbastanza rilevanti rispetto all'analisi iniziale.

In effetti, il modello Random si conferma il migliore in termini di accuratezza sul test set, insieme al modello XGBoost. Tuttavia, il modello Random Forest ha migliori valori di cv per la media, mentre XGBoost presenta bassi valori di cv della deviazione standard, indicando una buona generalizzazione e adattamento del modello.

Il modello AdaBoost supera il KNN sia in termini di accuratezza del test set sia nella media della cross-validation, sebbene presenti una deviazione standard più alta rispetto a KNN, il che indica una maggiore variabilità nelle performance.

Infine, il Decision Tree si conferma il modello con la minore accuratezza sul test set. Nonostante ciò, presenta una buona stabilità, come evidenziato da una deviazione standard contenuta nelle performance in cross-validation.