1. OBIETTIVO DEL PROGETTO

L'obiettivo del progetto è quello di acquisire esperienza nell'analizzare un dataset mediante l'applicazione di metodi di classificazione. In questa esperienza verranno utilizzati diversi classificatori, aggiustando inoltre i loro parametri per avere un'analisi soddisfacente.

Le fasi di questo progetto sono divise in:

1. Analisi preliminare, in cui si carica il dataset e si analizza la struttura, trovando la variabile target.
2. Il preprocessing dei dati, gestendo eventuali valori mancanti.
3. Dividere il dataset in training set e test set.
4. Applicazione dei classificatori.
5. Valutazione dei risultati.

2. DESCRIZIONE DEL DATASET

Il dataset fornito (Dry Bean Dataset.xslx) è un insieme di dati che contiene misurazioni e caratteristiche relative a fagioli secchi; le colonne (features) del dataset forniscono informazioni dettagliate sulle proprietà geometriche e statistiche di ciascun fagiolo, mentre la colonna "Class" (target) indica la classe a cui appartiene ciascun fagiolo.

Le colonne features sono le variabili indipendenti, utilizzate per predire il risultato e sono generalmente delle variabili input mentre la colonna Class è il target, la variabile che si cerca di predire.

Lo scopo finale è addestrare modelli di classificazione che possono identificare automaticamente la classe di nuovi fagioli sulla base delle loro caratteristiche misurate.

Di seguito, si riporta una sezione del dataset in questione.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	Class
2	28395	610.291	208.1781167	173.888747	1.197191424	0.549812187	28715	190.1410973	0.763922518	0.988855999	0.958027126	0.913357755	0.007331506	0.003147289	0.834222388	0.998723889	SEKER
3	28734	638.018	200.5247957	182.7344194	1.097356461	0.411785251	29172	191.2727505	0.783968133	0.984985603	0.887033637	0.953860842	0.006978659	0.003563624	0.909850506	0.998430331	SEKER
4	29380	624.11	212.8261299	175.9311426	1.209712656	0.562727317	29690	193.4109041	0.778113248	0.989558774	0.947849473	0.908774239	0.007243912	0.003047733	0.825870617	0.999066137	SEKER

Area: l'area dell'oggetto.
Perimeter: il perimetro dell'oggetto.
MajorAxisLength: la lunghezza dell'asse maggiore dell'oggetto.
MinorAxisLength: la lunghezza dell'asse minore dell'oggetto.
AspectRatio: il rapporto tra la lunghezza dell'asse maggiore e quella dell'asse minore.
Eccentricity: l'eccentricità dell'oggetto.
ConvexArea: l'area convessa dell'oggetto.
EquivDiameter: Il diametro equivalente dell'oggetto.
Extent: l'estensione dell'oggetto.
Solidity: la solidità dell'oggetto.
Roundness: la rotondità dell'oggetto.
Compactness: la compattezza dell'oggetto.
ShapeFactor1: fattore di forma dell'oggetto.
ShapeFactor2: fattore di forma dell'oggetto.
ShapeFactor3: fattore di forma dell'oggetto.
ShapeFactor4: fattore di forma dell'oggetto.

La colonna 'Class' assume uno dei seguenti valori, definendo la tipologia di oggetto (tipo di fagioli):

- Secker
- Barbunya
- Bombay
- Cali
- Horoz

- Sira
- Dermason

Analizzando il dataset, si è individuata la colonna ‘Class’ come variabile target, il che permette di impostare le fasi successive di sviluppo con il metodo dell’apprendimento supervisionato (in quanto il dataset è etichettato). Man mano che i dati di input vengono immessi nel modello, esso adegua i suoi coefficienti fino a quando il fitting del modello non sarà stato eseguito in modo appropriato, il che avviene come parte del processo di convalida incrociata.

2.1 ANALISI PRELIMINARI DEL DATASET

Prima di iniziare effettivamente a implementare i vari modelli, è necessario avere una panoramica generale del dataset nella sua totalità, per definire e analizzare le migliori soluzioni da applicare nelle fasi successive.
Dopo aver caricato il dataset, utilizzando la libreria Pandas, assegnando il dataframe alla variabile df, vengono fatte le seguenti operazioni:

- Stampa le dimensioni del dataset, cioè il numero di righe e colonne:

  ```
  (13611, 17)
  ```

  Il dataset completo, quindi, ha 13611 righe e 17 colonne.

- Fornire informazioni di base sul dataset, inclusi tipi di dati, presenza di valori nulli e memoria utilizzata.

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             13611 non-null  int64
 1   Perimeter        13611 non-null  float64
 2   MajorAxisLength  13611 non-null  float64
 3   MinorAxisLength  13611 non-null  float64
 4   AspectRation     13611 non-null  float64
 5   Eccentricity     13611 non-null  float64
 6   ConvexArea       13611 non-null  int64
 7   EquivDiameter    13611 non-null  float64
 8   Extent           13611 non-null  float64
 9   Solidity         13611 non-null  float64
 10  roundness        13611 non-null  float64
 11  Compactness      13611 non-null  float64
 12  ShapeFactor1     13611 non-null  float64
 13  ShapeFactor2     13611 non-null  float64
 14  ShapeFactor3     13611 non-null  float64
 15  ShapeFactor4     13611 non-null  float64
 16  Class            13611 non-null  object
dtypes: float64(14), int64(2), object(1)
memory usage: 1.8+ MB
```

Questa panoramica è fondamentale per evidenziare eventuali valori da gestire prima di darli in input ai modelli (es. valori nulli, normalizzazione, variabili categoriche...)

Un’indicazione importante è data dalle colonne di tipo ‘object’, che rappresentando variabili categoriche devono essere convertite in un tipo di dato numerico; in questo caso, solo il target deve essere pre-elaborato in questo modo.

- Visualizzazione delle prime cinque righe del dataset per avere un'idea iniziale dei dati.

```
First 5 Rows of the Dataset:
    Area  Perimeter  MajorAxisLength  ...  ShapeFactor3  ShapeFactor4  Class
0  28395    610.291       208.178117  ...      0.834222      0.998724  SEKER
1  28734    638.018       200.524796  ...      0.909851      0.998430  SEKER
2  29380    624.110       212.826130  ...      0.825871      0.999066  SEKER
3  30008    645.884       210.557999  ...      0.861794      0.994199  SEKER
4  30140    620.134       201.847882  ...      0.941900      0.999166  SEKER

[5 rows x 17 columns]
```

Vengono stampate le prime cinque righe per comprendere e avere una vista uniforme dei valori presenti nel dataset e del loro valore.
Utile, soprattutto, per evidenziare eventuali normalizzazioni necessarie.

- Stampa delle statistiche riassuntive (media, deviazione standard, minimo, massimo) per le colonne numeriche del dataset.

```
Summary Statistics:
                Area     Perimeter  ...  ShapeFactor3  ShapeFactor4
count   13611.000000  13611.000000  ...  13611.000000  13611.000000
mean    53048.284549    855.283459  ...      0.643590      0.995063
std     29324.095717    214.289696  ...      0.098996      0.004366
min     20420.000000    524.736000  ...      0.410339      0.947687
25%     36328.000000    703.523500  ...      0.581359      0.993703
50%     44652.000000    794.941000  ...      0.642044      0.996386
75%     61332.000000    977.213000  ...      0.696006      0.997883
max    254616.000000   1985.370000  ...      0.974767      0.999733

[8 rows x 16 columns]
```

Come si può vedere, le colonne saranno solo 16, in quanto la colonna target è di tipo ‘object’ e viene ignorata.
Questi valori possono essere fondamentali in caso di data augmentation o dati mancanti (si potrebbero sostituire con la media).

- Stampa del numero di valori unici in ciascuna colonna del dataset.

```
Unique Values in Each Column:
Area: 12011 unique values
Perimeter: 13416 unique values
MajorAxisLength: 13543 unique values
MinorAxisLength: 13543 unique values
AspectRation: 13543 unique values
Eccentricity: 13543 unique values
ConvexArea: 12066 unique values
EquivDiameter: 12011 unique values
Extent: 13535 unique values
Solidity: 13526 unique values
roundness: 13543 unique values
Compactness: 13543 unique values
ShapeFactor1: 13543 unique values
ShapeFactor2: 13543 unique values
ShapeFactor3: 13543 unique values
ShapeFactor4: 13543 unique values
Class: 7 unique values
```

Questi valori aiutano a capire la differenziazione tra i dati di ciascuna feature e il grado di dettaglio al quale si andrà a lavorare.

- Stampa del nome della variabile target

```
Target Variable: Class
```

- Visualizzazione della distribuzione dei valori nella colonna target, associando a ogni etichetta il numero di esempi presenti nella classe ‘Class’.

```
Distribution of Class:
DERMASON    3546
SIRA        2636
SEKER       2027
HOROZ       1928
CALI        1630
BARBUNYA    1322
BOMBAY       522
Name: Class, dtype: int64
```

- Stampa l'elenco delle features.

```
Features:
['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength', 'AspectRation', 'Eccentricity', 'ConvexArea', 'EquivDiameter', 'Extent', 'Solidity', 'roundness', 'Compactness', 'ShapeFactor1',
'ShapeFactor2', 'ShapeFactor3', 'ShapeFactor4']
```

2.2 PROCESSING DEI DATI

La seconda fase necessaria riguarda il processing dei dati, cioè l’analisi dettagliata e la conversione eventuale di questi al fine di evitare la presenza eventuale di valori o situazioni che possono portare problemi o errori durante l’addestramento dei modelli.

- Rimuove le righe con valori mancanti.
- Identifica e rimuove gli outlier basandosi sullo z-score.
- Converte le colonne categoriche in numeriche, concentrandosi sulla colonna target ('Class').

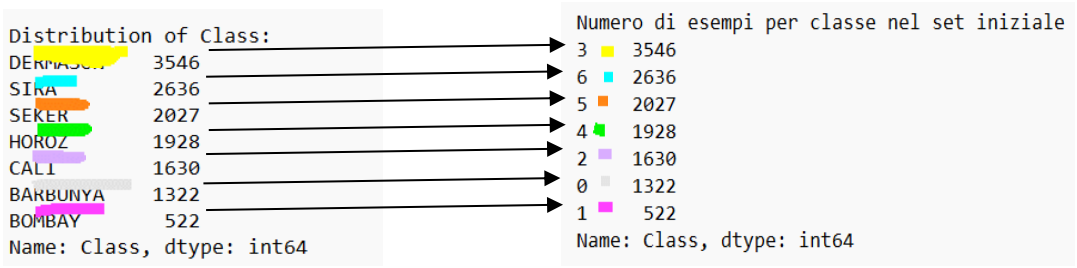
```
Categorical Columns: Index(['Class'], dtype='object')
```

Come già individuato visivamente in precedenza, si cercano tutte le colonne di tipo ‘object’ (ottenendo solo la variabile target).

- Stampa il numero di esempi per classe nel dataset completo, questa volta sostituendo alle etichette il valore numerico associato.

```
Numero di esempi per classe nel set iniziale (prima della divisione e prima del bilanciamento):
3    3546
6    2636
5    2027
4    1928
2    1630
0    1322
1     522
Name: Class, dtype: int64
```

Notiamo, quindi, il seguente mapping dei valori categorici con valori numerici:



2.3 SPLIT IN TRAIN / TEST SET

Una volta preparati i dati, questi devono essere divisi in un set di training (usato per addestrare i modelli) e un set di test (usato per valutare le prestazioni di questi).
I dati utilizzati nell'addestramento, devono avere una numerosità maggiore rispetto a quelli riservati al test, per evitare problemi di addestramento non sufficiente (underfitting).
Un' altra sfida critica è rappresentata dall'overfitting.
Questo fenomeno si verifica quando il modello impara in modo eccessivamente dettagliato dai dati di addestramento, memorizzando pattern e associazioni errate o del tutto casuali. Tale comportamento può condurre a notevoli difficoltà nella generalizzazione del modello su nuovi dati.
Si è cercato di mitigare queste due problematiche nel seguente modo:

UNDERFITTING: Si è riservato il 70% dei valori al training e il 30% al test, garantendo un addestramento su un set di dati significativo ed il più generale possibile.

OVERFITTING: Invece di optare per una classica suddivisione train / test set, si è scelto di implementare la cross - validation nell'addestramento dei modelli (spiegata nei punti successivi) e un bilanciamento del dataset di training, per evitare la numerosità troppo elevata di campioni di una stessa classe e limitare quasi al massimo la casualità.

In questa fase avviene:

- Split del dataset in set di training e test.
- Stampa le dimensioni sui set di training e test (features e etichette)

```
Train Set Info:
X_train Shape: (9527, 16)
y_train Shape: (9527,)

Test Set Info:
X_test Shape: (4084, 16)
y_test Shape: (4084,)
```

Come si può vedere, il numero totale di campioni nel dataset individuato prima (13611) è stato diviso in 2 parti, una (train) con 9527 campioni e l'altra (test) con 40842.

- Stampa del numero di esempi per classe nel set di training (prima del bilanciamento).

```
Numero di esempi per classe nel set di train (dopo la divisione e prima del bilanciamento):
3    2503
6    1837
5    1408
4    1340
2    1151
0     927
1     361
Name: Class, dtype: int64
```

  La nuova distribuzione evidenzia come classe con numerosità minima la numero 1 (usata nel bilanciamento sotto).

- Bilanciamento del set di training e stampa della nuova distribuzione (post bilanciamento).

```
Numero di esempi per classe nel set di train (dopo la divisione e dopo il bilanciamento):
0    361
1    361
2    361
3    361
4    361
5    361
6    361
Name: Class, dtype: int64
```

2.4 APPLICAZIONE DEI CLASSIFICATORI

Dopo la fase di preparazione e suddivisione del dataset, la procedura si sposta alla definizione e all'addestramento dei modelli, focalizzandosi sul set di training. In tutte le proposte implementazioni di modelli, è stata adottata la tecnica di cross-validation di tipo k-fold, integrata con un meccanismo di early stopping per evitare iterazioni inutili.
Una volta diviso il dataset, il set di test viene lasciato da parte e utilizzato solo dopo aver implementato e ottenuto i modelli finali, al fine di valutare le prestazioni su dati mai visti in precedenza. L'attenzione è quindi concentrata sul dataset di training.
La cross-validation di tipo k-fold funziona suddividendo il set di dati di addestramento in k parti uguali e utilizzandone una per la validazione, mentre le rimanenti k-1 vengono utilizzate per il training. Questo processo viene ripetuto k volte, in modo che ciascuna fold sia usata esattamente k-1 volte per il training e una volta per la validazione. Questa tecnica permette di considerare come set di training insiemi di dati sempre diversi, generalizzando al massimo e limitando la casualità del campionamento.
Inoltre, la cross-validation offre il vantaggio di fornire misure di bontà del modello basate sulla media delle valutazioni ottenute su ciascun fold. Ciò genera stime più affidabili rispetto alla classica suddivisione statica in set di training e validation.

2.4.1 DECISION TREE

Nella fase di definizione e addestramento dei modelli, il primo modello implementato è stato un Decision Tree Classifier. Questo modello è stato configurato con la possibilità di riproducibilità dei risultati grazie al parametro random_state fissato a 42.
Per trovare la combinazione ottimale di iperparametri per il Decision Tree, è stata utilizzata la tecnica della grid search, una procedura che esplora diverse combinazioni di parametri. La differenza principale tra parametri ed iperparametri consiste nell' esternalità dei secondi al modello, dovendo essere impostati manualmente dal ML engineer.
Alcuni metodi necessitano gli iperparametri, come in questo caso, la Grid Search, metodo per l'ottimizzazione e per la ricerca della combinazione ottimale degli iperparametri.

In particolare, si sono variati la profondità massima dell'albero(max_depth), il numero minimo di campioni richiesti per suddividere un nodo interno (min_samples_split) e il numero minimo di campioni richiesti per formare una foglia(min_samples_leaf).
Al fine di ottenere stime affidabili delle performance del modello, è stata implementata la cross-validation di tipo k-fold, con k pari a 100.
Inoltre, è stato implementato l'early stopping, che interrompe l'addestramento se non si osservano miglioramenti dell'accuratezza del modello per un numero specificato di iterazioni (specificato da max_no_improvement_dt)
Il modello migliore ottenuto dalla grid search è stato selezionato, e successivamente è stato addestrato e valutato su ogni fold della cross-validation. In ogni iterazione, l'accuratezza del Decision Tree è stata monitorata, sia sul train che sul validation set e il processo è stato interrotto anticipatamente se non si osservavano miglioramenti per un numero massimo di iterazioni prestabilito (2). Questa strategia di addestramento e validazione mira a ottenere un modello generalizzato, evitando l'overfitting e garantendo allo stesso tempo un'adeguata accuratezza.

Le macro-fasi sono le seguenti:

- Definizione del Modello: viene istanziato un modello di albero decisionale (DecisionTreeClassifier) con un seed per la riproducibilità.

- Definizione dei Parametri per la Grid Search: Si specificano i parametri dell'albero decisionale da testare durante la grid search. I parametri includono la profondità massima dell'albero (max_depth), il numero minimo di campioni richiesti per suddividere un nodo interno (min_samples_split), e il numero minimo di campioni richiesti per formare una foglia (min_samples_leaf).

- Configurazione della Grid Search e Addestramento del Modello: Viene configurata una grid search (GridSearchCV) per esplorare tutte le combinazioni possibili di parametri. Il modello viene quindi addestrato sul set di dati di addestramento.

- Valutazione del Modello con Early Stopping: Il modello addestrato viene valutato attraverso una cross-validation stratificata con uno schema di K-fold. Durante ogni iterazione, vengono registrate le previsioni sul set di addestramento e di validazione, e viene monitorata l'accuratezza sul set di validazione.
  Si implementa un meccanismo di early stopping per interrompere l'addestramento se non si osserva un miglioramento dell'accuratezza entro un numero massimo di iterazioni.

- Calcolo e Stampa delle Metriche: Alla fine delle iterazioni, vengono calcolate e stampate le metriche di valutazione del modello. Queste metriche includono la matrice di confusione, la precisione, il richiamo e l'F1 score, sia per il set di addestramento che per quello di validazione.

ACCURACY RAGGIUNTA CON CROSS VALIDATION E EARLY STOPPING:

```
Fold 1: Accuratezza Decision Tree = 0.8846153846153846
Fold 2: Accuratezza Decision Tree = 0.9615384615384616
Fold 3: Accuratezza Decision Tree = 1.0
Fold 4: Accuratezza Decision Tree = 0.9230769230769231
Fold 5: Accuratezza Decision Tree = 0.8461538461538461

Early stopping Decision Tree! Miglior accuratezza raggiunta: 1.0
```

TRAINING SET

```
Matrice di Confusione Decision Tree (Set di Addestramento):
[[1767    0    4    0    0    0   14]
 [   0 1785    0    0    0    0    0]
 [  12    0 1767    0   11    0    0]
 [   0    0    0 1717    0    0   68]
 [   0    0    0    5 1757    0   23]
 [   0    0    0    7    0 1763   20]
 [   3    0    0   47    0    3 1732]]

Precision Decision Tree (Set di Addestramento): 0.9829934294045047
Recall Decision Tree (Set di Addestramento): 0.9826469412235106
F1 Score Decision Tree (Set di Addestramento): 0.9827523820770337
```

VALIDATION SET

```
Matrice di Confusione Decision Tree (Set di Validazione):
[[19  0  1  0  0  0  0]
 [ 0 20  0  0  0  0  0]
 [ 2  0 13  0  0  0  0]
 [ 0  0  0 18  0  0  2]
 [ 0  0  0  0 19  0  1]
 [ 0  0  0  0  0 15  0]
 [ 0  0  0  3  1  0 16]]

Precision Decision Tree (Set di Validazione): 0.9231443994601889
Recall Decision Tree (Set di Validazione): 0.9230769230769231
F1 Score Decision Tree (Set di Validazione): 0.9227391788302702
```

2.4.2 KNN

Nella fase successiva dell'analisi, ci siamo concentrati sull'implementazione di un modello basato su K-Nearest Neighbors (KNN) Classifier. Questo modello è stato scelto per esplorare ulteriori approcci alla classificazione dei dati.
Il processo è stato avviato con la normalizzazione delle feature attraverso lo StandardScaler per garantire che tutte le variabili avessero un impatto equo sulla classificazione.
Questa fase è fondamentale per assicurare che le misure delle diverse features siano omogenee, contribuendo così a migliorare le prestazioni del modello.
Successivamente, è stata eseguita una Grid Search per identificare la combinazione ottimale di iperparametri per il modello KNN. I parametri considerati includono il numero di vicini (n_neighbors), il peso assegnato ai vicini (weights), e l'algoritmo utilizzato per il calcolo della distanza (algorithm).
Questa procedura ha esplorato varie combinazioni di parametri al fine di individuare la configurazione più performante per il nostro modello.
Per valutare le prestazioni del modello e ottenere stime affidabili, abbiamo implementato la cross-validation di tipo k-fold, suddividendo il set di addestramento in 100 parti uguali. Ogni iterazione di questa procedura ha utilizzato una parte per la validazione e le restanti per l'addestramento, consentendo così di generalizzare al massimo il modello evitando una staticità nella suddivisione del dataset.
Al fine di gestire eventuali problemi di overfitting e migliorare l'efficienza del processo di addestramento, abbiamo introdotto l'early stopping. Questa strategia ci ha consentito di interrompere l'addestramento se non si osservavano miglioramenti dell'accuratezza del modello per un numero specificato di iterazioni.
Infine, il modello migliore ottenuto dalla Grid Search è stato selezionato e valutato su ciascuna fold della cross-validation. In ogni iterazione, l'accuratezza del KNN è stata monitorata sia sul set di addestramento che su quello di validazione, interrompendo il processo anticipatamente se non si osservavano miglioramenti per un numero massimo di iterazioni prestabilito. Questa strategia mira a ottenere un modello generalizzato, evitando l'overfitting, e contemporaneamente garantendo un'adeguata accuratezza nella classificazione dei dati.

Step eseguiti:
• Normalizzazione delle Feature: Prima di addestrare il modello KNN, le feature sono state normalizzate utilizzando lo StandardScaler per garantire che abbiano la stessa scala e non influenzino in modo sbilanciato il modello.

• Definizione del Modello KNN e Parametri per la Grid Search: È stato istanziato un modello di classificazione KNN (KNeighborsClassifier) e sono stati definiti i parametri da esplorare durante la Grid Search. Questi parametri includono il numero di vicini (n_neighbors), il peso assegnato ai vicini (weights), e l'algoritmo utilizzato per il calcolo della distanza (algorithm).

• Configurazione della Grid Search e Addestramento del Modello KNN: È stata configurata una procedura di Grid Search (GridSearchCV) per esplorare tutte le combinazioni possibili di parametri per il modello KNN. Successivamente, il modello è stato addestrato sul set di dati di addestramento utilizzando la combinazione ottimale di iperparametri identificata dalla Grid Search.

• Valutazione del Modello con Early Stopping: Il modello addestrato è stato valutato mediante una cross-validation stratificata con uno schema di K-fold, dove K è pari a 100. Durante ogni iterazione, sono state registrate le previsioni sia sul set di addestramento che su quello di validazione. Inoltre, è stato implementato un meccanismo di early stopping per interrompere l'addestramento se non si osservava alcun miglioramento dell'accuratezza entro un numero massimo di iterazioni specificato.

• Calcolo e Stampa delle Metriche: Al termine delle iterazioni, sono state calcolate e stampate diverse metriche di valutazione del modello KNN. Queste metriche includono la matrice di confusione, la precisione, il richiamo e l'F1 score, sia per il set di addestramento che per quello di validazione. Questo processo fornisce una valutazione completa delle prestazioni del modello KNN sulla base dei dati di addestramento e di validazione.

ACCURACY RAGGIUNTA CON CROSS VALIDATION E EARLY STOPPING

```
Fold 1: Accuratezza KNN = 0.9230769230769231
Fold 2: Accuratezza KNN = 1.0
Fold 3: Accuratezza KNN = 1.0
Fold 4: Accuratezza KNN = 1.0

Early stopping KNN! Miglior accuratezza raggiunta: 1.0
```

TRAINING SET

```
Matrice di Confusione KNN (Set di Addestramento):
[[1428    0    0    0    0    0    0]
 [   0 1428    0    0    0    0    0]
 [   0    0 1432    0    0    0    0]
 [   0    0    0 1428    0    0    0]
 [   0    0    0    0 1428    0    0]
 [   0    0    0    0    0 1432    0]
 [   0    0    0    0    0    0 1428]]

Precision KNN (Set di Addestramento): 1.0
Recall KNN (Set di Addestramento): 1.0
F1 Score KNN (Set di Addestramento): 1.0
```

VALIDATION SET

```
Matrice di Confusione KNN (Set di Validazione):
[[15  0  1  0  0  0  0]
 [ 0 16  0  0  0  0  0]
 [ 0  0 12  0  0  0  0]
 [ 0  0  0 16  0  0  0]
 [ 0  0  0  0 16  0  0]
 [ 0  0  0  0  0 12  0]
 [ 0  0  0  1  0  0 15]]

Precision KNN (Set di Validazione): 0.9820744865993736
Recall KNN (Set di Validazione): 0.9807692307692307
F1 Score KNN (Set di Validazione): 0.9807970524099557
```

2.4.3 SVM

Nella fase successiva dell'analisi, è stato implementato un modello basato su Support Vector Machine (SVM).
il primo passo è quello di normalizzare le feature attraverso lo Standard Scaler, assicurandoci che tutte le variabili avessero un impatto equo sulla classificazione (valori tra 0 e 1). Questo passaggio è fondamentale per uniformare le misure delle diverse features e migliorare l'efficacia complessiva del modello, evitando che alcuni valori elevati causassero problematiche nell'addestramento.
Successivamente, viene eseguita una Grid Search per identificare la combinazione ottimale di iperparametri per il modello SVM. I parametri considerati includono il parametro di regolarizzazione (C), il tipo di kernel (linear, poly, rbf, sigmoid), e il grado del polinomio (degree) nel caso di kernel polinomiale. Questa procedura ha esplorato varie combinazioni di parametri per individuare la configurazione più performante per il modello.
Per valutare le prestazioni del modello e ottenere stime affidabili, viene implementata la cross-validation di tipo k-fold, suddividendo il set di addestramento in 100 parti uguali. Ogni iterazione di questa procedura ha utilizzato una parte per la validazione e le restanti per l'addestramento, massimizzando così la generalizzazione del modello.
Al fine di mitigare eventuali problemi di overfitting e ottimizzare l'efficienza del processo di addestramento, viene utilizzato anche qui l'early stopping.
Infine, il modello migliore ottenuto dalla Grid Search è stato selezionato e valutato su ciascuna fold della cross-validation. In ogni iterazione, l'accuratezza del SVM è stata monitorata sia sul set di addestramento che su quello di validazione, interrompendo il processo anticipatamente se non si osservavano miglioramenti per un numero massimo di iterazioni prestabilito.

• Normalizzazione delle Feature: Prima di addestrare il modello SVM, le feature sono state normalizzate utilizzando lo StandardScaler per garantire una scala uniforme e un impatto bilanciato delle variabili sul modello.

• Definizione del Modello SVM e Parametri per la Grid Search: È stato istanziato un modello di Support Vector Machine (SVM) con il kernel lineare e sono stati definiti i parametri da esplorare durante la Grid Search. Questi includono il parametro di regolarizzazione (C), il tipo di kernel (linear, poly, rbf, sigmoid), e il grado del polinomio (degree).

• Configurazione della Grid Search e Addestramento del Modello SVM: È stata configurata una procedura di Grid Search (GridSearchCV) per esplorare tutte le possibili combinazioni di parametri per il modello SVM. Successivamente, il modello è stato addestrato sul set di dati di addestramento utilizzando la combinazione ottimale di iperparametri identificata dalla Grid Search.

• Valutazione del Modello con Early Stopping: Il modello addestrato è stato valutato mediante una cross-validation stratificata con uno schema di K-fold, dove K è pari a 100. Durante ogni iterazione, sono state registrate le previsioni sia sul set di addestramento che su quello di validazione. Inoltre, è stato implementato un meccanismo di early stopping per interrompere l'addestramento se non si osservava alcun miglioramento dell'accuratezza entro un numero massimo di iterazioni specificato.

• Calcolo e Stampa delle Metriche: Al termine delle iterazioni, sono state calcolate e stampate diverse metriche di valutazione del modello SVM, incluse la matrice di confusione, la precisione, il richiamo e l'F1 score, sia per il set di addestramento che per quello di validazione. Questo processo fornisce una valutazione completa delle prestazioni del modello SVM sulla base dei dati di addestramento e di validazione.

ACCURACY RAGGIUNTA CON CROSS VALIDATION E EARLY STOPPING

```
Fold 1: Accuratezza SVM = 0.9230769230769231
Fold 2: Accuratezza SVM = 1.0
Fold 3: Accuratezza SVM = 1.0
Fold 4: Accuratezza SVM = 1.0

Early stopping SVM! Miglior accuratezza raggiunta: 1.0
```

TRAINING SET

```
Matrice di Confusione SVM (Set di Addestramento):
[[1350    0   46    0    0   16   16]
 [   0 1428    0    0    0    0    0]
 [  28    0 1380    0   16    4    4]
 [   0    0    0 1334    0   12   82]
 [   4    0   16   12 1372    0   24]
 [   4    0    0   20    0 1376   32]
 [   0    0    4  115    4   20 1285]]

Precision SVM (Set di Addestramento): 0.952697745078423
Recall SVM (Set di Addestramento): 0.9521191523390644
F1 Score SVM (Set di Addestramento): 0.9523048147850074
```

VALIDATION SET

```
Matrice di Confusione SVM (Set di Validazione):
[[15  0  1  0  0  0  0]
 [ 0 16  0  0  0  0  0]
 [ 0  0 12  0  0  0  0]
 [ 0  0  0 16  0  0  0]
 [ 0  0  0  0 16  0  0]
 [ 0  0  0  0  0 12  0]
 [ 0  0  0  1  0  0 15]]

Precision SVM (Set di Validazione): 0.9820744865993736
Recall SVM (Set di Validazione): 0.9807692307692307
F1 Score SVM (Set di Validazione): 0.9807970524099557
```

2.4.4 RANDOM FOREST

Nella fase successiva dell'analisi, viene implementato un modello basato su Random Forest Classifier, una scelta che offre una robustezza aggiuntiva attraverso l'ensemble di alberi decisionali. L'obiettivo è ottimizzare il modello e valutarne accuratamente le prestazioni.
Il processo è iniziato con la definizione del modello Random Forest e la successiva esecuzione di una Grid Search.
Quest'ultima aveva l'obiettivo di identificare la combinazione ottimale di iperparametri per il modello.
I parametri considerati includevano il numero di alberi nell'ensemble (n_estimators), la massima profondità di ciascun albero (max_depth), il numero minimo di campioni richiesti per suddividere un nodo interno (min_samples_split), e il numero minimo di campioni richiesti per formare una foglia (min_samples_leaf).
Per mitigare eventuali problemi di overfitting e ottimizzare l'efficienza del processo di addestramento, viene implementato l'early stopping. Questa strategia permette di interrompere l'addestramento se non si osservavano miglioramenti dell'accuratezza del modello entro un numero specificato di iterazioni.
Il modello migliore ottenuto dalla Grid Search è stato quindi selezionato e valutato su ciascuna fold della cross-validation. In ogni iterazione, l'accuratezza del Random Forest è stata monitorata sia sul set di addestramento che su quello di validazione, interrompendo il processo anticipatamente se non si osservavano miglioramenti per un numero massimo di iterazioni prestabilito. Questa strategia mira a ottenere un modello generalizzato, evitando l'overfitting, e garantendo al contempo un'accuratezza adeguata nella classificazione dei dati.

- Definizione del Modello Random Forest e Parametri per la Grid Search: È stato istanziato un modello di Random Forest Classifier (RandomForestClassifier) con un numero iniziale di alberi e sono stati definiti i parametri da esplorare durante la Grid Search. Questi parametri includono il numero di alberi nell'ensemble (n_estimators), la massima profondità di ciascun albero (max_depth), il numero minimo di campioni richiesti per suddividere un nodo interno (min_samples_split), e il numero minimo di campioni richiesti per formare una foglia (min_samples_leaf).

- Configurazione della Grid Search e Addestramento del Modello Random Forest: È stata configurata una procedura di Grid Search (GridSearchCV) per esplorare tutte le possibili combinazioni di parametri per il modello Random Forest. Successivamente, il modello è stato addestrato sul set di dati di addestramento utilizzando la combinazione ottimale di iperparametri identificata dalla Grid Search.

- Valutazione del Modello con Early Stopping: Il modello addestrato è stato valutato mediante una cross-validation stratificata con uno schema di K-fold, dove K è pari a 5. Durante ogni iterazione, sono state registrate le previsioni sia sul set di addestramento che su quello di validazione. Inoltre, è stato implementato un

meccanismo di early stopping per interrompere l'addestramento se non si osservava alcun miglioramento dell'accuratezza entro un numero massimo di iterazioni specificato.

- Calcolo e Stampa delle Metriche: Al termine delle iterazioni, sono state calcolate e stampate diverse metriche di valutazione del modello Random Forest, incluse la matrice di confusione, la precisione, il richiamo e l'F1 score, sia per il set di addestramento che per quello di validazione. Questo processo fornisce una valutazione completa delle prestazioni del modello Random Forest sulla base dei dati di addestramento e di validazione.

ACCURACY RAGGIUNTA CON CROSS VALIDATION E EARLY STOPPING

```
Fold 1: Accuratezza Random Forest = 0.9308300395256917
Fold 2: Accuratezza Random Forest = 0.9525691699604744
Fold 3: Accuratezza Random Forest = 0.9405940594059405
Fold 4: Accuratezza Random Forest = 0.9227722772277228

Early stopping Random Forest! Miglior accuratezza raggiunta: 0.9525691699604744
```

TRAINING SET

```
Matrice di Confusione Random Forest (Set di Addestramento):
[[1152    0    0    0    0    3    0]
 [   0 1155    0    0    0    0    0]
 [   0    0 1154    0    1    0    0]
 [   0    0    0 1148    0    5    2]
 [   1    0    1    6 1141    0    7]
 [   0    0    0    3    0 1150    2]
 [   0    0    0    3    3    1 1148]]

Precision Random Forest (Set di Addestramento): 0.995310485605556
Recall Random Forest (Set di Addestramento): 0.9953005194162751
F1 Score Random Forest (Set di Addestramento): 0.9953006465440095
```

VALIDATION SET

```
Matrice di Confusione Random Forest (Set di Validazione):
[[268   0  10   0   0   6   5]
 [  0 289   0   0   0   0   0]
 [ 17   0 266   0   5   0   1]
 [  0   0   0 263   2   2  22]
 [  3   0   6   2 271   0   6]
 [  1   0   0   2   0 278   8]
 [  4   0   1  20   3   2 259]]

Precision Random Forest (Set di Validazione): 0.9372918949433452
Recall Random Forest (Set di Validazione): 0.9366963402571711
F1 Score Random Forest (Set di Validazione): 0.9368988154345289
```

3. VALUTAZIONE DEI RISULTATI

Le metriche utilizzate per valutare le prestazioni del modello sui dati di test forniscono una panoramica completa delle capacità predittive del modello, in quanto prendono in input dati mai visti durante l'addestramento e quindi indicano la capacità di generalizzazione del modello stesso.
Le metriche utilizzate nel considerare la bontà dei modelli sono:

Matrice di Confusione
La matrice di confusione mostra il numero di veri positivi (TP), falsi positivi (FP), veri negativi (TN) e falsi negativi (FN). Questo fornisce una comprensione dettagliata delle performance del modello per ciascuna classe e di quali sono gli esempi che non vengono classificati nel giusto modo.
Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali.
La matrice viene implementata inserendo sulle colonne le classi effettive e sulle righe le classi predette.
Da questa metrica, possono essere ricavate tutte le altre.

Precision
La precision è la proporzione di istanze correttamente predette come appartenenti a una determinata classe rispetto al totale delle istanze predette come appartenenti a quella classe. È utile quando l'obiettivo è ridurre i falsi positivi.
Precision: TP / (TP+FP)

Recall (Sensibilità o True Positive Rate)
Il recall è la proporzione di istanze correttamente predette come appartenenti a una determinata classe rispetto al totale delle istanze effettivamente appartenenti a quella classe. È utile quando l'obiettivo è ridurre i falsi negativi.
Recall: TP / (TP+FN)

F1 Score
L'F1 Score è la media armonica tra precision e recall. Questa metrica tiene conto sia dei falsi positivi che dei falsi negativi, offrendo un bilanciamento tra precision e recall. È particolarmente utile quando le classi sono sbilanciate.
F1 Score: 2 × ((Precision+Recall) / (Precision×Recall))

Accuracy

L'accuracy rappresenta la proporzione di istanze correttamente classificate rispetto al totale delle istanze. Misura la correttezza complessiva del modello. Va utilizzata con cautela quando le classi sono sbilanciate, in quanto non è veritiera.
Accuracy: TP / (TP+FP+TN+FN)

DECISION TREE

```
Matrice di Confusione Decision Tree (Set di Test):
[[339   0  38   0   1   3  14]
 [  0 161   0   0   0   0   0]
 [ 25   0 435   0  14   2   3]
 [  0   0   0 933   5  15  90]
 [  4   0  19   7 542   0  16]
 [  9   0   5  28   0 559  18]
 [  8   0   4  73  13  22 679]]

Precision Decision Tree (Set di Test): 0.8940016872580014
Recall Decision Tree (Set di Test): 0.8932419196865817
F1 Score Decision Tree (Set di Test): 0.8934806472103902
Accuracy Decision Tree (Set di Test): 0.8932419196865817
```

KNN

```
Matrice di Confusione KNN (Set di Test):
[[361   0  23   0   1   2   8]
 [  0 161   0   0   0   0   0]
 [ 17   0 451   0   6   2   3]
 [  0   0   0 922   1  27  93]
 [  3   0  11   1 557   0  16]
 [  6   0   0   8   0 589  16]
 [  2   0   1  59   9  10 718]]

Precision KNN (Set di Test): 0.92176588042365
Recall KNN (Set di Test): 0.9204211557296768
F1 Score KNN (Set di Test): 0.9207054966351643
Accuracy Decision Tree (Set di Test): 0.9204211557296768
```

SVM

```
Matrice di Confusione SVM (Set di Test):
[[363   0  23   0   1   2   6]
 [  0 161   0   0   0   0   0]
 [ 12   0 454   0   7   2   4]
 [  0   0   0 944   3  19  77]
 [  2   0  11   4 560   0  11]
 [  7   0   0  15   0 584  13]
 [  4   0   1  47   9   9 729]]

Precision SVM (Set di Test): 0.9299937391694213
Recall SVM (Set di Test): 0.9292360430950048
F1 Score SVM (Set di Test): 0.9294189863574752
Accuracy Decision Tree (Set di Test): 0.9292360430950048
```

RANDOM FOREST

```
Matrice di Confusione Random Forest (Set di Test):
[[358   0  25   0   1   2   9]
 [  0 161   0   0   0   0   0]
 [ 22   0 448   0   6   2   1]
 [  0   0   0 922   4  18  99]
 [  4   0   9   3 559   0  13]
 [ 11   0   0  18   0 574  16]
 [  5   0   4  43  11  15 721]]

Precision Random Forest (Set di Test): 0.9180367608309183
Recall Random Forest (Set di Test): 0.9165034280117532
F1 Score Random Forest (Set di Test): 0.9168435548529469
Accuracy Decision Tree (Set di Test): 0.9165034280117532
```

Tra i modelli, quelli addestrati con gli algoritmi del KNN e il SVM, sono quelli che riescono a performare meglio e pertanto possono essere considerati i migliori con un'accuratezza superiore al 92%.