Raccomandazione

Agricola per la semina

Autore: Matteo Buongiorno

m.buongiorno13@studenti.uniba.it

Link progetto:

sch3rry/Icon_24-25

Indice

1. Introduzione

Il presente lavoro ha l'obiettivo di individuare la coltura agricola più adatta sulla base di specifici parametri ambientali e agronomici, fornendo uno strumento di supporto alle decisioni per migliorare la pianificazione e ottimizzare la resa.

In questo progetto si affrontano i seguenti argomenti:

- Creazione di un'ontologia
- KB, base di conoscenza effettuata con prolog
- Utilizzo di modelli per apprendimento supervisionato

1.1 Strumenti

Il progetto è stato sviluppato interamente in linguaggio **Python**, avvalendosi di librerie esterne per la rappresentazione grafica dei dati e per l'importazione di modelli preesistenti disponibili in raccolte come *scikit-learn*. La piattaforma **GitHub** è stata utilizzata come sistema di repository ufficiale per la gestione e la condivisione del progetto.

1.1.1 Librerie

Per lo sviluppo del progetto sono state adottate le seguenti librerie **Python**, selezionate in funzione di obiettivi specifici legati all'analisi, alla modellazione e alla presentazione dei dati:

- **Pandas:** scelta per la sua robustezza nella gestione di dataset complessi e per le numerose funzioni di **data cleaning** e trasformazione, fondamentali per garantire la qualità dell'analisi.
- **Numpy:** adottata per le prestazioni elevate nelle operazioni su array e strutture numeriche.
- **Sklearn:** libreria utilizzata per importazione di modelli ed apprendimento.
- **Matplotlib:** utilizzata per la sua versatilità nella creazione di grafici personalizzati e nella produzione di visualizzazioni statiche adatte a report e documentazione.
- **Pyswip:** scelta per abilitare l'integrazione tra Python e **Prolog.**
- **Seaborn:** adottata per migliorare l'estetica e la leggibilità dei grafici statistici, in particolare per la creazione di **heatmaps** utili nell'analisi delle correlazioni.
- **Owlready2:** selezionata per la possibilità di consultare e manipolare ontologie direttamente in Python.
- **Pickle:** scelta per la sua semplicità ed efficienza nel salvare e caricare oggetti Python in formato binario, consentendo la persistenza dei modelli e delle configurazioni.

1.1.2 Software

Oltre alle librerie Python, lo sviluppo del progetto ha previsto l'impiego di strumenti software scelti per garantire efficienza nello sviluppo e precisione nella gestione della conoscenza:

- **PyCharm**: IDE adottato per la scrittura e la gestione del codice Python.
- **Protege**: piattaforma utilizzata per la creazione, modifica e consultazione di ontologie, selezionata per la sua compatibilità con standard semantici (come OWL).

2. Dataset

Il dataset impiegato per l'addestramento e la validazione dei modelli proviene dal portale **Kaggle** ed è composto da variabili misurative e descrittive finalizzate alla previsione della coltura ottimale.

Variabili (Features):

- **N**: Rateo di Azoto contenuto nel terreno
- **P**: Rateo Fosforo contenuto nel terreno
- **K**: Rateo di Potassio contenuto nel terreno
- **Temperature**: Temperatura in gradi Celsius
- **Humidity**: Umidità in percentuale
- **Ph**: Valore di ph del Terreno
- **Rainfall**: pioggia misurata in mm

Etichetta(Label):

- **Label**: Nome della coltura, 22 colture diverse

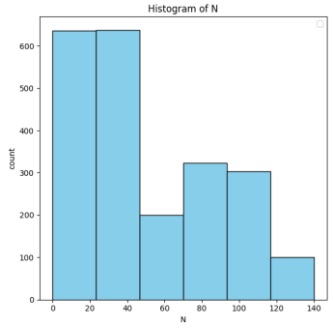
2.1 Data Analisis

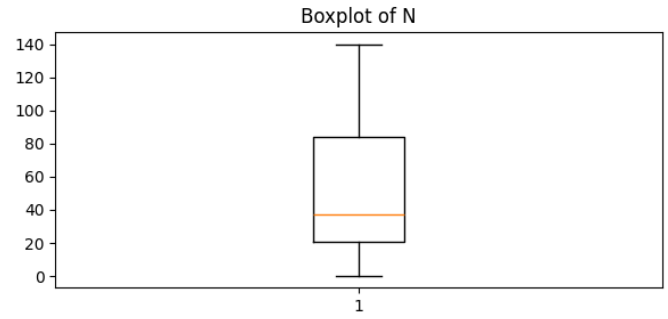
In fase preliminare è stata condotta un'analisi esplorativa volta a:

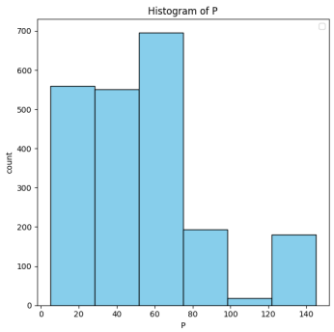
- **Valutare la dimensione complessiva del dataset,** verificando il numero di record e di variabili disponibili.
- **Individuare la presenza di valori nulli o mancanti,** al fine di stimare l'eventuale necessità di operazioni di data cleaning.
- **Esaminare la distribuzione di Ciascuna variabile** mediante istogrammi e grafici di densità, per comprendere la natura dei dati e rilevare eventuali anomalie.

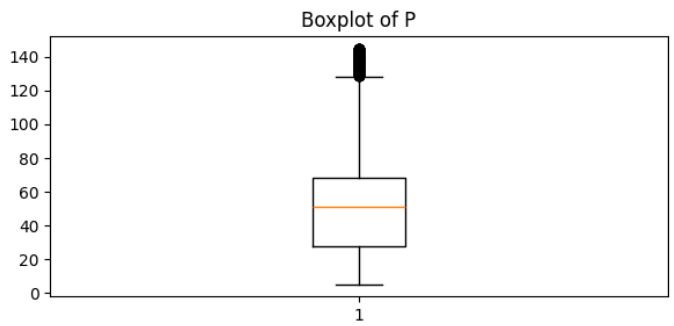
- **Identificare possibili ouliers** tramite l'utilizzo di boxplot, con l'obiettivo di valutare l'impatto di valori anomali sulle fasi successive di modellazione.

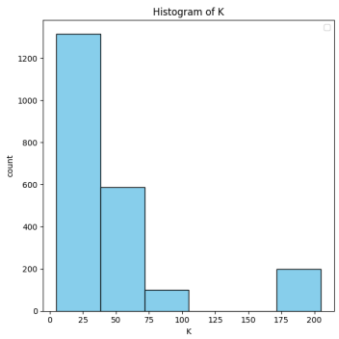
Le distribuzioni sono le seguenti:

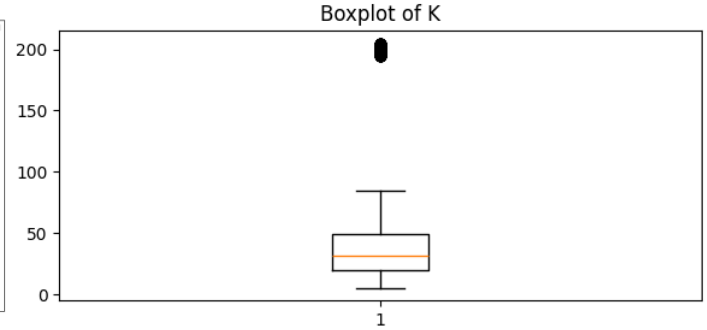


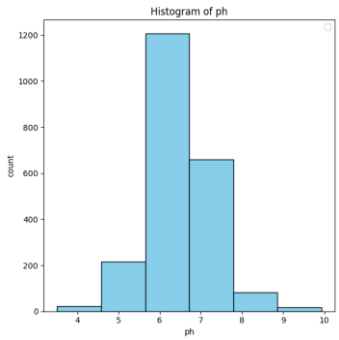


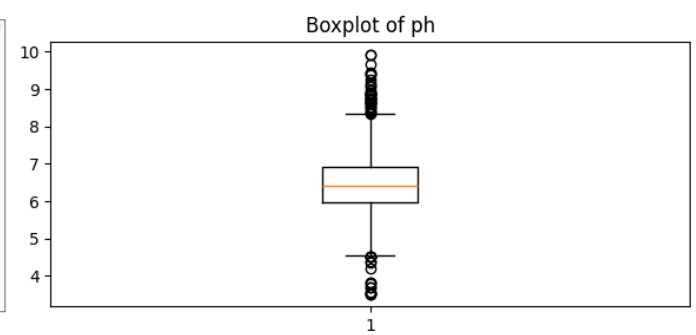


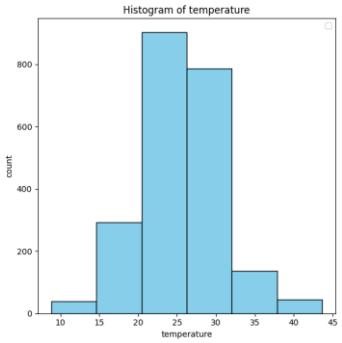


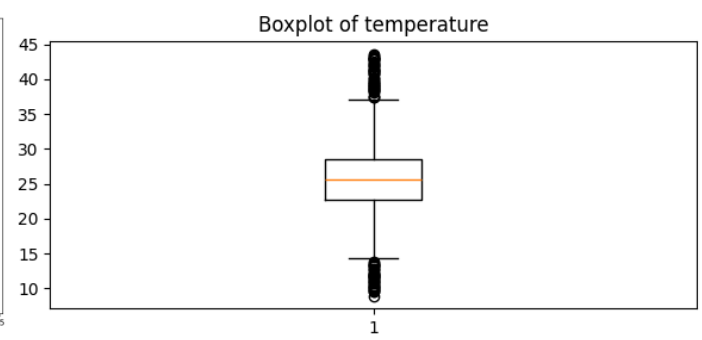


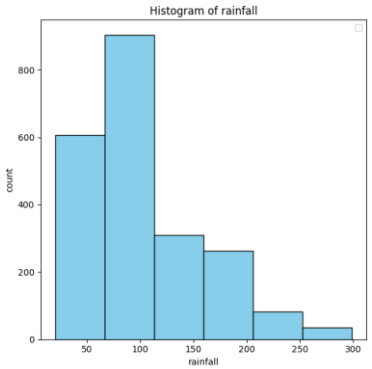


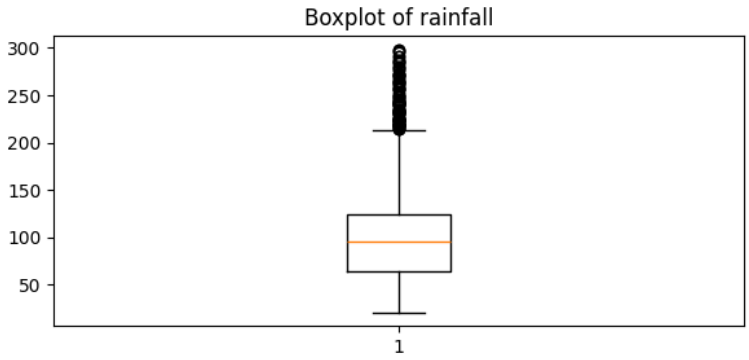


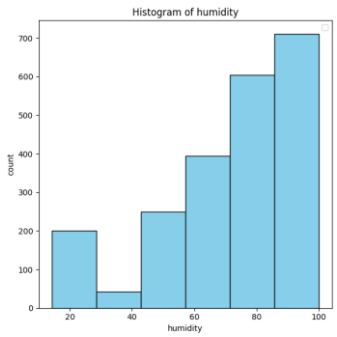


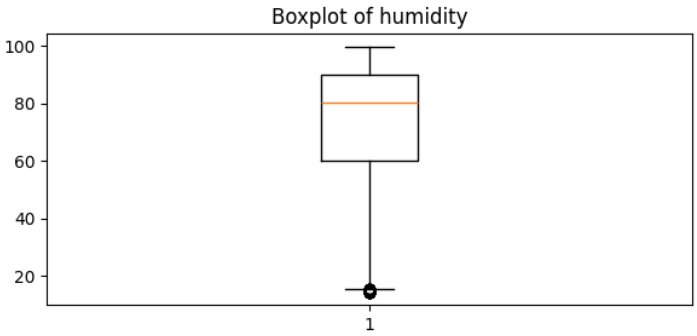










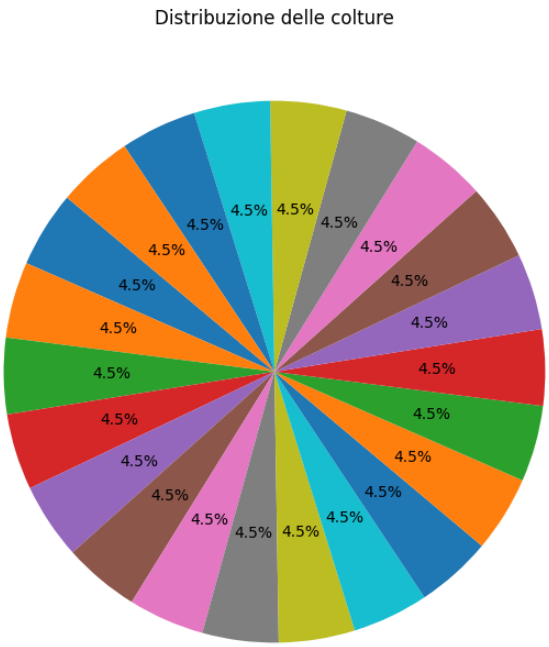


Dall'analisi grafica emerge che la maggior parte delle variabili presenta distribuzioni non perfettamente gaussiane, ad eccezione di **pH** e **Temperature**, le quali mostrano un andamento più regolare e simile a una curva normale.

Un caso particolare è rappresentato dalla variabile **K (Potassio):**

- L'istogramma evidenzia un'assenza totale di valori nell'intervallo compreso tra 100 e 175.
- Il **boxplot** segnala la presenza di diversi **outlier** nella parte superiore della distribuzione, suggerendo campioni con concentrazioni di potassio anormalmente elevate rispetto alla maggioranza dei dati.

Per valutare la distribuzione della variabile **Label** (riferita alle 22 tipologie di colture presenti nel dataset), è stato realizzato un grafico a torta.



L’analisi conferma che tutte le classi risultano perfettamente bilanciate, con una distribuzione uniforme tra le categorie.

La rimozione degli outlier non è stata effettuata al fine di preservare il bilanciamento originario delle classi e mantenere la rappresentatività di scenari rari ma potenzialmente significativi.

Per valutare le relazioni lineari tra le diverse **features** del dataset, è stata realizzata una **heatmap** della matrice di correlazione.

Osservandola possiamo notare una forte relazione tra Fosforo e potassio, La correlazione positiva elevata indica che terreni con un’alta concentrazione di fosforo tendono, in media, ad avere anche un’alta concentrazione di potassio.

Notiamo anche che la relazione tra Azoto e Fosforo è una correlazione negativa, questa correlazione segnala che, al crescere dell’azoto nel terreno, il fosforo tende a diminuire e viceversa.

Il dataset utilizzato ai fini del progetto è composto da **2200 record** e **8 variabili** (colonne), comprendenti sia **features** che **label** di classificazione. L’analisi preliminare ha confermato l’**assenza di valori nulli o mancanti** in tutte le colonne, garantendo un’elevata qualità e completezza del campione di dati.

```
(2200, 8)
N              0
P              0
K              0
ph             0
temperature    0
rainfall       0
humidity       0
label          0
dtype: int64
```

2.2 Data Preprocessing

In questa fase, il dataset è stato sottoposto a operazioni di **preprocessing** finalizzate a renderlo idoneo all'addestramento dei modelli.

1. **Codifica delle Variabili**:
   - **LabelEncoder**: per la variabile Label, è stato utilizzato **LabelEncoder** al fine di assegnare a ciascuna classe un codice numerico univoco.
   - **Scaling delle feature**: Le variabili numeriche sono state normalizzate tramite **StandardScaler**, scelta per la sua efficacia con i modelli adottati.

2. Gestione degli outlier:

   Poiché la loro rimozione avrebbe comportato uno sbilanciamento delle classi, si è deciso di mantenerli, decidendo poi in futuro se eliminarli tramite il metodo IQR se i modelli presenteranno prestazioni non adatte

3. Assegnazione delle variabili

   Definizioni delle matrici **X** (Features) e **Y** (Label) per l'addestramento.

4. Suddivisione del dataset
   Il dataset è stato suddiviso in **Train Set** (80%) e **Test Set** (20%).

   Lo **scaler** è stato calcolato esclusivamente sui dati di training e poi applicato agli altri set, per evitare fenomeni di Data Leakage-

5. Salvataggio del dataset

   I Dati preprocessati sono stati salvati in formato .pkl traimite il modulo Pickle, per un utilizzo efficiente nelle fasi successive di sviluppo e testing.

3. Ontologia

Un'ontologia rappresenta una **struttura formale e organizzata della conoscenza** relativa al dominio considerato, definendo concetti, relazioni e proprietà logiche tra gli elementi rilevanti.

L'analisi del dominio ha permesso di identificare le entità fondamentali e le loro caratteristiche, con l'obiettivo di rappresentare in modo chiaro i fattori che influenzano la coltivazione delle colture.

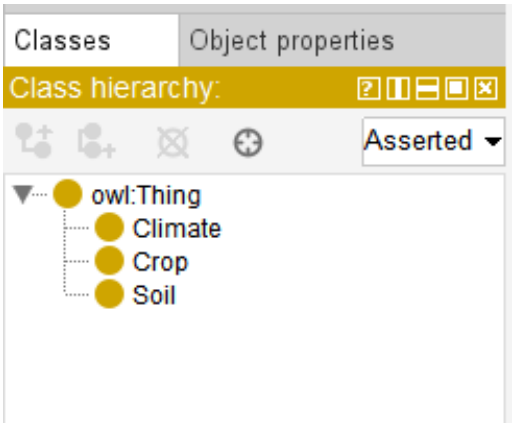
3.1 Scelte Progettuali

Per la creazione dell'ontologia è stato impiegato il software **Protégé**, selezionato per la sua capacità di offrire una modellazione intuitiva, leggibile e conforme agli standard semantici (es. OWL).

3.1.1 Classi

L'ontologia è strutturata attorno a tre classi principali:

- **Crop:** Rappresenta la coltura.
- **Climate:** Include i fattori climatici che possono influenzare la coltura
- **Soil:** Comprende i fattori legati alla tipologia e qualità del terreno

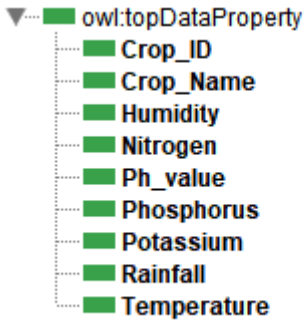


3.1.2 Data Property

Per ogni classe sono state associate delle proprietà:

- **Crop**
  - Crop_ID: Identificatore univoco della coltura.
  - Name: Nome della coltura.
- **Climate**
  - Temperature: indica la temperatura favorevole per la coltura.
  - Humidity: indica l'umidità favorevole per la crescita della coltura.

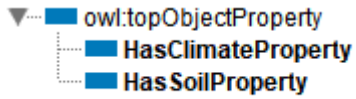
    - Rainfall: indica la quantità di pioggia in mm favorevole per la crescita della coltura.
- **Soil**:
    - Ph_value: acidità o basicità del terreno in base al valore del ph.
    - N: Quantità di Azoto presente nel terreno.
    - P: Quantità di Fosforo presente nel terreno.
    - K: Quantità di Potassio presente nel terreno.



3.1.3 Object Property

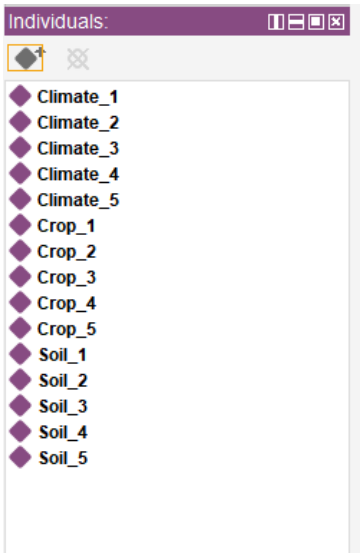
Le object property definiscono le relazioni tra le classi:

- **HasClimateProperty**: relaziona la classe Crop con la classe Climate.
- **HasSoilPropert**: relaziona la classe Crop con con la classe Soil.



3.1.4 Individui

Per ogni classe sono stati definiti 5 individui di esempio, utili a rappresentare istanze reali di colture, fattori climatici e tipologie di suolo.



3.2 Owlready

La seguente libreria consente la consultazione in ambiente Python di ontologie ed interrogarla attraverso delle query

```
--- MENU ONTOLOGIA ---
1. Lista classi
2. Lista data property
3. Lista object property
4. Lista di individui
5. Filtra in base ad un valore terreno (azoto, ph, potassio, fosforo)
6. Filtra in base al clima (umidità, pioggia ,temperatura)
0. Torna indietro
Scelta:
```

Menù principale

```
Scelta: 1
['Crop', 'Climate', 'Soil']
```

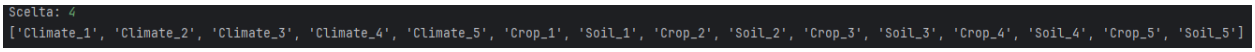
Lista classi

```
Scelta: 2
['Crop_ID', 'Crop_Name', 'Humidity', 'Nitrogen', 'Ph_value', 'Phosphorus', 'Potassium', 'Rainfall', 'Temperature', 'HasClimateProperty', 'HasSoilProperty']
```

Lista Data Property

```
Scelta: 3
['HasClimateProperty', 'HasSoilProperty']
```

Lista Object Property



Lista individui

```
Scelta: 5
Inserisci fattore terreno (Nitrogen, Ph_value, Potassium, Phosphorus): Nitrogen
Valore minimo: 10
Valore massimo: 100
[90.0]
[71.0]
[40.0]
[13.0]
[3.0]
Individuo: Crop_1, Nitrogen = 90.0
Individuo: Crop_2, Nitrogen = 71.0
Individuo: Crop_3, Nitrogen = 40.0
Individuo: Crop_4, Nitrogen = 13.0
```

Lista individui in base ad un fattore terreno

```
Scelta: 6
Inserisci fattore climatico (temperature, humidity, rainfall): Temperature
Valore minimo: 10
Valore massimo: 100
Crop_1
Crop_2
Crop_3
Crop_4
Crop_5
Individuo: Crop_1, Temperature = 21.0
Individuo: Crop_2, Temperature = 22.6
Individuo: Crop_3, Temperature = 17.0
Individuo: Crop_4, Temperature = 17.13
```

Lista individui in base ad un fattore climatico

4. Modelli e Apprendimento

L'attività di modellazione è stata condotta adottando un approccio di **apprendimento supervisionato**, un approccio automatico che si serve di dati in input dette '**Features**', e consente un output delle '**Labels**', Il dataset è stato suddiviso in:

- **Train set**: insieme di elementi del dataset utilizzato in ambito di training per il modello per apprendere meglio i pattern per l'output.

- **Test set**: insieme di elementi appartenenti al dataset utilizzati come set di testing per verificare l'accuratezza del modello.

4.1 Scelte Progettuali

Sono stati selezionati 4 modelli così da consentire un confronto ed eventualmente e identificare l'algoritmo più performante per il dataset in uso.

I modelli scelti sono:

- **Random Forest**: Modello ensemble costituito da molteplici alberi decisionali addestrati su differenti sottocampioni di dati e feature. È stato selezionato per la sua robustezza, la capacità di ridurre l'overfitting e le buone prestazioni anche in presenza di variabili rumorose.

- **K-Nearest Neighbor**: Algoritmo basato sulla vicinanza tra istanze: la classe di un nuovo campione viene determinata in base alla maggioranza tra i *k* vicini più prossimi. È stato scelto per la sua semplicità, interpretabilità e per la capacità di adattarsi a distribuzioni di dati non lineari.

- **Support Vector Classifier**: Classificatore che ricerca l'iperpiano ottimale in grado di massimizzare il margine tra le classi. È stato incluso per la sua efficacia in spazi ad alta dimensionalità e la possibilità di gestire dati non linearmente separabili tramite l'uso di funzioni kernel.

- **Logistic Regression**: modello lineare che predice la probabilità di appartenenza ad una classe tramite la funzione logistica. Essendo un modello lineare consente di valutare se le colture si distinguono già con confini semplici tra le feature.

Per ogni algoritmo selezionato è stata condotta una fase di **tuning**, con l'obiettivo di individuare la configurazione ottimale in relazione alle caratteristiche specifiche del dataset. Questa fase ha permesso di migliorare le prestazioni dei modelli rispetto ai parametri predefiniti, riducendo il rischio di overfitting o underfitting.

Successivamente, le performance sono state valutate tramite:

- **Matrice di Confusione:** per analizzare il numero di classificazioni corrette ed errate.
- **Classification Report:** Per ottenere metriche dettagliate quali Precision, Recall, F1-score e support per ciascuna classe.

Per verificare la stabilità e generalizzabilità dei modelli è stata utilizzata la tecnica **Stratified K-Fold Cross Validation**, suddividendo il dataset in K partizioni mantenendo

il bilanciamento delle classi in ciascun fold. La quantità di partizioni scelte è 5, un numero di suddivisioni molto usato per ottenere un **equilibrio tra accuratezza e tempo di calcolo**.

4.2 Tuning

Per tuning si intende il processo di ottimizzazione dei parametri di un modello alla fine di migliorare le prestazioni sul dataset in uso.

In questa fase, per ogni algoritmo selezionato è stata costruita una griglia di possibili valori per ciascun iperparametro rilevante, definendo quindi un insieme di configurazioni candidate.

La ricerca della combinazione ottimale è stata effettuata mediante **Grid Search Cross Validation**.

4.2.1 Random Forest

Per il modello **Random Forest** sono stati presi in considerazione i seguenti iperparametri:

- **N_estimators**: numero di alberi nella foresta, un numero maggiore può aumentare la stabilità e accuratezza del modello, a scapito del tempo di addestramento.
- **Max_depth**: profondità massima consentita per ciascun albero, controlla la complessità del modello.
- **Min_samples_split**: numero di campioni richiesti per suddividere un nodo, valori più alti corrispondono ad alberi meno profondi, valori più bassi ad alberi più complessi e dettagliati.
- **Max_features**: numero massimo di feature da considerare ad ogni split, con meno feature per split abbiamo maggiore diversità tra alberi e quindi meno overfitting.
- **Bootstrap**: aumenta la diversità degli alberi migliorando la robustezza del modello.

Griglia di ricerca degli iperparametri

```
rf_params = {
    "n_estimators": [100, 200, 400, 800, 1200],
    "max_depth": [None, 10, 20, 30, 50],
    "min_samples_split": [2, 5, 8, 10],
    "min_samples_leaf": [1, 2, 4],
    "max_features": ['sqrt', 'log2', None],
    "bootstrap": [True, False]
}
```

Il risultato ottenuto dopo Grid Search è il seguente

#best parameters: {'n_estimators': 50, 'max_depth': None, 'min_samples_split': 2, 'min_samples_leaf': 2}

4.2.2 K-Nearest Kneighbor

Per il classificatore **KNN** sono stati considerati i seguenti iperparametri nella fase di ottimizzazione:

- **N_neighbors**: numero di vicini da considerare per fare la classificazione.
- **Weights**: Strategia di ponderazione dei vicini (*uniform* assegna lo stesso peso a tutti, *distance* attribuisce maggior peso ai più vicini).
- **Algorithm**: Metodo di ricerca dei vicini.
- **Leaf_size**: dimensione delle foglie usate in ball_tree e kd_tree, influisce sulla velocità di costruzione dell'albero e sulla memoria, può migliorare le prestazioni.
- **Metric**: Distanza utilizzata per calcolare la vicinanza.

Griglia di ricerca degli iperparametri

```
Knn_params = {
    'n_neighbors': list(range(1, 11)),
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size': [16, 32, 64],
    "metric": ["euclidean", "manhattan", "minkowski"]
}
```

Configurazione ottimale

```
knn = KNeighborsClassifier(n_neighbors=6, weights='distance', algorithm='auto', leaf_size=16, metric='manhattan')
```

4.2.3 Support Vector Classifier

I parametri presi in considerazione per **SVM** sono i seguenti:

- **Kernel**: funzione utilizzata per trasformare lo spazio delle feature al fine di ottenere una separazione ottimale tra le classi.
- **C**: parametro di regolarizzazione che controlla la penalizzazione degli errori di classificazione in fase di training. Valori bassi consentono più errori, generando un iperpiano più semplice; valori elevati riducono la tolleranza agli errori, aumentando la complessità del modello.
- **Gamma**: parametro per i kernel RBF, Poly e Sigmoid; controlla quanto influisce ogni singolo punto di addestramento.
- **Degree**: usato solo con kernel polinomiale 'poly', indica il grado del polinomio, più è alto il valore più alta sarà la complessità e il tempo di calcolo.

Griglia di ricerca degli iperparametri

```
svc_params={
    'kernel':['rbf','linear', 'poly', 'sigmoid'],
    'C':[0.1, 2, 5, 10],
    'gamma':[0.001, 0.01, 0.1, 1.0],
    'degree':[2,3,4,5]

}
```

Configurazione ottimale

```
#Best Parameters: {'C': 5, 'degree': 2, 'gamma': 0.1, 'kernel': 'rbf'}
```

4.2.4 Logistic Regression

Per il modello di **Regressione Logistica** sono stati presi in considerazione i seguenti iperparametri:

- **Penalty**: indica il tipo di regolarizzazione applicata, per controllare la complessità del modello.
- **Solver**: algoritmo utilizzato per ottimizzare la funzione di costo.
- **Multi_class**: definisce come gestire la classificazione multi classe.
- **C**: inverso della forza di regolarizzazione; valori piccoli forte regolarizzazione, modello più semplice, valori alti modello più complesso.

Griglia di ricerca degli iperparametri

```
log_param = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'solver': ['liblinear', 'sag', 'saga'],
    'multi_class': ['ovr', 'cramming', 'multinomial'],
    'C': np.logspace(-3, stop: 3, num: 20)
}
```

Configurazione Ottimale

```
#{'C': np.float64(215.44346900318823), 'l1_ratio': 0, 'multi_class': 'multinomial', 'penalty': 'elasticnet', 'solver': 'saga'}
```

4.3 Metodi di valutazione

Per valutare le prestazioni dei modelli addestrati sono state utilizzate diverse metriche e strumenti.

Classification Report:

Metodo di valutazione più utilizzato, fornisce una sintesi statistica delle prestazioni del modello per ciascuna classe, includendo:

- **Precision**: percentuale di predizioni corrette rispetto a tutte le predizioni effettuate per quella classe.

- **Recall**: percentuale di istanze di una classe correttamente identificate rispetto al totale delle istanze reali.
- **F1-score**: media armonica di precision e recall, utile per bilanciare le metriche.
- **Support**: numero di istanze reali presenti per ciascuna classe.

Matrice di confusione:

Rappresentazione tabellare che mostra il numero di predizioni corrette e incorrette suddivise per classe.

Ogni riga della matrice rappresenta le istanze reali di una classe, mentre ogni colonna rappresenta le predizioni del modello.

È uno strumento chiave per individuare pattern di errore e potenziali ambiguità tra classi.

Cross Validation:

È stata adottata la tecnica **Stratified K-Fold**, che suddivide il dataset in *k* partizioni mantenendo il bilanciamento delle classi in ciascun fold.

Questo approccio riduce il rischio di overfitting e fornisce una valutazione più affidabile, poiché ogni istanza viene utilizzata sia per l'addestramento che per il test in fasi diverse. Sono stati calcolati **media**, **deviazione standard** e **varianza** delle metriche di performance per stimare la stabilità del modello, quali:

- **Accuracy**
- **Precision**
- **Recall**
- **F1- score**

Per tutte le metriche esclusa l'accuracy è stata presa in considerazione sia la componente weighted che macro:

- **Macro**: ogni classe viene considerata con peso equo
- **Weighted**: le classi hanno un peso proporzionale alla frequenza nel set

Learning Curve:

Sono stati generati grafici delle curve di apprendimento e di errore per ciascun modello, al fine di valutare:

- l'andamento delle performance di training e validazione al crescere del numero di esempi utilizzati;
- la presenza di **overfitting** (prestazioni elevate sul training ma inferiori sulla validazione) o di **underfitting** (prestazioni basse su entrambi).

4.4 Risultati

Ora riportiamo i risultati ottenuti e confrontiamo i vari modelli

4.4.1 Random Forest

L'analisi tramite **Cross Validation** ha prodotto i seguenti valori:

```
- accuracy: mean=0.9949, std=0.0021, var=0.000005
- precision: mean=0.9941, std=0.0023, var=0.000005
- recall: mean=0.9927, std=0.0024, var=0.000006
- f1_macro: mean=0.9948, std=0.0020, var=0.000004
- f1_weighted: mean=0.9932, std=0.0014, var=0.000002
- precision_weighted: mean=0.9962, std=0.0014, var=0.000002
- recall_weighted: mean=0.9949, std=0.0021, var=0.000005
```

Abbiamo una media molto alta, pari a 0,99, e deviazioni standard davvero basse. Questo indica che il modello funziona bene, classificando correttamente quasi tutti i campioni. Valori così stabili nelle deviazioni standard mostrano anche che il modello è affidabile e mantiene prestazioni costanti.

Classification Report:

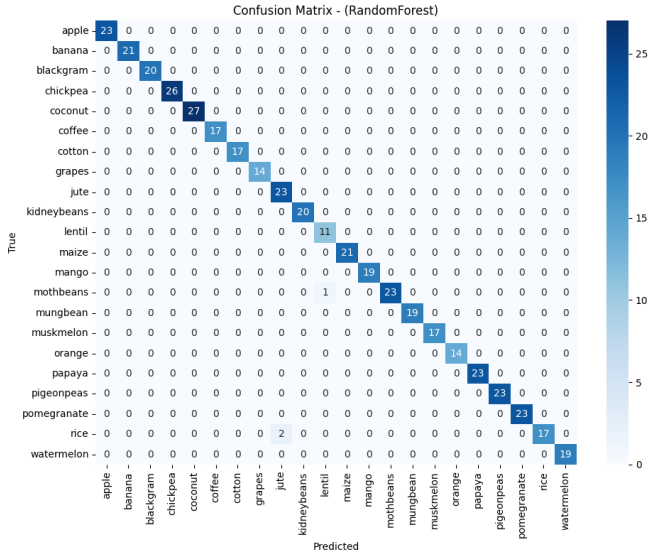
```
Classification report (test):
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        23
      banana       1.00      1.00      1.00        21
   blackgram       1.00      1.00      1.00        20
    chickpea       1.00      1.00      1.00        26
     coconut       1.00      1.00      1.00        27
      coffee       1.00      1.00      1.00        17
      cotton       1.00      1.00      1.00        17
      grapes       1.00      1.00      1.00        14
        jute       0.92      1.00      0.96        23
 kidneybeans       1.00      1.00      1.00        20
      lentil       0.92      1.00      0.96        11
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        19
   mothbeans       1.00      0.96      0.98        24
    mungbean       1.00      1.00      1.00        19
   muskmelon       1.00      1.00      1.00        17
      orange       1.00      1.00      1.00        14
      papaya       1.00      1.00      1.00        23
  pigeonpeas       1.00      1.00      1.00        23
 pomegranate       1.00      1.00      1.00        23
        rice       1.00      0.89      0.94        19
  watermelon       1.00      1.00      1.00        19

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

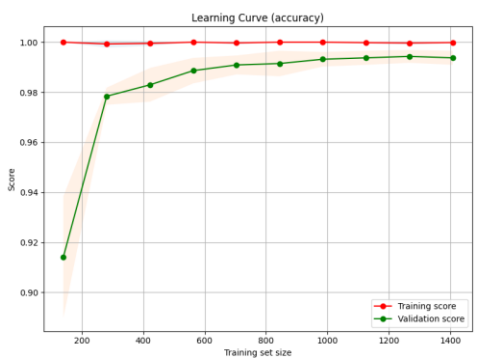
Il modello mostra prestazioni eccellenti, con una classificazione perfetta nella maggior parte delle classi. Solo alcune classi presentano valori compresi tra 0,89 e 0,98, che restano comunque molto buoni. L'accuratezza complessiva raggiunge il 99%, indice di prestazioni praticamente perfette.

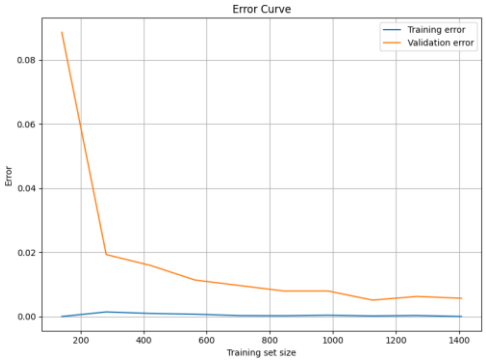
Confusion Matrix:



I valori lungo la diagonale principale risultano molto elevati, confermando ancora una volta l'accuratezza e l'affidabilità del modello. Si rilevano soltanto pochi errori isolati nelle classi *mungbean* e *muskmelon*.

Learning Curves:





Le curve di apprendimento evidenziano chiaramente l'elevata capacità predittiva del modello. La curva dell'accuratezza mostra come il punteggio sul training set rimanga costantemente vicino al 100%, mentre quello sul validation set cresce rapidamente con l'aumentare dei dati, stabilizzandosi attorno al 99%. Questo comportamento indica una buona capacità di generalizzazione, senza segni di overfitting. La seconda curva conferma pienamente questa lettura. In sintesi, le due curve dimostrano che il modello Random Forest è robusto, accurato e ben generalizzato, con performance eccellenti sia in addestramento che in validazione.

4.4.2 KNN

Cross Validation:

```
- accuracy: mean=0.9807, std=0.0068, var=0.000046
- precision: mean=0.9830, std=0.0056, var=0.000032
- recall: mean=0.9808, std=0.0064, var=0.000041
- f1_macro: mean=0.9807, std=0.0066, var=0.000044
- f1_weighted: mean=0.9806, std=0.0068, var=0.000046
- precision_weighted: mean=0.9830, std=0.0056, var=0.000031
- recall_weighted: mean=0.9807, std=0.0068, var=0.000046
```

Il modello KNN evidenzia buone prestazioni, con un'accuratezza media pari al 98,07%. Le metriche di precisione, recall e F1-score si mantengono su valori molto vicini, confermando l'efficacia generale del modello nel classificare correttamente i dati. Rispetto al Random Forest, si osserva però una lieve riduzione delle performance e una maggiore variabilità nei risultati.

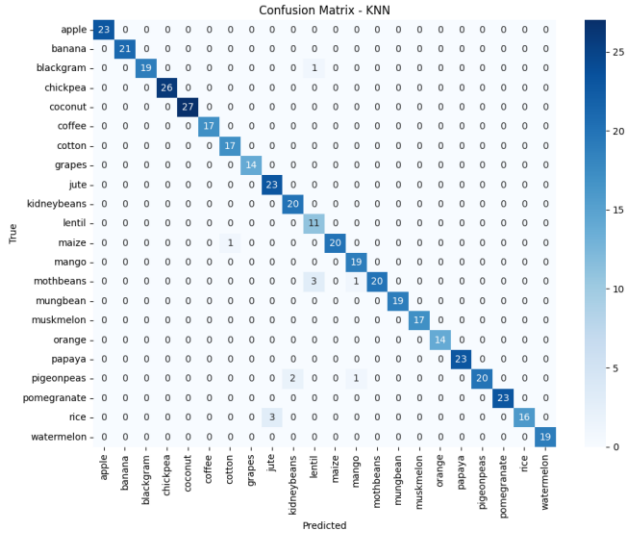
Classification Report:

Classification report (test):

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	23
banana	1.00	1.00	1.00	21
blackgram	1.00	0.95	0.97	20
chickpea	1.00	1.00	1.00	26
coconut	1.00	1.00	1.00	27
coffee	1.00	1.00	1.00	17
cotton	0.94	1.00	0.97	17
grapes	1.00	1.00	1.00	14
jute	0.88	1.00	0.94	23
kidneybeans	0.91	1.00	0.95	20
lentil	0.73	1.00	0.85	11
maize	1.00	0.95	0.98	21
mango	0.90	1.00	0.95	19
mothbeans	1.00	0.83	0.91	24
mungbean	1.00	1.00	1.00	19
muskmelon	1.00	1.00	1.00	17
orange	1.00	1.00	1.00	14
papaya	1.00	1.00	1.00	23
pigeonpeas	1.00	0.87	0.93	23
pomegranate	1.00	1.00	1.00	23
rice	1.00	0.84	0.91	19
watermelon	1.00	1.00	1.00	19
accuracy			0.97	440
macro avg	0.97	0.97	0.97	440
weighted avg	0.98	0.97	0.97	440

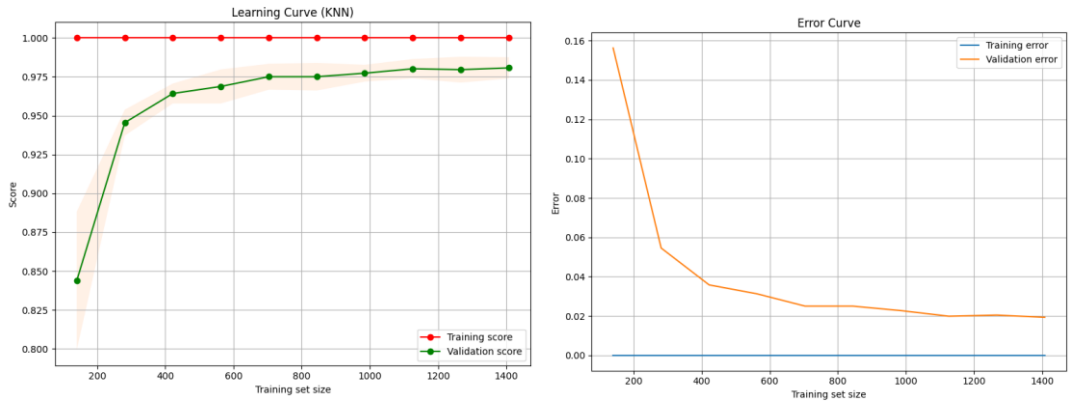
Molte classi risultano classificate in modo impeccabile, con valori di precisione e recall pari a 1,00. Alcune classi, tuttavia, evidenziano prestazioni leggermente inferiori.

Confusion Matrix:



L'elevata intensità dei valori lungo la diagonale principale conferma che la maggior parte delle predizioni è corretta. Si notano però alcuni valori fuori dalla diagonale, indicativi di errori di classificazione, leggermente più frequenti rispetto al modello precedente.

Learning Curves:



Nella *learning curve* (grafico a sinistra), la linea rossa del *training score* rimane costantemente a 1,00, segno che il modello KNN classifica sempre in modo perfetto i dati di addestramento. La linea verde del *validation score* parte da circa 0,835 e cresce rapidamente con l'aumentare dei campioni, stabilizzandosi intorno a 0,975. Questo andamento indica che il modello trae notevoli benefici dall'incremento dei dati, riducendo gli errori di generalizzazione.

Nella *error curve* (grafico a destra), la linea blu dell'errore di training resta fissa a 0,00, mentre la linea arancione dell'errore di validazione inizia da circa 0,14 e diminuisce progressivamente fino a stabilizzarsi attorno a 0,025. Ciò conferma che l'aggiunta di dati riduce gli errori, ma oltre una certa soglia i vantaggi aggiuntivi diventano trascurabili.

4.4.3 SVC

Cross Validation:

```
- accuracy: mean=0.9858, std=0.0065, var=0.000042
- precision: mean=0.9875, std=0.0056, var=0.000032
- recall: mean=0.9859, std=0.0063, var=0.000040
- f1_macro: mean=0.9857, std=0.0064, var=0.000042
- f1_weighted: mean=0.9857, std=0.0066, var=0.000043
- precision_weighted: mean=0.9876, std=0.0057, var=0.000033
- recall_weighted: mean=0.9858, std=0.0065, var=0.000042
```

Il modello presenta prestazioni complessivamente molto elevate e ben equilibrate: l'accuratezza media si attesta intorno al 98,6%, con valori di precisione e recall pressoché coincidenti. L'F1-score conferma l'equilibrio tra queste metriche, mentre le deviazioni standard ridotte indicano un comportamento stabile nei diversi fold della cross-validation.

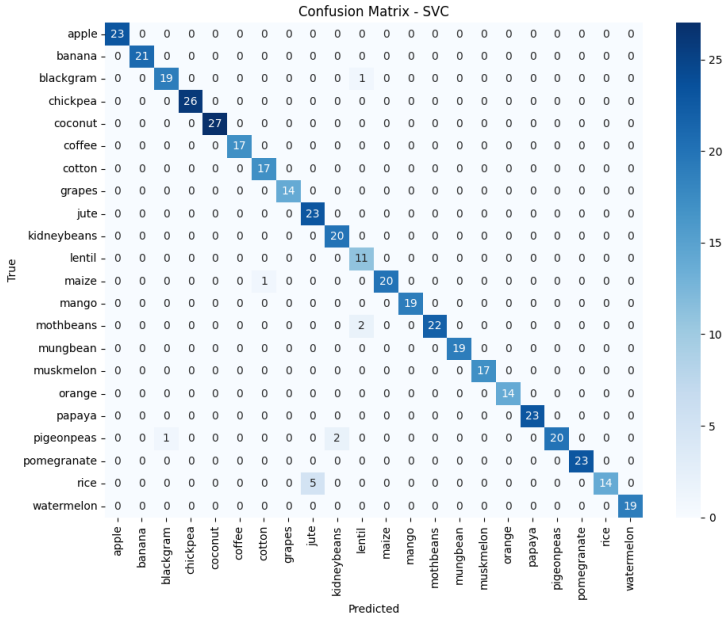
Classification Report:

Classification report (test):

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	23
banana	1.00	1.00	1.00	21
blackgram	0.95	0.95	0.95	20
chickpea	1.00	1.00	1.00	26
coconut	1.00	1.00	1.00	27
coffee	1.00	1.00	1.00	17
cotton	0.94	1.00	0.97	17
grapes	1.00	1.00	1.00	14
jute	0.82	1.00	0.90	23
kidneybeans	0.91	1.00	0.95	20
lentil	0.79	1.00	0.88	11
maize	1.00	0.95	0.98	21
mango	1.00	1.00	1.00	19
mothbeans	1.00	0.92	0.96	24
mungbean	1.00	1.00	1.00	19
muskmelon	1.00	1.00	1.00	17
orange	1.00	1.00	1.00	14
papaya	1.00	1.00	1.00	23
pigeonpeas	1.00	0.87	0.93	23
pomegranate	1.00	1.00	1.00	23
rice	1.00	0.74	0.85	19
watermelon	1.00	1.00	1.00	19
accuracy			0.97	440
macro avg	0.97	0.97	0.97	440
weighted avg	0.98	0.97	0.97	440

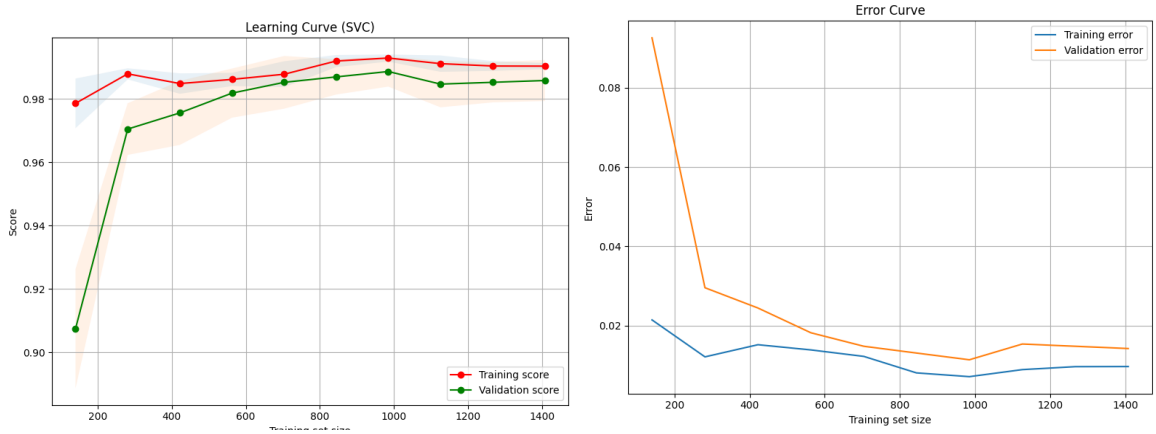
Dal *classification report* dell'SVC emerge un'accuratezza complessiva del 97% sul test set, con valori medi di precisione, recall e F1-score coerenti e ben bilanciati. Numerose classi sono state riconosciute in modo impeccabile, mentre alcune categorie presentano prestazioni più contenute.

Confusion Matrix:



Gli errori fuori dalla diagonale sono rari e circoscritti, concentrati in particolare nelle classi già evidenziate dal *classification report*. L'intensità dei colori conferma che, per molte categorie, le predizioni risultano sempre corrette, senza alcuna confusione con altre classi.

Learning Curve:



Osservando il grafico della Learning Curve possiamo concludere che il modello beneficia sia in training che in test l'aumentare del numero di esempi, partendo da un valore di trainng pari a 0.98 passando per 0.99 tuttavia vediamo che negli esempi successivi lo score diminuisce, questo vale anche in fase di validation partendo da uno score molto basso arrivando poi ad un picco vicino allo 0.99 per poi diminuire. Lo stesso vale per la curva di errore.

4.4.4 Logistic Regression

Cross Validation:

```
- accuracy: mean=0.9790, std=0.0075, var=0.000057
- precision: mean=0.9806, std=0.0073, var=0.000054
- recall: mean=0.9788, std=0.0075, var=0.000057
- f1_macro: mean=0.9789, std=0.0075, var=0.000057
f1_weighted: mean=0.9789, std=0.0076, var=0.000058
- precision_weighted: mean=0.9806, std=0.0075, var=0.000056
- recall_weighted: mean=0.9790, std=0.0075, var=0.000057
```

Il modello evidenzia prestazioni complessivamente molto buone, con un'accuratezza media intorno al 97,9% e valori di precisione, recall e F1-score (sia macro che *weighted*) pressoché coincidenti. Le deviazioni standard ridotte confermano la stabilità dei risultati.

Classification Report:

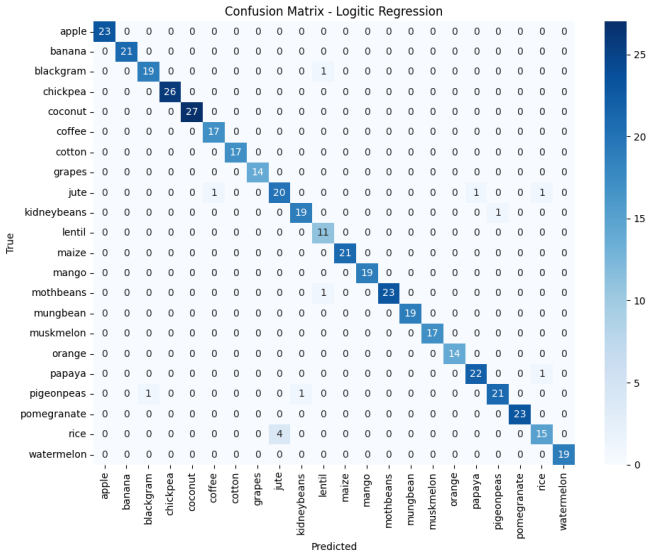
```
Classification report (test):
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        23
      banana       1.00      1.00      1.00        21
   blackgram       0.95      0.95      0.95        20
    chickpea       1.00      1.00      1.00        26
     coconut       1.00      1.00      1.00        27
      coffee       0.94      1.00      0.97        17
      cotton       1.00      1.00      1.00        17
      grapes       1.00      1.00      1.00        14
        jute       0.83      0.87      0.85        23
  kidneybeans       0.95      0.95      0.95        20
      lentil       0.85      1.00      0.92        11
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        19
   mothbeans       1.00      0.96      0.98        24
    mungbean       1.00      1.00      1.00        19
   muskmelon       1.00      1.00      1.00        17
      orange       1.00      1.00      1.00        14
      papaya       0.96      0.96      0.96        23
  pigeonpeas       0.95      0.91      0.93        23
 pomegranate       1.00      1.00      1.00        23
        rice       0.88      0.79      0.83        19
  watermelon       1.00      1.00      1.00        19

    accuracy                           0.97       440
   macro avg       0.97      0.97      0.97       440
weighted avg       0.97      0.97      0.97       440
```

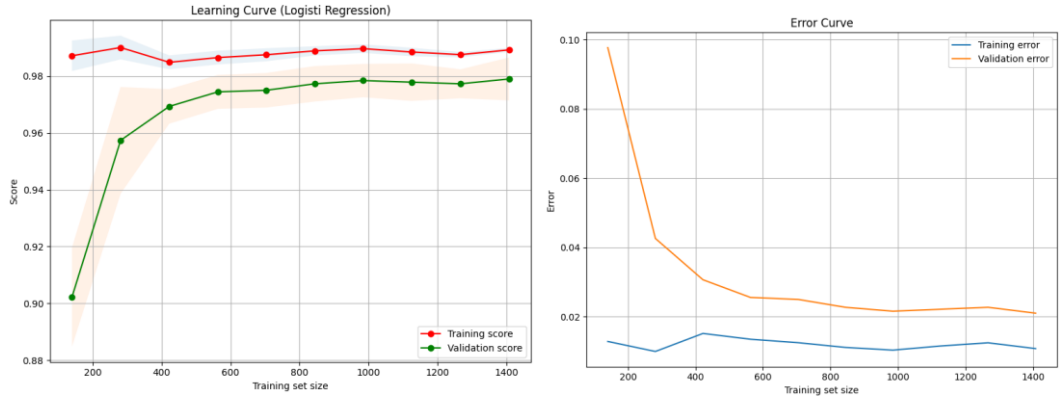
Dal *classification report* emerge che il modello ottiene risultati eccellenti su diverse classi, con una precisione complessiva del 97%. Pur essendo leggermente inferiore rispetto al valore massimo registrato, il modello si conferma comunque affidabile e performante.

Confusion Matrix:



Gli errori fuori dalla diagonale sono pochi e ben localizzati, riguardando soprattutto le classi già segnalate nel *classification report*. L'intensità dei colori conferma che, per la maggior parte delle categorie, le predizioni risultano sempre corrette e prive di confusione con altre classi.

Learning Curve:



Nella learning curve (a sinistra) il punteggio sul training set resta piuttosto stabile, mentre quello di validazione cresce in modo netto con l'aumentare dei dati, fino a stabilizzarsi poco sotto il 94%. Questo comportamento indica che il modello beneficia dell'incremento di campioni, migliorando la propria capacità di generalizzazione, ma raggiunge una soglia oltre la quale i miglioramenti diventano marginali.

La curva dell'errore (a destra) racconta la stessa dinamica al contrario.

In sintesi, il modello è solido, prevedibile e ben generalizzato, sfrutta bene l'aumento dei dati, ma raggiunge presto il suo limite fisiologico di accuratezza, che rimane comunque elevato e affidabile.

4.5 Considerazioni Finali

Dall'analisi complessiva dei modelli addestrati e ottimizzati, emerge in maniera chiara che la **Random Forest** si è dimostrata l'algoritmo più performante tra quelli testati. Con un'accuratezza media prossima al **99%**, questo modello ha garantito risultati solidi e costanti, distinguendosi per robustezza e bassa sensibilità agli outlier. è stato

scelto di salvare il modello pre-addestrato in formato .pkl, così da poterlo riutilizzare rapidamente senza necessità di un nuovo addestramento.

Ciò non esclude, tuttavia, la possibilità di impiegare anche gli altri algoritmi analizzati come strumenti di confronto o per applicazioni specifiche.

5. Knowledge Base

La Knowledge Base rappresenta un dominio di conoscenza e consente di risolvere problemi al suo interno attraverso l'inferenza logica.

Si distingue in:

- Fatti: Affermazioni sempre vere, utilizzate per costruire la base di inferenza
- Regole: Affermazioni vere solo se tutte le proposizioni e/o sotto-regole che la compongono risultano vere

5.1 Scelte di Progetto

La Knowledge Base è stata sviluppata in Prolog e ha lo scopo di supportare le attività di predizione, offrendo un approfondimento più ampio e strutturato sul dominio di riferimento.

5.2 Funzionalità

La Knowledge Base ha lo scopo di fornire raccomandazioni sulle colture da adottare secondo diversi criteri, tra cui:

- **Periodo di semina**: suggerimenti in base al mese previsto per la semina.
- **Risorse disponibili**: indicazioni considerando vincoli come budget e superficie coltivabile.
- **Somiglianza**: proposte di colture simili a una coltura specificata come input.
- **Condizioni meteorologiche locali**: selezione delle colture in funzione del clima dell'area di riferimento.

5.3 Struttura

5.3.1 Fatti

Sono presenti due tipologie di Fatti.

- 'Crop', rappresentante la singola classe di coltura, costruita su parametri quali:
    - Nome: denominazione della coltura

- N: livello di azoto presente nel terreno
- P: livello di fosforo presente nel terreno
- K: livello di potassio presente nel terreno
- Temp_min: temperatura minima necessaria per la coltura
- Temp_max: temperatura massima tollerata per la coltura
- Ph_%: livello di ph del terreno
- Humidity: percentuale di umidità ottimale
- Pioggia: quantitativo di precipitazioni utile alla coltura
- Costo per acro: costo stimato per acro per di coltivazione

```
% crop(Nome, N, P, K, temp_MIN, temp_MAX, ph_%, humidity, piogge, costo_per_acro)
```

- Month, indica il mese di semina:
  - Nome: nome del mese
  - Tempmin: temperatura minima registrata in quel mese;
  - Tempmax: temperatura massima registrata in quel mese;

```
% month(NomeMese, TempMin, TempMax)
```

5.3.2 Regole

La Knowledge base è composta dalle seguenti regole:

- Consiglia quale coltura utilizzare in un dato mese

```
suggest_crops_by_month(Month, Crops) :-
    findall(Crop, crop_suitable_for_month(Crop, Month), Crops).
```

- Questa regola stabilisce l'idoneità di una coltura in base al mese: un *crop* è considerato adatto se l'intervallo di temperatura del mese rientra interamente nel range ottimale previsto per quella coltura.

```
crop_suitable_for_month(Crop, Month) :-
    crop(Crop,_, _, _, Tmin, Tmax, _, _, _, _),
    month(Month, TminMonth, TmaxMonth),
    Tmin =< TminMonth,
    Tmax >= TmaxMonth.
```

- Questa regola individua, tra le colture considerate idonee, quelle con il costo per acro minimo, restituendo un elenco delle opzioni economicamente più vantaggiose.

```
min_cost_crop(Acri, Budget, BestCrop) :-
    findall((CostoTot, C),
        (sowing_economical_info(Acri, Budget, C),
        crop(C, _, _, _, _, _, _, _, _, Costo),
        CostoTot is Acri * Costo),
        Lista),
    sort(Lista, [(_, BestCrop)|_]).
```

- Questa regola verifica se una coltura può essere seminata con il budget e la superficie di terreno disponibili.

```
sowing_economical_info(Acri, Budget, Crop) :-
    crop(Crop, _, _, _, _, _, _, _, _, Costo),
    Totale is Acri * Costo,
    Totale =< Budget.
```

- Questa regola restituisce l'elenco di tutte le colture compatibili con le condizioni climatiche e del terreno, valutando parametri quali pH, umidità e pioggia.

```
sowing_zone_info(PHzone, UmidZone, RainZone, Crops) :-
    findall(Crop,
        (
            crop(Crop, _, _, _, _, _, PH, Umid, Rain, _),
            ph_category(PH, PHzone),
            humidity_category(Umid, UmidZone),
            rainfall_category(Rain, RainZone)
        ),
        Crops).
```

- Queste tre regole classificano il valore del ph in neutro, acido o basico.

```
ph_category(PH, acido)  :- PH < 6.5.
ph_category(PH, neutro) :- PH >= 6.5, PH =< 7.5.
ph_category(PH, basico) :- PH > 7.5.
```

- Queste tre regole classificano l'umidità in valori come bassa media e alta.

```
humidity_category(Val, bassa) :- Val < 60.
humidity_category(Val, media) :- Val >= 60, Val =< 75.
humidity_category(Val, alta)  :- Val > 75.
```

- Queste tre regole classificano il livello di pioggia in arida piovosa e standard.

```
rainfall_category(Rain, arida)    :- Rain < 75.
rainfall_category(Rain, standard) :- Rain >= 75, Rain =< 150.
rainfall_category(Rain, piovosa)  :- Rain > 150.
```

- Questa regola utilizza la distanza euclidea per individuare la coltura più simile a quella fornita come input.

```
most_sim(RefCrop, SimCrop) :-
    crop(RefCrop, N1, P1, K1, T11, T12, PH1, H1, R1, _),
    findall((Dist, C),
        (crop(C, N2, P2, K2, T21, T22, PH2, H2, R2, _),
         C \= RefCrop,
         Dist is sqrt((N1-N2)^2 + (P1-P2)^2 + (K1-K2)^2 + (T11-T12)^2 + (T21-T22)^2 + (PH1-PH2)^2 +  (H1-H2)^2 + (R1-R2)^2)),
        Lista),
    sort(Lista, [(_, SimCrop)|_]).
```

- Questa regola consiglia la coltura più adatta in base ai valori forniti in input.

```
recommended_crop(Acri, Budget, PHzone, UmidZone, RainZone, Crop) :-
    sowing_zone_info(PHzone, UmidZone, RainZone, Crop),
    crop(Crop, _, _, _, _, _, _, _, _, Costo),
    Totale is Acri * Costo,
    Totale =< Budget.
```

5.4 Esempi

Coltura scelta in base al Mese

```
1. Semina in base al Mese
2. Scelta della semina in base al Budget e terreno disponibili
3. Scelta della semina per condizioni climatiche
4. Colture simili a quella data
5. Coltura consigliata in base ai parametri
0. Torna indietro

|___: 1
Inserisci il mese (es. january): august
Colture suggerite: ['pigeonpeas', 'lentil', 'orange']
```

Coltura scelta in base al budget e terreno disponibili, include la ricerca della coltura a costo minimo

```
Select one of the following options:
1. Semina in base al Mese
2. Scelta della semina in base al Budget e terreno disponibili
3. Scelta della semina per condizioni climatiche
4. Colture simili a quella data
5. Coltura consigliata in base ai parametri
0. Torna indietro

|___: 2
Inserisci numero di acri: 10
Inserisci budget disponibile: 6000
Colture possibili: ['kidneybeans', 'pigeonpeas', 'mothbeans', 'mungbean', 'blackgram', 'lentil']
Coltura con costo minimo: blackgram
```

Coltura consigliata in base alle condizioni climatiche

```
Select one of the following options:
1. Semina in base al Mese
2. Scelta della semina in base al Budget e terreno disponibili
3. Scelta della semina per condizioni climatiche
4. Colture simili a quella data
5. Coltura consigliata in base ai parametri
0. Torna indietro

|____: 3
Inserisci zona PH (acido/neutro/basico): neutro
Inserisci zona umidità (bassa/media/alta): media
Inserisci zona pioggia (arida/standard/piovosa): arida
Colture consigliate: [['blackgram', 'lentil']]
```

Coltura simile a quella data

```
Select one of the following options:
1. Semina in base al Mese
2. Scelta della semina in base al Budget e terreno disponibili
3. Scelta della semina per condizioni climatiche
4. Colture simili a quella data
5. Coltura consigliata in base ai parametri
0. Torna indietro

|____: 4
Inserisci il nome della coltura di riferimento: maize
```

Coltura consigliata in base ai dati forniti

```
1. Semina in base al Mese
2. Scelta della semina in base al Budget e terreno disponibili
3. Scelta della semina per condizioni climatiche
4. Colture simili a quella data
5. Coltura consigliata in base ai parametri
0. Torna indietro

|____: 5
Inserisci numero di acri: 1
Inserisci budget disponibile: 500
Inserisci zona PH (acido/neutro/basico): acido
Inserisci zona umidità (bassa/media/alta): bassa
Inserisci zona pioggia (arida/standard/piovosa): arida
Coltura consigliata: ['pigeonpeas', 'mothbeans', 'mungbean', 'blackgram', 'lentil']
```

6. Conclusione e sviluppi futuri

Il progetto è orientato a un'evoluzione continua, con l'obiettivo di ampliare il numero di colture e la quantità di dati disponibili. Questa espansione mira a migliorare ulteriormente la precisione delle previsioni e ad arricchire il sistema con nuove funzionalità dedicate sia alle modalità di semina, sia agli aspetti tecnici, sempre all'interno del contesto agricolo.

7. Riferimenti

1. Crop Recommendation Dataset
2. pandas - Python Data Analysis Library
3. NumPy
4. User Guide — scikit-learn 1.7.0 documentation - sklearn
5. User guide and tutorial — seaborn 0.13.2 documentation
6. SVC — scikit-learn 1.7.1 documentation
7. Qual è l'algoritmo k-nearest neighbors? | IBM
8. RandomForestClassifier — scikit-learn 1.7.1 documentation
9. StratifiedKFold — scikit-learn 1.7.1 documentation