UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI INFORMATICA
Corso di Laurea Triennale in Informatica

CASO DI STUDIO DEL CORSO IN INGEGNERIA DELLA CONOSCENZA

SMART FARM

SISTEMA INTELLIGENTE PER LA CLASSIFICAZIONE E OTTIMIZZAZIONE DELLE COLTURE

–

GESTIONE DELLE MALATTIE ATTRAVERSO UNA KNOWLEDGE BASE

Realizzato dal: ***Team SmartFarm***

Valletta Antonio
Matr. 698686

Paparella Alessandro
Matr. 697384

Lobascio Francesco
Matr. 697383

Tempesta Felice
Matr. 706463

GitHub Repository: https://github.com/franz-ops/Icon2021

INDICE

1. INTRODUZIONE

1.1 IDEA DI PROGETTO

Per la realizzazione del caso di studio, ci si è posti l'obiettivo di simulare le reali esigenze di un imprenditore e della sua azienda agricola pugliese, desiderosi di abbandonare le tradizionali metodologie di previsione, di lavoro, di studio e di fornire loro un sistema intelligente, che sia in grado di analizzare le informazioni fornite, allo scopo di restituire dati specifici.

Nel dettaglio, ci si è posto l'obiettivo di supportare il lavoro di quest'azienda, su pratiche comuni, quanto laboriose, che richiedono normalmente la presenza di figure esperte, di lunghi studi e ricerche. Pratiche rese ad oggi semplici, veloci, dinamiche ed intelligenti, grazie all'implementazione di algoritmi, metodologie e strumenti, che hanno consentito di automatizzare procedure di ottimizzazione per la gestione dei campi, classificazione delle migliori aree da coltivare ed individuazione delle malattie.

Migliorarsi, automatizzarsi ed essere all'avanguardia con le nuove tecnologie è l'obiettivo e il Team, con il suo lavoro, ha provato a fornire un vantaggio competitivo nel mercato pugliese e nazionale.

1.2 OBIETTIVO DELL'AZIENDA AGRICOLA

Analizzando più nel dettaglio gli scopi e le richieste di tale azienda, possiamo definire i seguenti punti:

- L'azienda agricola desidera che vengano gestisce automaticamente e in modo intelligente le seguenti procedure:
  - Analisi dei dati del territorio, quali umidità, temperatura, ph, quantità di piogge ed altri, che debbano permettere di classificare le aree migliori nella quale seminare e dunque, definire le migliori coltivazioni per uno specifico terreno;
  - Partendo da un budget aziendale, predefinito e da rispettare, adottare metodologie ed algoritmi, che possano ottimizzare le coltivazioni in base ai costi di mantenimento (acqua da erogare, servizi vari, trattamenti, concimi ecc…), con l'obiettivo primario di minimizzare sprechi e costi e massimizzare i profitti.
  - Sistema esperto che sia in grado di possedere informazioni dettagliate sulle comuni malattie delle colture interessate e, supportare l'utente a riconoscere la specifica malattia, indicando esclusivamente i sintomi visibili. Non tutti gli agricoltori o i dipendenti aziendali possono ricordare ogni tipo di malattia. Per tal ragione, scopo di tale richiesta è quello di identificare subito il tipo di problema in base ai sintomi fisici.
  - Un'interfaccia utente, che permetta di raccogliere le informazioni su una specifica coltura e un sistema di query che possa interagire con l'utente e interrogare la KB per fornire i dati richiesti.

- Per quanto riguarda la capacità del sistema, in modo intelligente, di individuare la malattia, sulla base delle regole della KB applicate alle risposte fornite dall'utente (con indicazione dei sintomi). L'azienda ha esplicitamente richiesto di gestire: Riso, Mais, Pomodoro e Cotone. Tale volontà, è dipesa da una particolare gestione di tali colture e la necessità di automatizzare con un sistema esperto alcuni aspetti del loro ciclo di vita.
- La richiesta di operare su colture come quelle precedentemente indicate, nasce, anche, da una scarsa disponibilità nel mercato locale di tali prodotti e che può dunque, grazie al sussidio di tali sistemi esperti, rappresentare un punto di vantaggio per l'azienda in questione, sia da un punto di vista competitivo, sia economico.

1.2 ORGANIZZAZIONE

Per la conduzione del lavoro il Team si è periodicamente riunito per definire la << mission aziendale >> ed effettuare l'analisi dei requisiti e del bisogno informativo richiesto.
Ci si è serviti di strumenti di comunicazione che garantissero la condivisione e la collaborazione di ogni membro del gruppo. Per procedere, sono stati analizzati e studiati gli argomenti del corso di Ingegneria della Conoscenza, dal quale partire per adottare algoritmi, metodologie e strumenti utili agli obiettivi preposti.

1.3 ASSEGNAZIONE TASK

Dopo un'attenta analisi, attività di ricerca e individuazione di come realizzare e applicare gli obiettivi preposti, si è effettuata una suddivisione dei compiti, che avesse permesso a ciascun membro di lavorare su una specifica tematica. In seguito al completamento di ogni task, decisive sono state le call effettuate, per discutere del lavoro condotto con gli altri membri del Team e poter raccogliere idee, punti di vista, accorgimenti, utili al miglioramento del sistema.

2. LINGUAGGIO, AMBIENTI UTILIZZATI E ALGORITMI IMPLEMENTATI

2.1 LINGUAGGIO DI PROGRAMMAZIONE

Per la realizzazione del sistema esperto e di tutte le sue componenti si è deciso di utilizzare il linguaggio di programmazione Python.
A tal proposito la versione minima con il quale è possibile visualizzare ed eseguire il progetto è:

- Python 3.7+

2.2 AMBIENTI DI SVILUPPO – IDE

Per realizzare e testare il progetto in questione, non ci si è predisposti uno specifico IDE da utilizzare, ma la scelta è stata libera per ogni membro del Team.
A tal proposito, per quanto possa sembrare una scelta libera e casuale, in realtà è stata intrapresa anche per garantire i concetti di "Interoperabilità" e "Portabilità" e testare dunque il corretto funzionamento del progetto su macchine e ambienti differenti.
N.B. Importante sempre è possedere la versione minima di Python, al punto precedente indicato.

2.3 INTRODUZIONE AGLI ALGORITMI UTILIZZATI

1°Algoritmo: Branch-and-Bound.

Questa è una tecnica generale per la risoluzione di problemi di ottimizzazione combinatoria, ossia problemi che hanno uno spazio di soluzioni finito. Questa metodologia ci permette di effettuare la ricerca delle soluzioni ottimali, suddividendo il problema in sotto-problemi più semplici.
Maggiori informazioni sul suo utilizzo e sulle sue proprietà è possibile ritrovarli al punto 5.1.

2° Algoritmo: Random Forest.

Una delle parti più importanti dell'apprendimento automatico è la classificazione.
Questo modello ci permette di classificare in modo preciso le osservazioni relative a dati specifici forniti dall'azienda agricola, per poter capire se seminare una particolare coltura in uno specifico terreno.
Maggiori dettagli di utilizzo sono descritti al punto 4.1.

3. ATTIVITA' DI RICERCA E DATASET

3.1 Attività di ricerca

Il lavoro condotto dal Team si è basato sulla ricerca accurata di un dataset che meglio potesse soddisfare le richieste del committente.
Principalmente esso è utilizzato per le attività intelligenti di classificazione, nella quale, in base ai dati posseduti, si riesce a classificare la migliore area, nella quale una specifica coltura può essere piantata.
Nel dettaglio, tale dataset, è possibile reperirlo sul repository GitHub, nella sezione ***"Crop_recommendation.csv"***.
Nel dettaglio, l'obiettivo è quello di classificare il miglior terreno, per una specifica coltura in base a specifiche condizioni:

- ✓ Temperatura
- ✓ Umidità
- ✓ Ph
- ✓ Quantità di pioggia
- ✓ N (Azoto)
- ✓ P (Fosforo)
- ✓ K (Potassio)

3.2DATASET UTILIZZATO

N	P	K	temperature	humidity	ph	rainfall	label
90	42	43	20.87974371	82.00274423	6.502985292000001	202.9355362	rice
85	58	41	21.77046169	80.31964408	7.038096361	226.6555374	rice
60	55	44	23.00445915	82.3207629	7.840207144	263.9642476	rice
100	48	16	25.71895816	67.22190688	5.54990242	74.51490791	maize
79	51	16	25.33797709	68.49835977	6.586244581	96.46380213	maize
94	39	18	23.89114571	57.48775781	5.893093135	102.8301942	maize
75	49	15	21.53574127	71.50905983	5.918263801	102.4852929	maize
78	48	22	23.08974909	63.10459626	5.588650585	70.43473609	maize
59	55	79	20.36720401	16.89574311	8.766128654	82.2545577	chickpea
36	76	75	18.38120357	16.63805158	8.736337905	70.52056697	chickpea
57	68	81	17.17012591	17.30457712	8.081095263	72.78624223	chickpea
35	66	81	19.37101121	15.77458129	6.138243973	85.24819851	chickpea
35	64	78	17.92845928	14.27327988	7.496645259	85.37378769	chickpea
27	80	15	19.07096165	21.21092266	5.7883869510000014	86.21917578	kidneybeans
10	55	23	21.18853178	19.63438599	5.728233081	137.1948633	kidneybeans
23	65	20	23.0429097	22.42610972	5.833940084	108.3684316	kidneybeans
19	78	16	20.65375833	23.10538637	5.967533236	67.71768947	kidneybeans
19	65	25	18.09551014	18.29318436	5.6250964460000015	144.7902323	kidneybeans
22	70	19	18.23775702	21.07643273	5.515615023	69.44951585	kidneybeans
95	88	52	28.00316034	78.90085998	6.235461772000001	94.68180316	banana
104	73	46	29.1400919	80.1190228	6.28236237	90.45142867	banana
102	73	52	27.9122104	83.36307683	6.356090905	90.24211529	banana
100	74	52	25.43480512	81.53977797	5.837258235	96.47800391	banana

Esempi di informazioni su alcune colture

4. CLASSIFICAZIONE DELLE MIGLIORI AREE NELLA QUALE COLTIVARE

4.1 RANDOM FOREST

Questo algoritmo di classificazione, si basa essenzialmente nell'utilizzo di alberi di decisione, che sono stati utilizzati per tale algoritmo.

Abbiamo immaginato che il nostro dataset, fosse diviso da numeri (1 e 0), suddivisi per coltura. Pertanto, possiamo dire di effettuare una mediazione fra più alberi di decisione.

Dunque:

- per ogni tipo di coltura da classificare viene effettuata una predizione;
- Le predizioni per tutte le colture vengono successivamente unite per creare una predizione finale.

I valori, in precedenza citati, 1 e 0, verranno utilizzati per capire se uno specifico terreno, per una specifica coltura, può essere seminato, coltivato, irrigato ecc..

La classe predetta è un voto espresso dagli alberi della foresta, ponderato dalle loro stime di probabilità. Cioè, la classe prevista è quella con la stima di probabilità media più alta tra gli alberi.

Una metodologia semplice, ma potente.

COME LO ABBIAMO IMPLEMENTATO:

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

data = pd.read_csv('datasets/Crop_recommendation.csv')

X = data.iloc[:, 0:7]
y = data.iloc[:, 7:8]
fn = data.columns[:7].values

X_train, X_test, y_train, y_test = train_test_split(X, np.ravel(y), test_size=0.3, random_state=42)

clf = RandomForestClassifier(max_depth=7, random_state=10)

clf.fit(X_train, y_train)
```

5. OTTIMIZZAZIONE DELLE COLTIVAZIONI

5.1 Branch-and-Bound

Questa metodologia è da considerarsi una delle più sofisticate nell'ambito della ricerca di soluzioni.

Essa combina l'efficienza nello spazio delle strategie in profondità. E' molto efficace quando esistono molti cammini verso i/il goal.

All'interno del progetto, tale tecnica è stata utilizzata a scopo decisionale per rispettare il budget aziendale, ottimizzare le coltivazioni, con l'obiettivo di minimizzare i costi relativi all'irrigazione, ai concimi e ai trattamenti ed ottenere il massimo profitto.

Grazie a questa tecnica, testata ed implementa in modo randomico, ci permette di avere il giusto compromesso tra i costi di mantenimento e la produzione prevista.

A tal proposito, vengono consigliate le varietà di coltivazioni.

6. KNOWLEDGE BASE

6.1 Funzione della Knowledge Base

Per soddisfare un requisito richiesto dall'azienda agricola, ossia quello di realizzare un sistema esperto che, dati i sintomi ed alcune caratteristiche estetiche delle colture, siano in grado di definire il tipo di malattia.

Per poter effettuare questo, è stato deciso di utilizzare una Knowledge Base. Quest'ultima non proviene da dataset o documenti specifici, ma è frutto di un lavoro individuale di ciascun membro del gruppo.

Vista l'impossibilità di creare una KB per ogni coltura presente nel dataset usato, lo abbiamo realizzato per 4 tipologie, in modo da dividerci il lavoro.

Nel dettaglio:

- *Riso*
- *Mais*
- *Pomodoro*
- *Cotone*

Si è implementata una Knowledge base nella quale sono inseriti:

- *Malattie delle foglie*
- *Malattie batteriche*
- *Muffe*
- *Macchie*
- *Anomalie visibili sulle foglie*
- *Appassimento*

6.2 Costruzione della KB

Per la progettazione e creazione della Knowledge Base, ogni membro del Team ha adottato un proprio metodo di rappresentazione logica.

Per esempio, con l'ausilio di un albero decisionale, per poter progettare il funzionamento e l'interrogazione della KB.

Nel dettaglio, la Knowledge Base è stata realizzata con regole che utilizzano la metodologia del ***Backward Chaining*** sulla base di risposte fornite dall'utente alle domande presenti nella *"questions_base".*

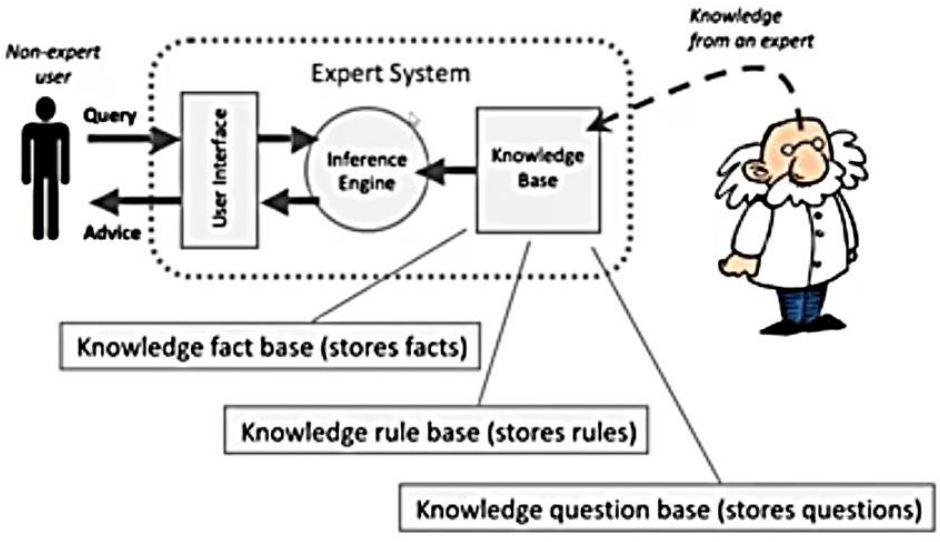
Lo strumento di cui ci si è serviti per costruire e gestire questo punto è **PyKE, ossia un sistema esperto. Un interprete e framework di codifica che ci ha permesso di costruire e gestire un sistema esperto.**

6.3 PyKE

COS'E' PyKE?

Python Knowledge Engine è:

- ✓ un sistema esperto;
- ✓ Integra una forma di programmazione logica in Python;
- ✓ Fornisce un **motore di inferenza che applica regole ai fatti;**
- ✓ Si concentra **sull'archiviazione di una base di conoscenza dei fatti e una base di domande.**
- ✓ Opera solo in Python,



UTILIZZO DI PyKE

STEP 1: Creare un oggetto motore

- Da Pyke importare Knowledge_egine
- my_engine = knowledge_engine.engine(__file__)

```
def bc_test_questions():
    engine.reset()  # Allows us to run tests multiple times.

    engine.activate('bc_simple_rules_questions')
```

STEP 2: Attivare la base delle regole

```
def bc_test_questions():
    engine.reset()  # Allows us to run tests multiple times.

    engine.activate('bc_simple_rules_questions')

    print("doing proof")
    cont = 0
    try:
        with engine.prove_goal('bc_simple_rules_questions.what_disease($malattia)') as gen:
            for vars, plan in gen:
                print("Malattia: %s" % (vars['malattia']))
                cont = cont + 1

    except Exception:
        # This converts stack frames of generated python functions back to the
        # .krb file.
        krb_traceback.print_exc()
        sys.exit(1)

    if cont == 0:
        print("Malattia: nessuna o sconosciuta")

    print("done")
```

STEP 3: Definire i goals

- da PyKE importare i goal
- my_goal= goal.compile('goal')

```
    print("doing proof")
    cont = 0
    try:
        with engine.prove_goal('bc_simple_rules_questions.what_disease($malattia)') as gen:
            for vars, plan in gen:
                print("Malattia: %s" % (vars['malattia']))
                cont = cont + 1

    except Exception:
        # This converts stack frames of generated python functions back to the
        # .krb file.
        krb_traceback.print_exc()
        sys.exit(1)

    if cont == 0:
        print("Malattia: nessuna o sconosciuta")

    print("done")
```

LOAD PyKE DRIVER

```
import sys

sys.path.append(r'kb')
import driver_simple as driver

if __name__ == '__main__':
    driver.bc_test_questions()
```

driver_simple.py

```
dir_path = 'kb/compiled_krb/'

try:
    shutil.rmtree(dir_path)
except OSError as e:
    print("Error: %s : %s" % (dir_path, e.strerror))

engine = knowledge_engine.engine(__file__)


def bc_test_questions():
    engine.reset()  # Allows us to run tests multiple times.

    engine.activate('bc_simple_rules_questions')

    print("doing proof")
    cont = 0
    try:
        with engine.prove_goal('bc_simple_rules_questions.what_disease($malattia)') as gen:
            for vars, plan in gen:
                print("Malattia: %s" % (vars['malattia']))
                cont = cont + 1

    except Exception:
        # This converts stack frames of generated python functions back to the
        # .krb file.
        krb_traceback.print_exc()
        sys.exit(1)

    if cont == 0:
        print("Malattia: nessuna o sconosciuta")

    print("done")
```


KNOWLEDGE BASES – BASE DEI FATTI E BASI DELLE REGOLE

- **Fact base** contiene l'insieme dei fatti **(.kfb);**
  - Fatti specifici che vengono cancellati una volta usati;
  - Fatti universali che ci sono sempre.
- **Rule base:** memorizza le regole nella forma di Knowledge Base e di file **(.krb)**
  - Può contenere regole di concatenamento sia in avanti che indietro

```
# bc_simple_rules.krb


# MAIS

what_malattia_Elmintosporiosi
    use what_disease(Elmintosporiosi)
    when
        questions.scelta_coltura($ans)
        check $ans in (2,)
        questions.macchie_foglie(True)
        questions.colore_macchie($c)
    check $c in (3,)
```

```
# POMODORO

what_disease_Peronospora_precoce
    use what_disease(Peronospora_precoce)
    when
        questions.scelta_coltura($ans)
        check $ans in (3,)
        questions.macchie(True)
        questions.macchie_scure(True)
        questions.macchie_anelli_foglie(True)
```

KNOWLEDGE BASES – QUESTION BASES

- **Question Bases:** basi di domande di conoscenza che vengono sottoposte all'utente finale **(.kqb).** Quest'ultimo dovrà fornire una risposta e sulla base di questo verrà richiamato il file **(.krb),** il quale determinerà la malattia.

```
scelta_coltura($ans)
    Per quale coltura stai chiedendo la diagnosi?
    ---
    $ans = select_1
        1: Riso
        2: Mais
        3: Pomodoro
        4: Cotone
```

ESEMPIO DI OUTPUT E INTERAZIONE CON L'UTENTE

Come è possibile notare, in base alla coltura scelta, il sistema pone delle domande all'utente sul tipo di sintomo riscontrato. Sulla base delle risposte fornite, verrà indicata la malattia corrispondente o l'assenza di essa.

```
Per quale coltura stai chiedendo la diagnosi?
  1. Riso
  2. Mais
  3. Pomodoro
  4. Cotone
? [1-4] 3

Sono presenti delle macchie? (y/n) Y

Ci sono delle macchie scure sul pomodoro? (y/n) N
Disease: Leveillula_Taurica
```

7. INTERFACCIA GRAFICA

7.1 Funzione e Realizzazione

Per consentire all'utente di interagire con il sistema intelligente realizzato, è stata sviluppata un'interfaccia, che permettesse in modo semplice, intuitivo ed efficace di raggiungere ogni singolo obiettivo.

Per la realizzazione dell'interfaccia grafica, è stato necessario importare le seguenti librerie:

1°: è stata importata la libreria `tkinter`.

```
import tkinter as tk
from tkinter import ttk
```

2°: importazione di `partial` per passare i parametri alla funzione del bottone e del classificatore.

```
from functools import partial

import switch as switch
from sklearn.feature_extraction import DictVectorizer

from terrain_classifier import *
import csv
```

In seguito a tali importazioni, è stata creata la finestra, essenziale per l'interazione con l'utente.

```
window = tk.Tk()
window.geometry("600x450")
window.title("Form")
window.resizable(False, False)
window.configure(background="#67BD66")
```

Fra le altre, sono state implementate le seguenti funzionalità/controlli:

- Funzione per il bottone: tutti i dati inseriti vengono inviati ad una lista;
- Una volta inviato l'input, il box viene resettato;
- Controlli sugli input inseriti;
- Creazione bottoni;
- Font, label e box di testo dedicati alle varie unità di misura;
- Assegnazione dei dati inseriti nella matrice;
- Pop-up delle colture selezionate;
- Inserimento degli elementi grafici in base a ciò che contiene il dizionario;
- Richiamo al classificatore, passando la matrice come parametro;
- Inizializzazione del dizionario.

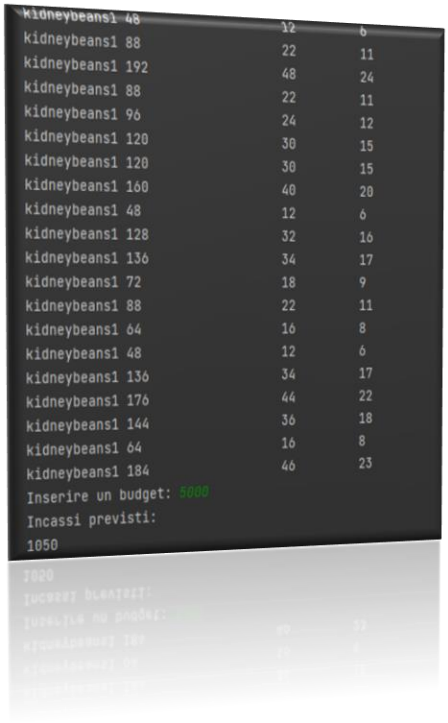
Vediamo, alcune schermate mostrate all'utente e le informazioni, con il supporto del dataset.

Menù di scelta – *Si consente all'utente di selezionare una delle opzioni disponibili*

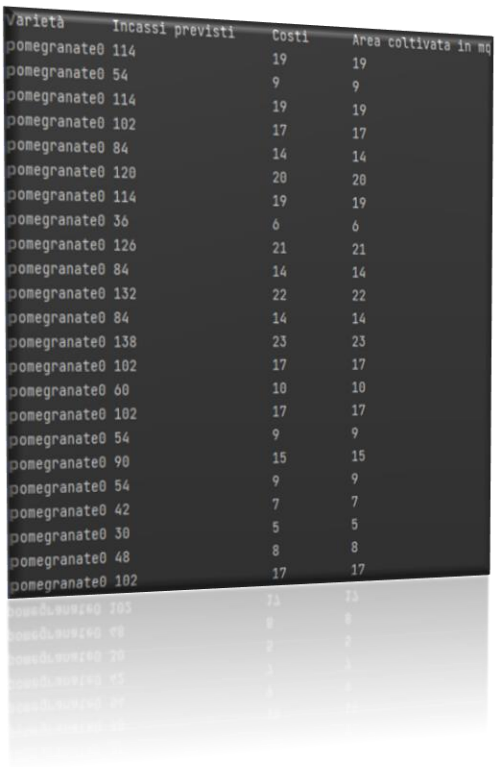
1° opzione del menù – **INSERIMENTO TERRENI,** *con dati inseriti correttamente*

INSERIMENTO TERRENI, *con dati inseriti erratamente*

COLTURE INSERITE – *A seconda dei dati inseriti, vengono mostrati i risultati prodotti.*



INSERIMENTO BUDGET – *Essenziale per problemi di ottimizzazione*



Esempio di stime effettuate su una singola coltura.

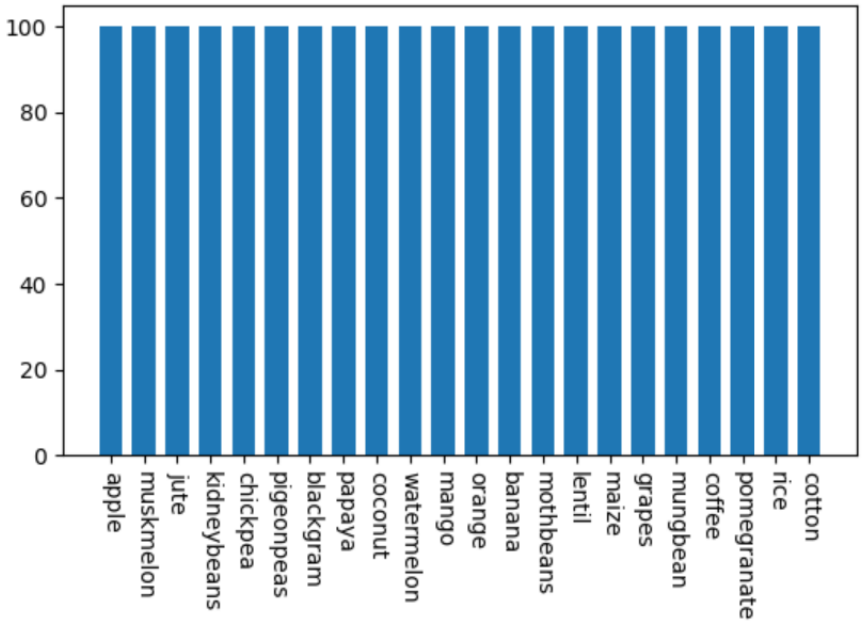
L'algoritmo **branch-and-bound** ci permette di consigliare all'utente le varietà da acquistare ed avere il giusto compromesso tra i costi di mantenimento e la produzione prevista.

8. ANALISI

Sulla base del dataset e le informazioni trattate, sono state messe a confronto le caratteristiche del dataset, per effettuare delle valutazioni più approfondite sulla struttura del dataset.

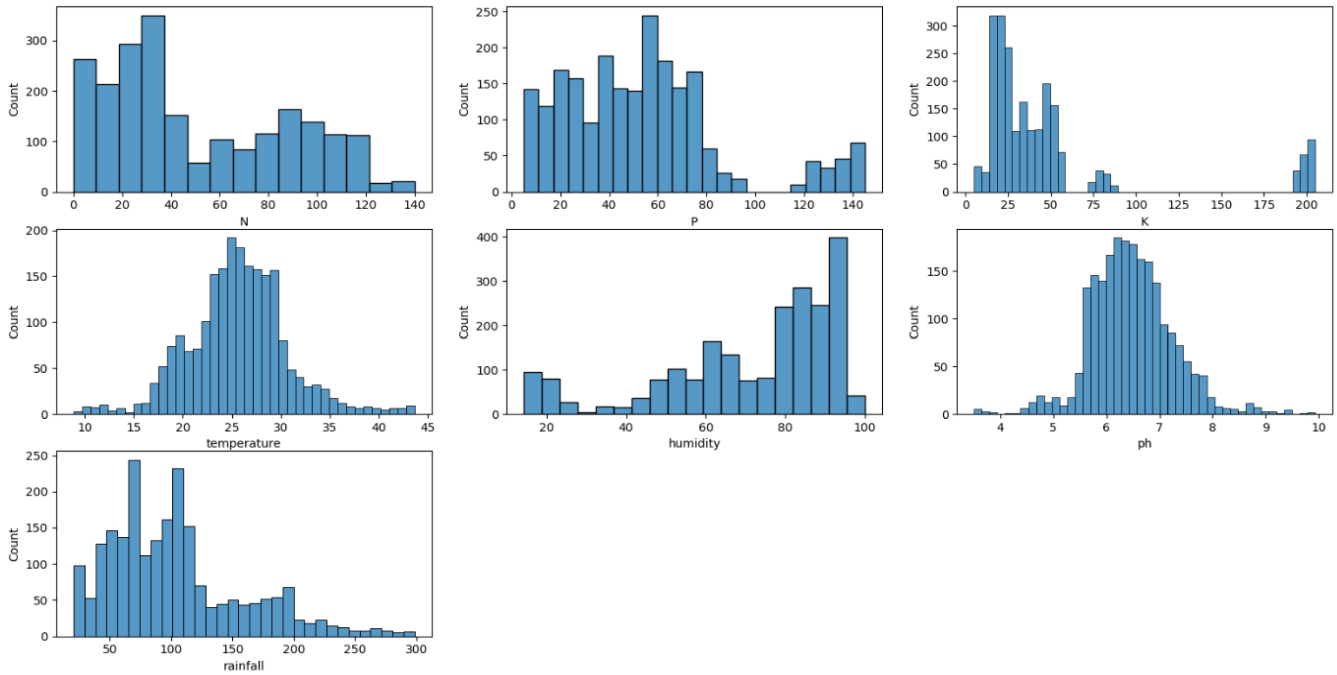
OSSERVAZIONI SUL DATASET:

Si può osservare come il dataset sia perfettamente bilanciato sulle varie etichette.

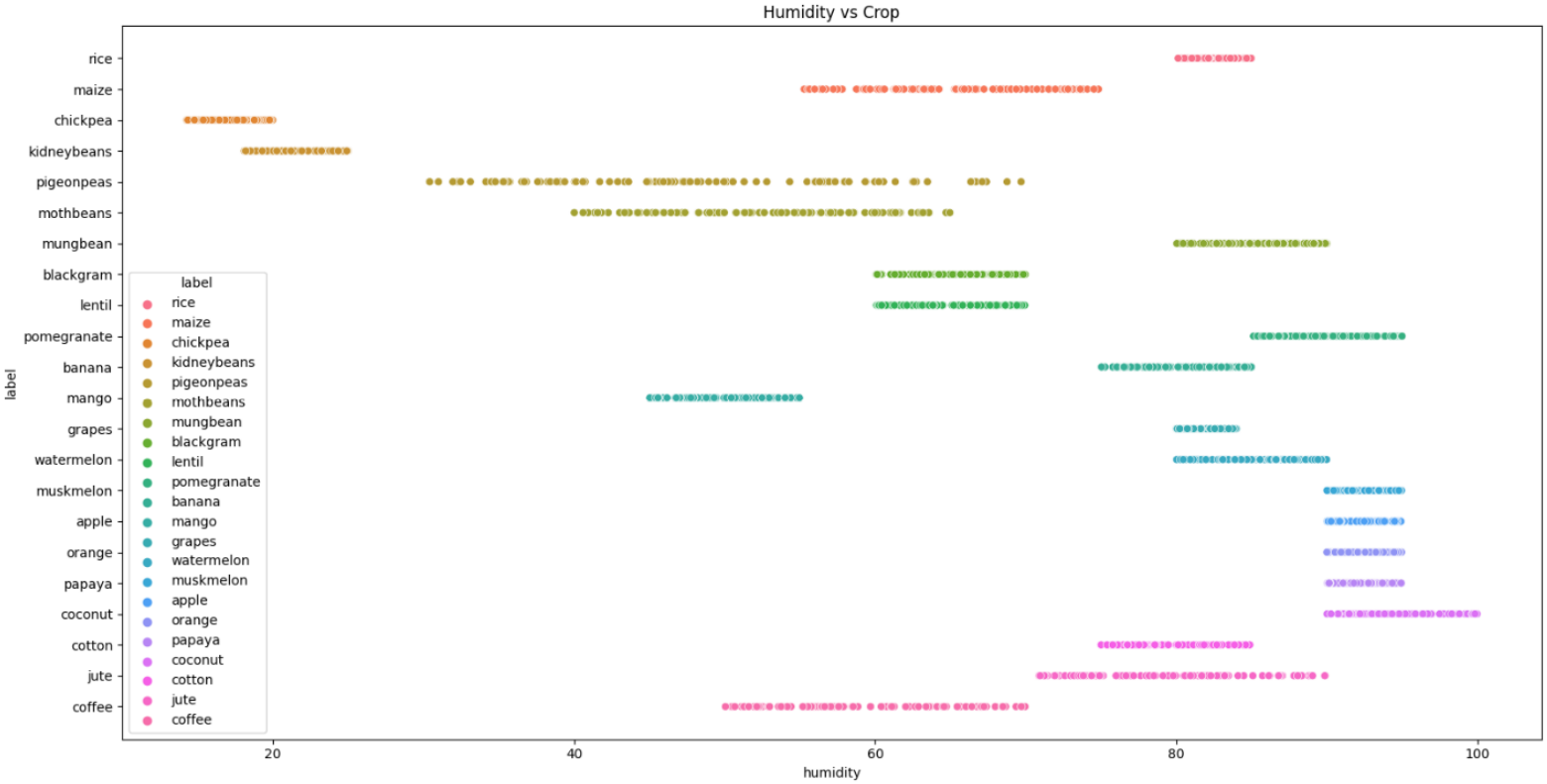


CARATTERISTICHE DELLE FEATURES:

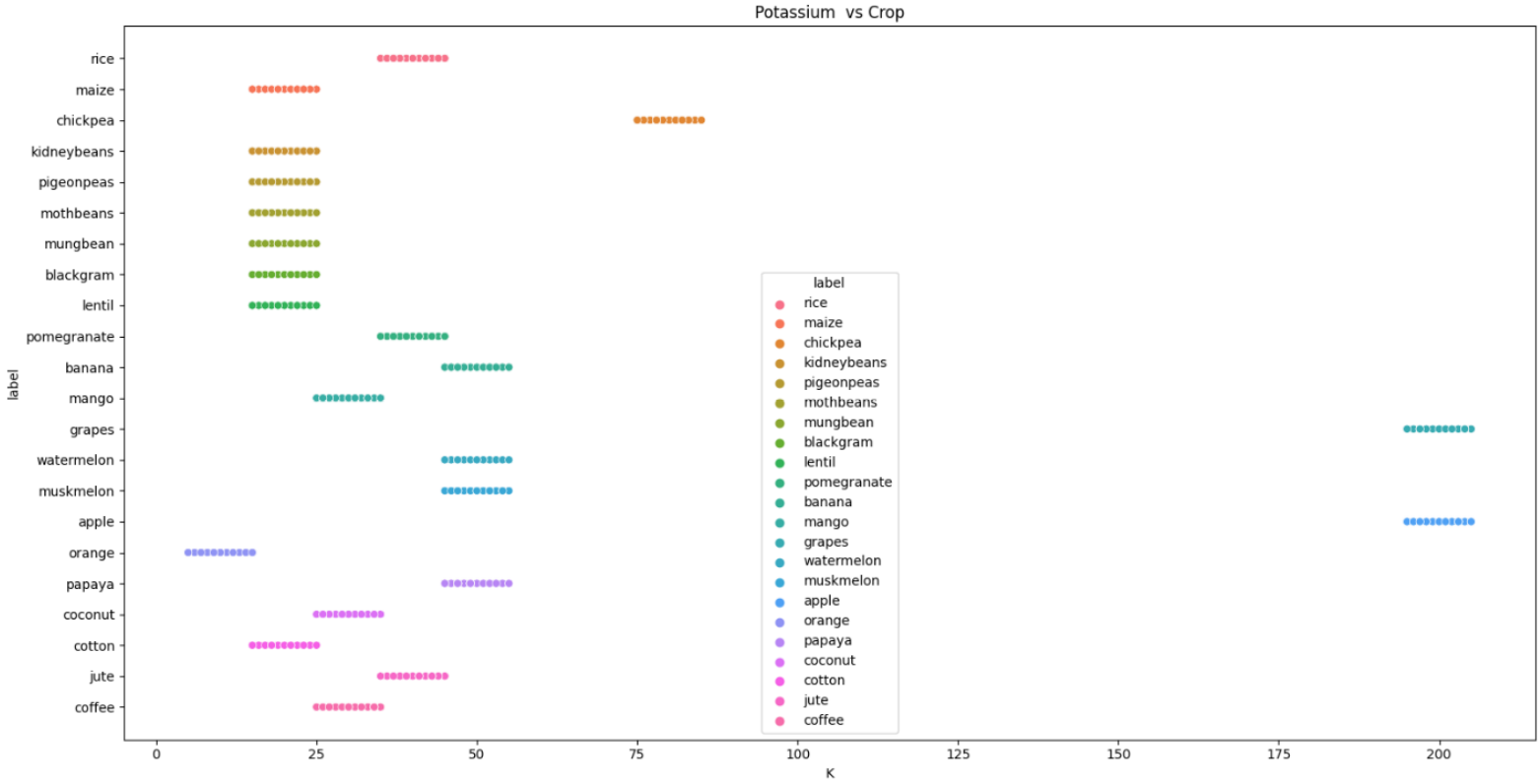
Viene osservata la frequenza dei valori delle features nel dataset



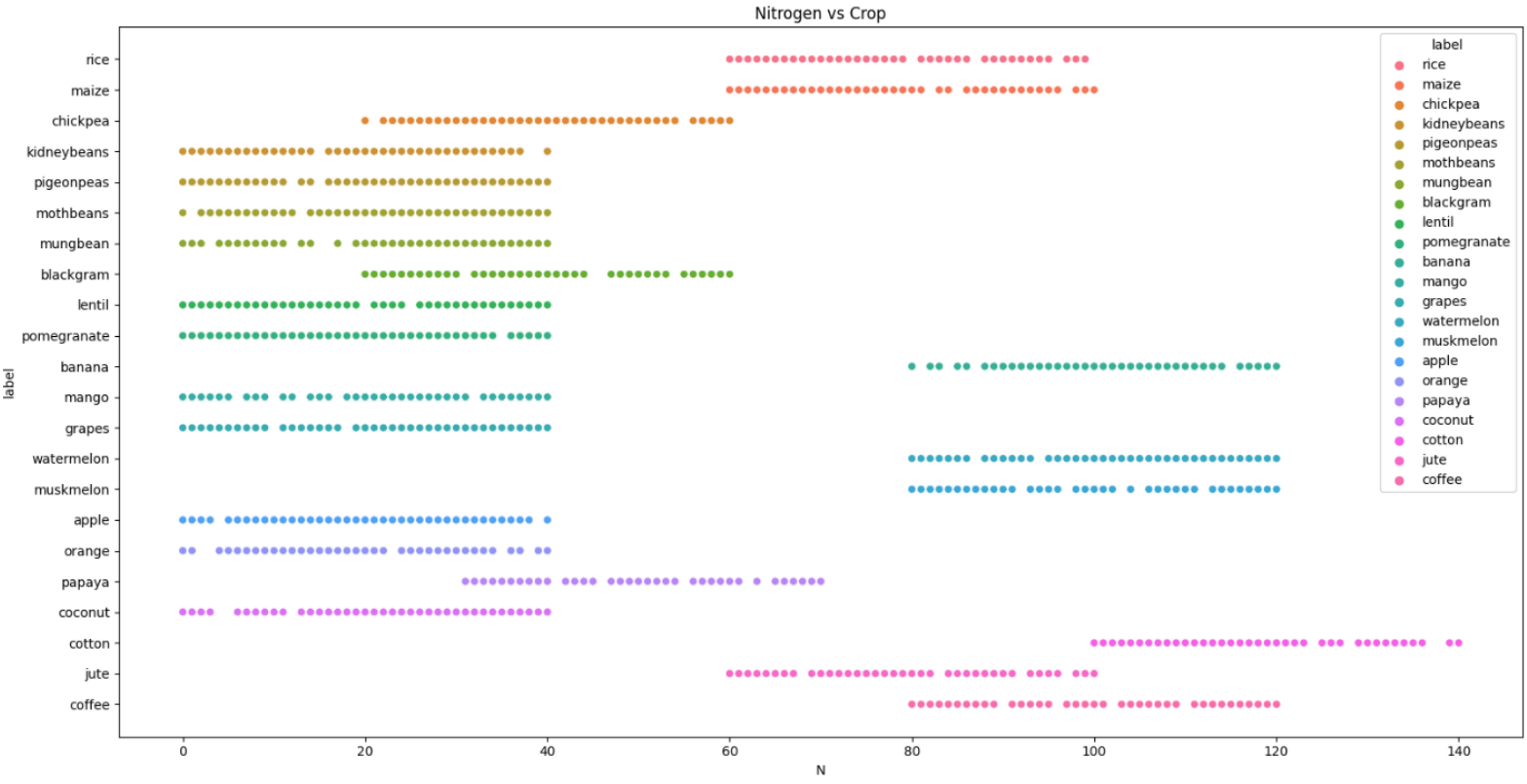
OSSERVAZIONE SULL'UMIDITA' DELLE COLTURE:



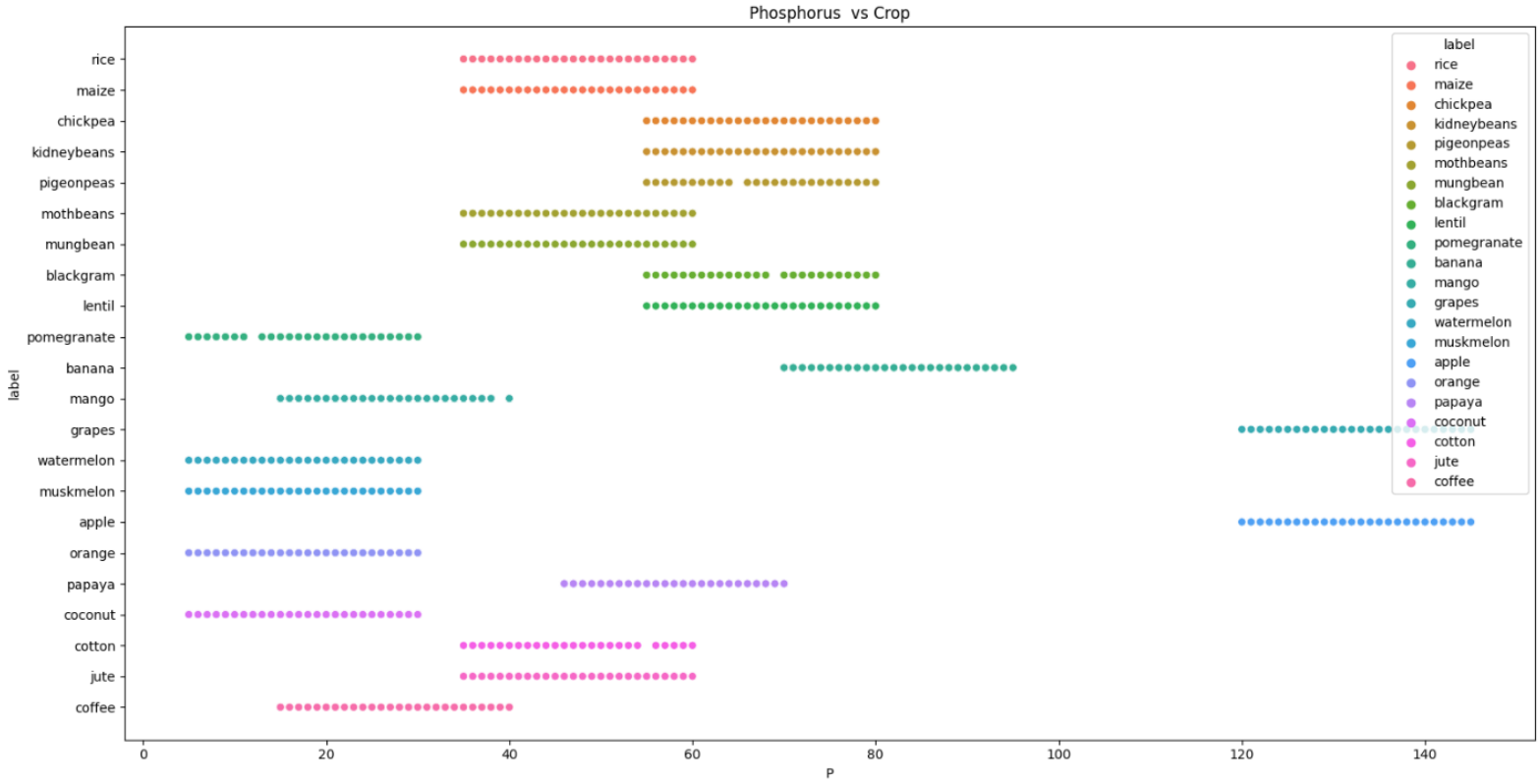
OSSERVAZIONI SUL POTASSIO (K):



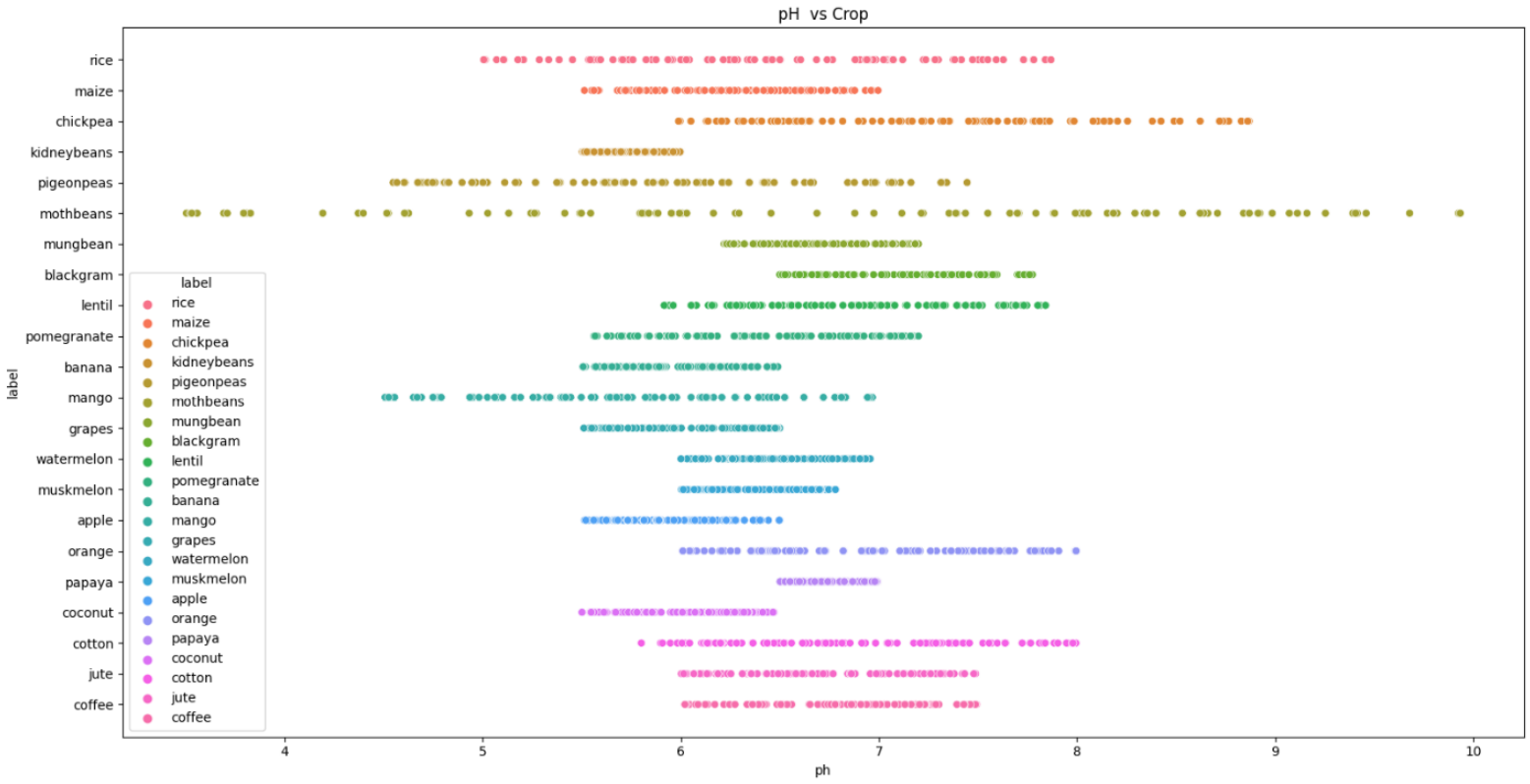
OSSERVAZIONE SULL'AZOTO (N) NELLE COLTURE:



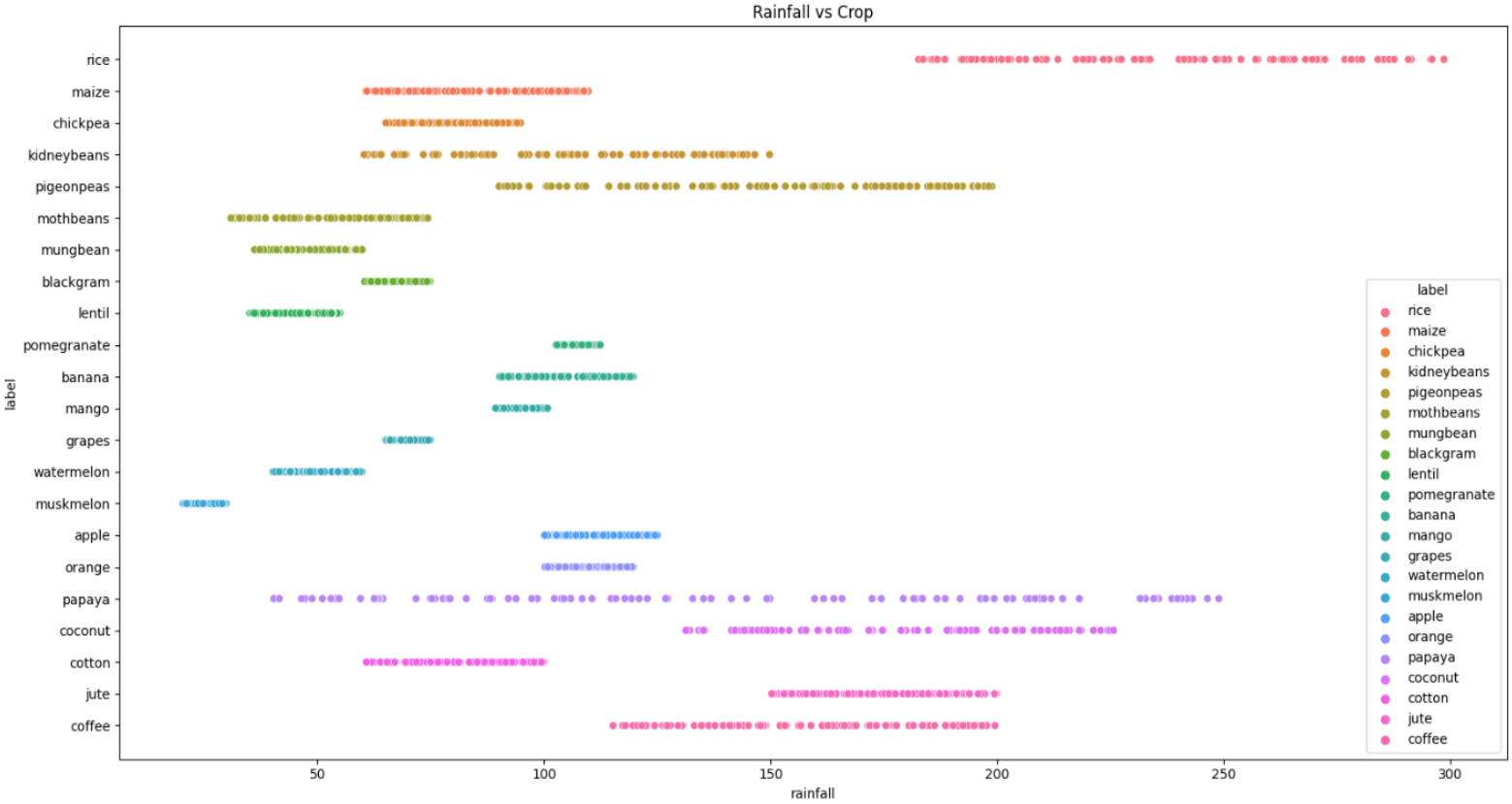
OSSERVAZIONE DEL FOSFORO (P) NELLE COLTURE:



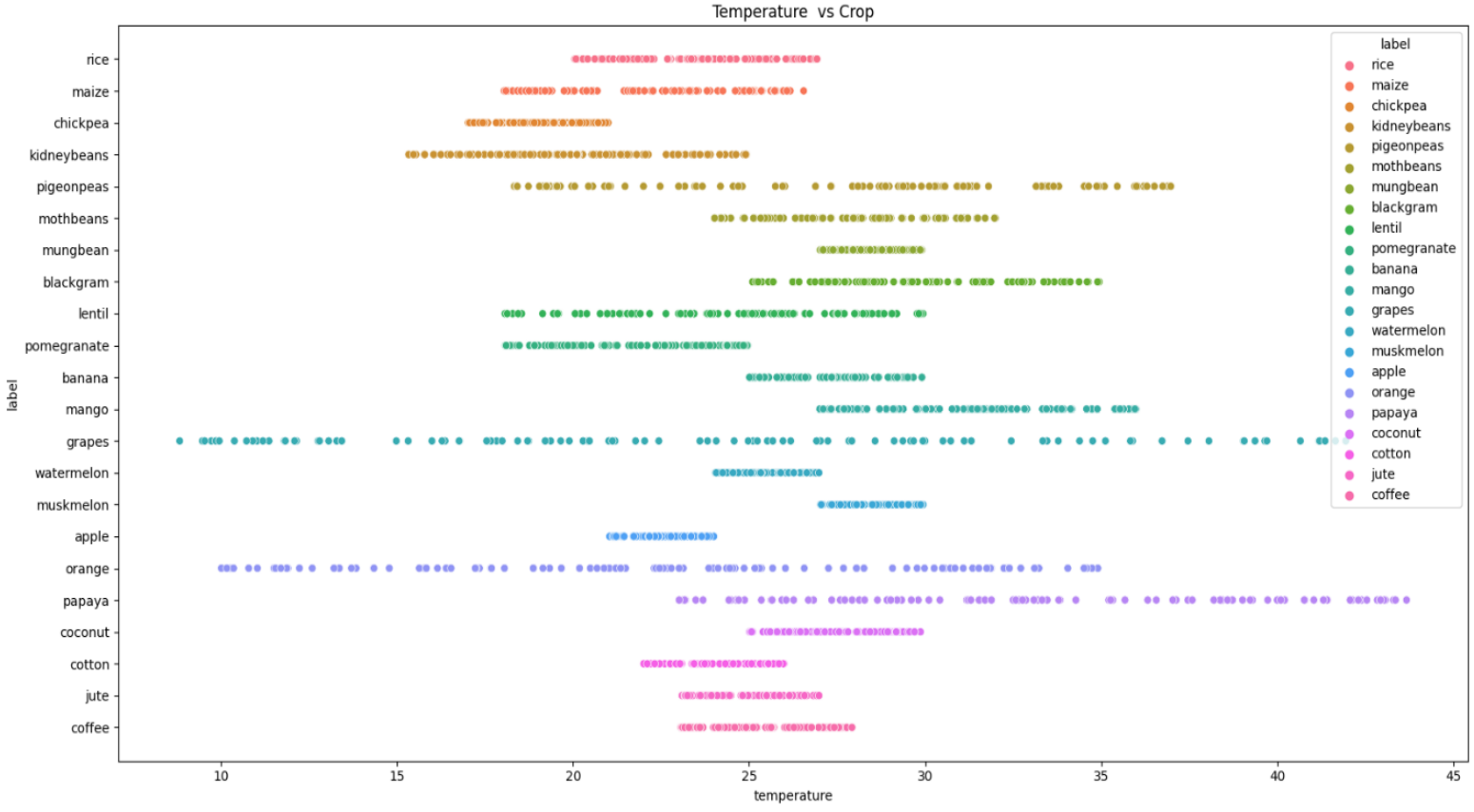
OSSERVAZIONE DEL PH NELLE COLTURE:



OSSERVAZIONE DELLE QUANTITA' DI PIOGGIE PER COLTURA:



OSSERVAZIONE DELLE TEMPERATURE PER COLTURA:



CONFRONTO PRESTAZIONE DEI DIVERSI MODELLI

Abbiamo confrontato i due modelli per verificare quello più accurato. Entrambi hanno buone prestazioni, ma il RANDOM FOREST ha fornito risultati leggermente migliori. Motivo per il quale abbiamo deciso di utilizzarlo.

RANDOM FOREST RESULTS

Random Forest(Depth=7) Results:

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	0.93	1.00	0.96	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	1.00	1.00	1.00	30
cotton	1.00	1.00	1.00	28
grapes	1.00	1.00	1.00	23
jute	0.87	1.00	0.93	34
kidneybeans	1.00	1.00	1.00	36
lentil	1.00	1.00	1.00	22
maize	1.00	1.00	1.00	26
mango	1.00	1.00	1.00	32
mothbeans	1.00	0.94	0.97	34
mungbean	1.00	1.00	1.00	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	1.00	1.00	1.00	37
pigeonpeas	1.00	1.00	1.00	37
pomegranate	1.00	1.00	1.00	38
rice	1.00	0.82	0.90	28
watermelon	1.00	1.00	1.00	23
accuracy			0.99	660
macro avg	0.99	0.99	0.99	660
weighted avg	0.99	0.99	0.99	660

KNN RESULTS:

Knn Results (K = 3):

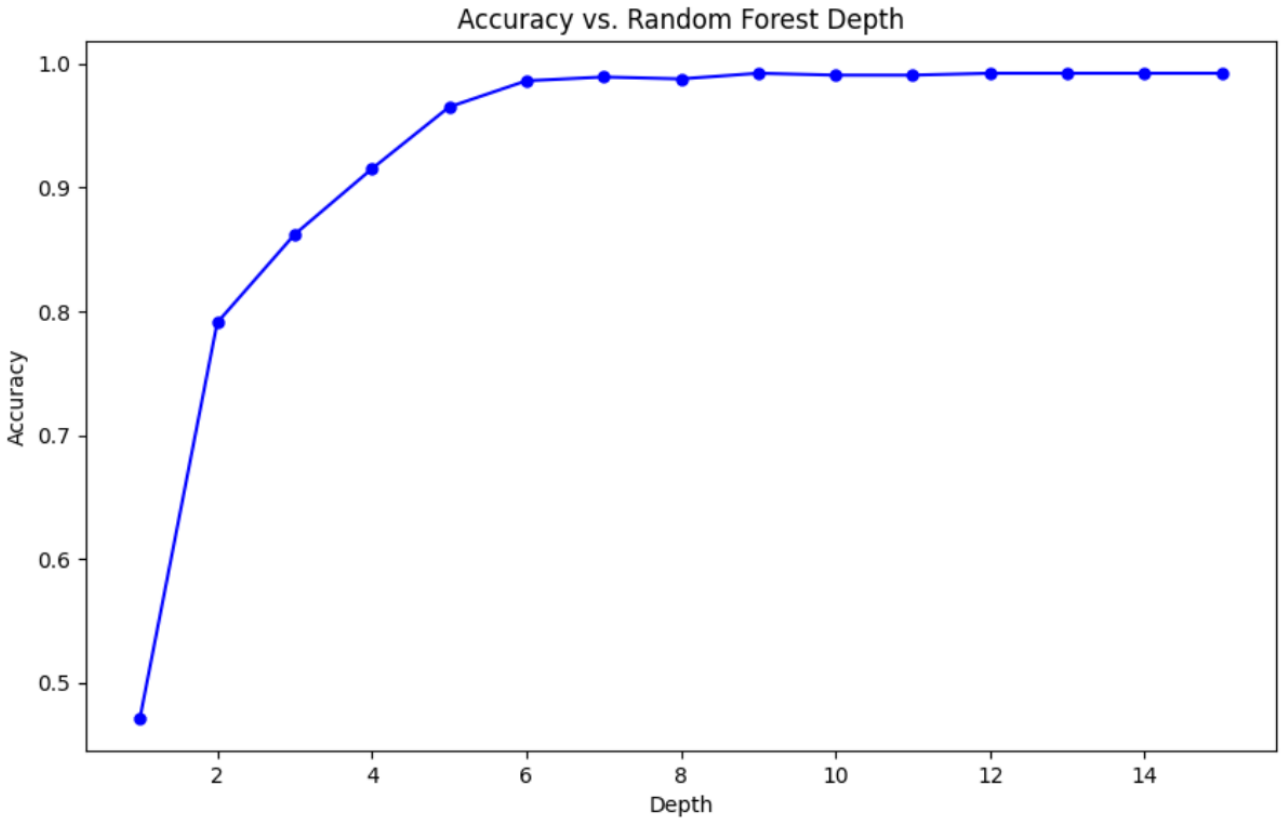
	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	1.00	1.00	1.00	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	1.00	0.97	0.98	30
cotton	0.96	0.96	0.96	28
grapes	1.00	1.00	1.00	23
jute	0.84	0.94	0.89	34
kidneybeans	1.00	1.00	1.00	36
lentil	1.00	1.00	1.00	22
maize	0.96	0.96	0.96	26
mango	1.00	1.00	1.00	32
mothbeans	1.00	1.00	1.00	34
mungbean	1.00	1.00	1.00	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	1.00	1.00	1.00	37
pigeonpeas	1.00	1.00	1.00	37
pomegranate	1.00	1.00	1.00	38
rice	0.92	0.82	0.87	28
watermelon	1.00	1.00	1.00	23
accuracy			0.98	660
macro avg	0.99	0.98	0.98	660
weighted avg	0.99	0.98	0.98	660

RANDOM FOREST DEPTH:

	Accuracy	Precision	Recall	Error Rate
Depth=3	0.86	0.91	0.88	0.137879
Depth=7	0.99	0.99	0.99	0.010606
Depth=13	0.99	0.99	0.99	0.007576

ACCURATEZZA vs RANDOM FOREST DEPTH

Questo grafico ci permette di notare i valori di accuratezza delle predizioni all'aumentare della profondità del Random Forest. Il valore ideale di profondità è compreso tra 6 e 8. Per tale ragione, abbiamo deciso di utilizzare il Random Forest con profondità 7.



9. CONCLUSIONI

In futuro, il progetto realizzato potrà essere utilizzato, ampliato e migliorato da aziende che operano nel settore, non solo regionale, ma anche nazionale e chissà…mondiale.

Il nostro obiettivo è sempre stato sempre quello di realizzare un sistema non solo efficiente, ma anche utile per le quotidiane mansioni lavorative di una singola azienda.

Funzionalità, quelle del sistema esperto, che consentono di automatizzare, migliorare e rendere più produttivo il lavoro imprenditoriale.

Per fare questo sarà necessario l'utilizzo del dataset. Ovviamente, anche la Knowledge Base usata è solo una minima parte, ma se ampliata permetterebbe di riconoscere e supportare l'utente nell'identificazione di più malattie.

Classificazione dei migliori terreni, ottimizzazione delle colture per un maggior profitto…è l'agricoltura del domani, è l'evoluzione non solo nell'ambito meccanico, ma anche agricolo.

Il Team ha solo rappresentato un modello di sistema intelligente, ma siamo sicuri che l'intelligenza artificiale e l'automazione saranno sempre più popolari nel mondo agricolo.

GRAZIE DELL'ATTENZIONE

SMART FARM

SISTEMA INTELLIGENTE PER LA CLASSIFICAZIONE E OTTIMIZZAZIONE DELLE COLTURE

–

GESTIONE DELLE MALATTIE ATTRAVERSO UNA KNOWLEDGE BASE

TEAM SmartFarm