Descripción del Dataset

El conjunto de datos contiene información sobre condiciones del suelo y clima, con el objetivo de predecir qué cultivo sería el más adecuado.

Cada fila representa una muestra de un lote, y contiene los siguientes atributos:

- `N`: Contenido de **nitrógeno** en el suelo (ppm)
- `P`: Contenido de **fósforo** en el suelo (ppm)
- `K`: Contenido de **potasio** en el suelo (ppm)
- `temperature`: Temperatura ambiente (°C)
- `humidity`: Humedad relativa (%)
- `ph`: Nivel de acidez del suelo
- `rainfall`: Precipitación anual (mm)
- `label`: 🌱 **Cultivo recomendado** (variable objetivo)

Este es un problema de **clasificación multiclase**, ya que la variable `label` puede tomar varios cultivos como valores posibles.

Comparación de Modelos

Modelo	Accuracy
Random Forest	99.55%
K-Nearest Neighbors (KNN)	97.73%
Support Vector Machine (SVM)	98.41%

> ✅ *Random Forest fue el modelo con mejor rendimiento. Además, es más estable ante valores atípicos y menos sensible a la escala que KNN o SVM, lo que lo hace ideal para este tipo de datos.*

Simulación de Recomendación de Cultivo

El bloque final permite simular una predicción personalizada, modificando las condiciones del suelo y el clima para obtener el cultivo más adecuado.

> ideal para entender cómo responden los modelos entrenados.

Conclusiones Finales

En este proyecto se abordó el problema de **recomendar el tipo de cultivo más adecuado** en función de variables como nutrientes del suelo, temperatura, humedad, pH y precipitaciones.

Luego de analizar el dataset y aplicar distintos modelos de clasificación, se concluyó que:

- El dataset se encuentra **balanceado**, lo cual favorece un entrenamiento justo para todas las clases.
- A nivel de correlación, la mayoría de las variables aportan información **independiente**, ideal para modelos predictivos.
- **Random Forest** fue el modelo con mejor desempeño, alcanzando un **accuracy del 99.55%**, superando a KNN (97.73%) y SVM (98.41%).

Este tipo de sistemas pueden resultar útiles para agricultores o sistemas de apoyo a la decisión, ayudando a optimizar el uso del suelo según las condiciones del entorno.

> ✨ *Como cierre, este proyecto permitió poner en práctica herramientas fundamentales de Data Science y Machine Learning, integrando análisis, visualización y evaluación de modelos en un flujo coherente y personalizado.*

> 🤝 *Se entrenó un modelo K-Nearest Neighbors (KNN) con `k=5`. Este modelo clasifica en base a los cultivos más cercanos según las condiciones del suelo y clima. A continuación se presentan los resultados.*

> 🧠 *Se entrenó un modelo Random Forest con 100 árboles. Este modelo se destaca por ser eficaz con datos tabulares y manejar bien variables categóricas sin requerir mucha preparación de datos. A continuación se muestran las métricas de rendimiento obtenidas.*

Este proyecto fue realizado como trabajo final del programa **DataUy – Formación Movistar**.

A través de este caso práctico, se integraron conceptos clave de análisis exploratorio, visualización, modelos predictivos y evaluación de resultados en un entorno aplicado y funcional.

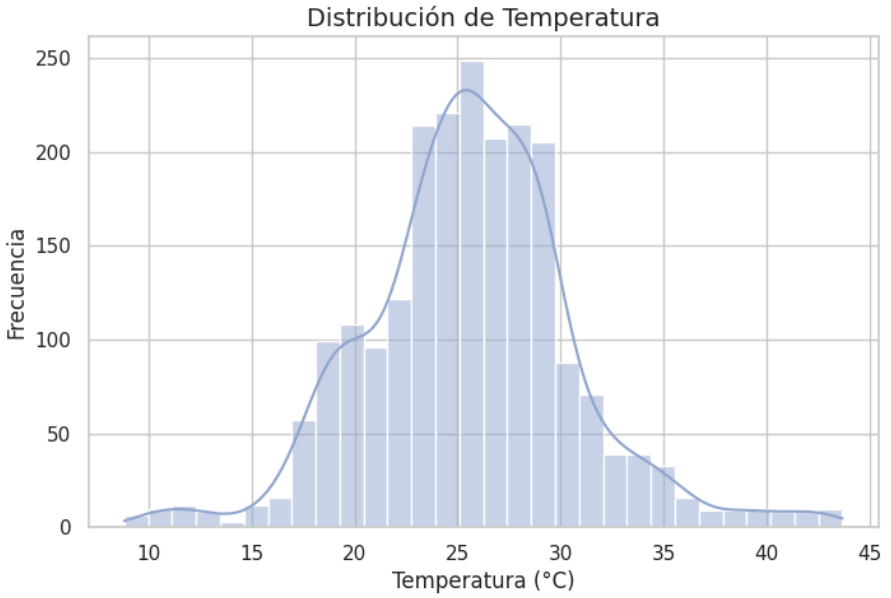
🎓 **Autora:** Victoria Crosina
📍 **Montevideo, Uruguay**

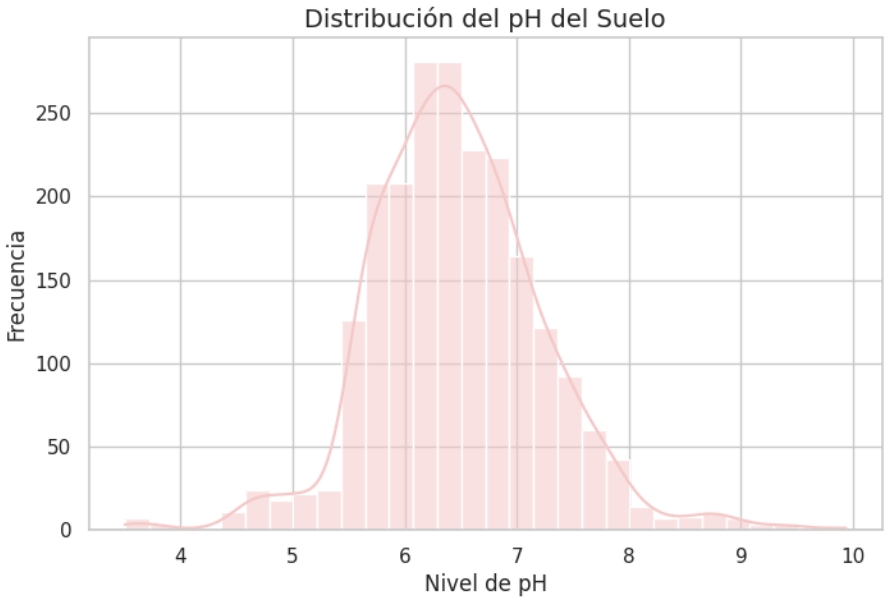
Gracias por acompañar 🌸

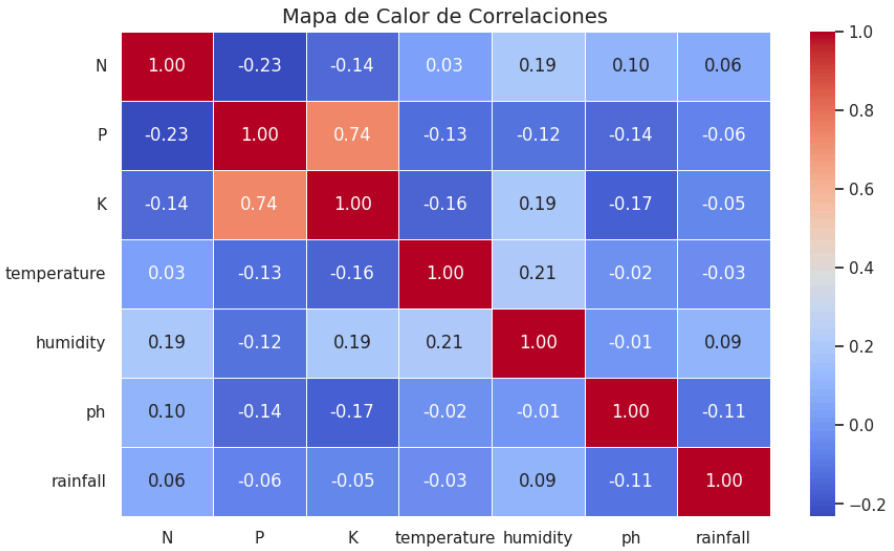
```
# 🌾 Formación MOVISTAR DataUy- Recomendación de Cultivos
### Proyecto Final - Programa DataUy
#### ✨ Autora: Victoria Crosina

#####  Este proyecto tiene como objetivo aplicar las herramientas vistas en el programa
######  Se busca no solo generar un modelo predictivo eficaz, sino también realizar un

# 📚 Cargo las librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# No quiero ver alertas que ensucian mucho el código
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module="IPython.core.pylabtools
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)


# Estilo (colores Glam Pastel Chic ✨)
sns.set(style="whitegrid", palette=["#F7CAC9", "#92A8D1", "#F4E1D2", "#C1B1A1", "#B2B2B

# Cargo el dataset desde el repositorio de ActiveSoft-uy (Es mi repo personal el de Act
url = "https://raw.githubusercontent.com/ActiveSoft-uy/datauy-proyecto/main/Crop_recomm
df = pd.read_csv(url)

# Muestra las primeras filas
df.head()


# 🌡️Temperatura
plt.figure(figsize=(8, 5))
sns.histplot(df['temperature'], bins=30, kde=True, color="#92A8D1")
plt.title("Distribución de Temperatura", fontsize=14)
plt.xlabel("Temperatura (°C)")
plt.ylabel("Frecuencia")
plt.grid(True)
plt.show()

# 💧 ph
plt.figure(figsize=(8, 5))
sns.histplot(df['ph'], bins=30, kde=True, color="#F7CAC9")
plt.title("Distribución del pH del Suelo", fontsize=14)
plt.xlabel("Nivel de pH")
plt.ylabel("Frecuencia")
plt.grid(True)
plt.show()

# 🔥 Mapa de calor
plt.figure(figsize=(10, 6))
correlation = df.drop("label", axis=1).corr()
sns.heatmap(correlation, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Mapa de Calor de Correlaciones", fontsize=14)
plt.show()

# 🌱 Conteo de cultivos
plt.figure(figsize=(12,6))
sns.countplot(
    x='label',
    data=df,
```

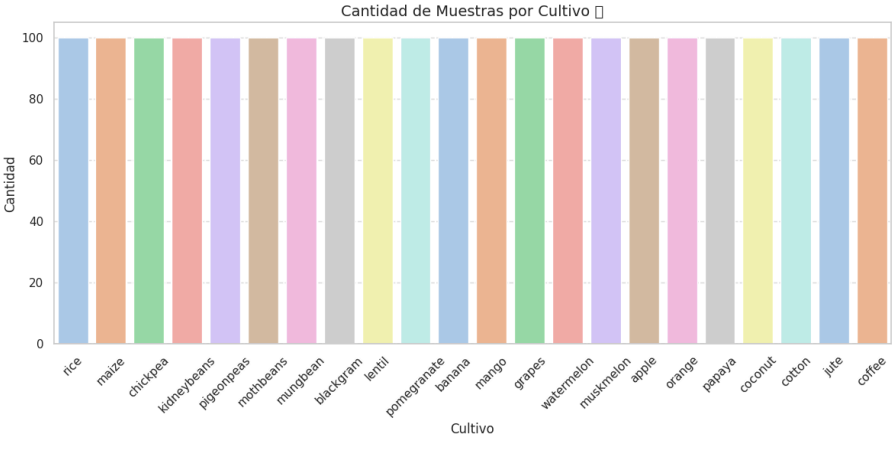
```
    order=df['label'].value_counts().index,
    hue='label',
    palette='pastel',
    dodge=False,
    legend=False
)
plt.title("Cantidad de Muestras por Cultivo 🌿", fontsize=14)
plt.xlabel("Cultivo")
plt.ylabel("Cantidad")
plt.xticks(rotation=45)
plt.grid(True, axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()


# Boxplot de pH por tipo de cultivo
plt.figure(figsize=(12,6))
sns.boxplot(x='label', y='ph', data=df, palette="pastel")
plt.title("Distribución del pH según Cultivo", fontsize=14)
plt.xticks(rotation=45)
plt.ylabel("pH del Suelo")
plt.xlabel("Cultivo")
plt.grid(True, axis='y', linestyle='--', alpha=0.6)
plt.show()

# Datos para el modelo predictivo
# variables predictoras (X) y variable objetivo (y)
X = df.drop('label', axis=1)
y = df['label']

# Entrenamiento y prueba
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    stratify=y)

print(f"Tamaño entrenamiento: {X_train.shape[0]} muestras")
print(f"Tamaño prueba: {X_test.shape[0]} muestras")

############################## INICIO Random Forest ###################################

# Entrenamiento
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Creación del modelo
model_rf = RandomForestClassifier(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)

# Evaluación
y_pred = model_rf.predict(X_test)

# Reporte
print("Reporte de Clasificación:\n")
print(classification_report(y_test, y_pred))

############################## FIN Random Forest ###################################


########################### Inicio Modelo KNN ###################################

# Modelo KNN para la comparación de modelos
from sklearn.neighbors import KNeighborsClassifier

# Entrenamiento
model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_train, y_train)

# Predicciones y evaluación
y_pred_knn = model_knn.predict(X_test)

print("KNN - Reporte de Clasificación:\n")
print(classification_report(y_test, y_pred_knn))

############################## FIN Modelo KNN ###################################


########################### Inicio Modelo SVC ###################################

from sklearn.svm import SVC
```

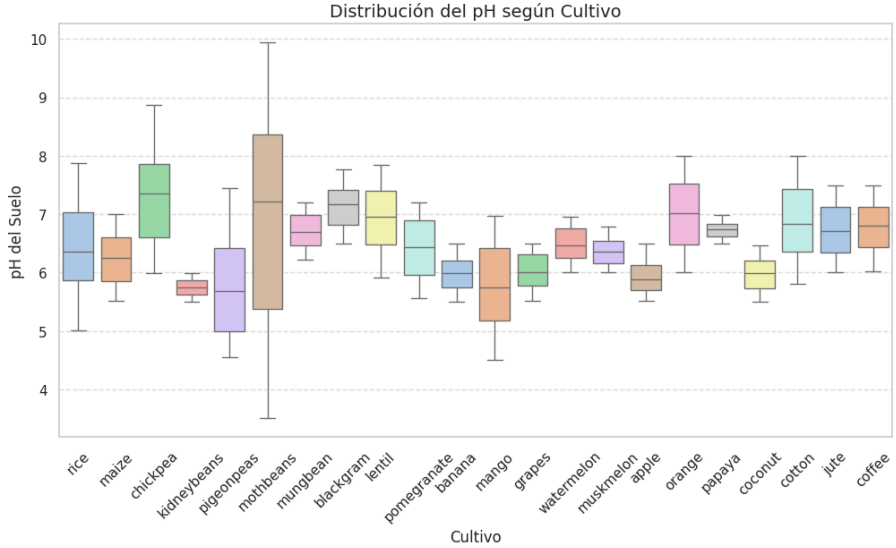
```
# Entrenamiento
model_svm = SVC(kernel='rbf', random_state=42)
model_svm.fit(X_train, y_train)

# Predicciones y evaluación
y_pred_svm = model_svm.predict(X_test)

print("SVM - Reporte de Clasificación:\n")
print(classification_report(y_test, y_pred_svm))

############################## FIN Modelo SVC ###################################

# Calculo la presición por modelo...
from sklearn.metrics import accuracy_score

acc_rf = accuracy_score(y_test, y_pred)
acc_knn = accuracy_score(y_test, y_pred_knn)
acc_svm = accuracy_score(y_test, y_pred_svm)

print("Accuracy por modelo:")
print(f"Random Forest: {acc_rf:.4f}")
print(f"KNN:           {acc_knn:.4f}")
print(f"SVM:           {acc_svm:.4f}")

# Comparación visual
import matplotlib.pyplot as plt

# Datos con los resultados
modelos = ['Random Forest', 'KNN', 'SVM']
accuracy = [0.9955, 0.9773, 0.9841]

# Colores
colores = ['#F7CAC9', '#92A8D1', '#C1B1A1']

plt.figure(figsize=(8, 5))
bars = plt.bar(modelos, accuracy, color=colores)
plt.ylim(0.95, 1.0)
plt.ylabel('Accuracy')
plt.title('Comparación de Accuracy por Modelo', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.6)

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.001, f'{yval:.2%}', ha='center',

plt.show()

########################################## PREDICCION FINAL ############################
# Predicción interactiva de cultivo
#(la idea es que quien entre puede cambiar los valores como quiera
#y que le muestre una predicción como para darle un cierre a la entrega)
#El ejemplo es para Arroz..

mi_muestra = pd.DataFrame([{
    'N': 85,
    'P': 55,
    'K': 45,
    'temperature': 24.0,
    'humidity': 75.0,
    'ph': 6.2,
    'rainfall': 220.0
}])

# Predicción
cultivo = model_rf.predict(mi_muestra)[0]

# Resultado
print("🧾 Condiciones ingresadas:")
display(mi_muestra)
print("\n🌱 Cultivo recomendado por el modelo:", cultivo)
```


Distribución de Temperatura
Frecuencia
Temperatura (°C)


Distribución del pH del Suelo
Frecuencia
Nivel de pH


Mapa de Calor de Correlaciones
N 1.00 -0.23 -0.14 0.03 0.19 0.10 0.06
P -0.23 1.00 0.74 -0.13 -0.12 -0.14 -0.06
K -0.14 0.74 1.00 -0.16 0.19 -0.17 -0.05
temperature 0.03 -0.13 -0.16 1.00 0.21 -0.02 -0.03
humidity 0.19 -0.12 0.19 0.21 1.00 -0.01 0.09
ph 0.10 -0.14 -0.17 -0.02 -0.01 1.00 -0.11
rainfall 0.06 -0.06 -0.05 -0.03 0.09 -0.11 1.00
N P K temperature humidity ph rainfall


Cantidad de Muestras por Cultivo
Cantidad
rice maize chickpea kidneybeans pigeonpeas mothbeans mungbean blackgram lentil pomegranate banana mango grapes watermelon muskmelon apple orange papaya coconut cotton jute coffee
Cultivo



Tamaño entrenamiento: 1760 muestras
Tamaño prueba: 440 muestras
Reporte de Clasificación:

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	20
banana	1.00	1.00	1.00	20
blackgram	1.00	0.95	0.97	20
chickpea	1.00	1.00	1.00	20
coconut	1.00	1.00	1.00	20
coffee	1.00	1.00	1.00	20
cotton	1.00	1.00	1.00	20
grapes	1.00	1.00	1.00	20
jute	0.95	1.00	0.98	20
kidneybeans	1.00	1.00	1.00	20
lentil	1.00	1.00	1.00	20
maize	0.95	1.00	0.98	20
mango	1.00	1.00	1.00	20
mothbeans	1.00	1.00	1.00	20
mungbean	1.00	1.00	1.00	20
muskmelon	1.00	1.00	1.00	20
orange	1.00	1.00	1.00	20
papaya	1.00	1.00	1.00	20
pigeonpeas	1.00	1.00	1.00	20
pomegranate	1.00	1.00	1.00	20
rice	1.00	0.95	0.97	20
watermelon	1.00	1.00	1.00	20
accuracy			1.00	440
macro avg	1.00	1.00	1.00	440
weighted avg	1.00	1.00	1.00	440

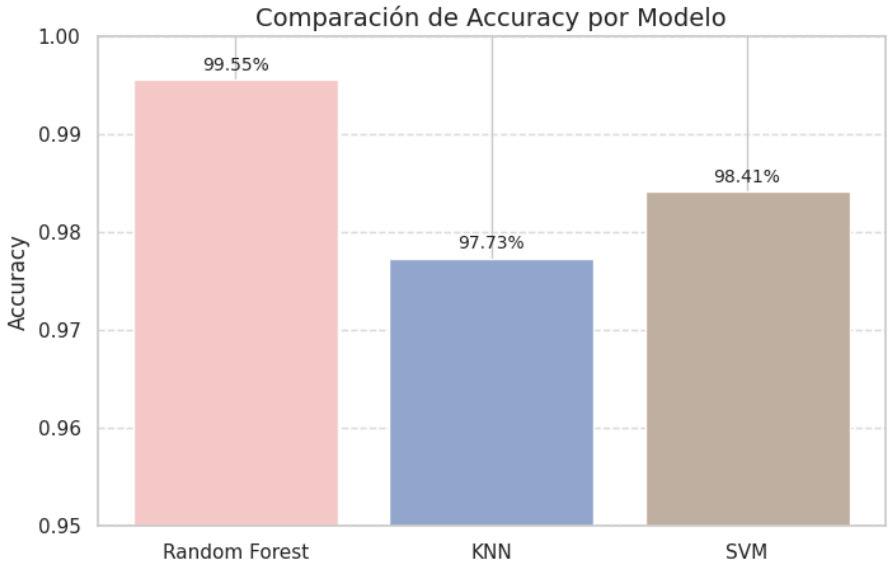
KNN - Reporte de Clasificación:

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	20
banana	1.00	1.00	1.00	20
blackgram	0.91	1.00	0.95	20
chickpea	1.00	1.00	1.00	20
coconut	1.00	1.00	1.00	20
coffee	1.00	1.00	1.00	20
cotton	0.95	1.00	0.98	20
grapes	1.00	1.00	1.00	20
jute	0.78	0.90	0.84	20
kidneybeans	1.00	1.00	1.00	20
lentil	1.00	1.00	1.00	20
maize	1.00	0.95	0.97	20
mango	1.00	1.00	1.00	20
mothbeans	1.00	0.95	0.97	20
mungbean	1.00	1.00	1.00	20
muskmelon	1.00	1.00	1.00	20
orange	1.00	1.00	1.00	20
papaya	1.00	1.00	1.00	20
pigeonpeas	1.00	0.95	0.97	20
pomegranate	1.00	1.00	1.00	20
rice	0.88	0.75	0.81	20
watermelon	1.00	1.00	1.00	20
accuracy			0.98	440
macro avg	0.98	0.98	0.98	440
weighted avg	0.98	0.98	0.98	440

SVM - Reporte de Clasificación:

	precision	recall	f1-score	support

```
       apple       1.00      1.00      1.00        20
      banana       1.00      1.00      1.00        20
   blackgram       0.95      1.00      0.98        20
    chickpea       1.00      1.00      1.00        20
     coconut       1.00      1.00      1.00        20
      coffee       1.00      1.00      1.00        20
      cotton       0.95      1.00      0.98        20
      grapes       1.00      1.00      1.00        20
        jute       0.80      1.00      0.89        20
 kidneybeans       1.00      1.00      1.00        20
      lentil       1.00      1.00      1.00        20
       maize       1.00      0.95      0.97        20
       mango       1.00      1.00      1.00        20
   mothbeans       1.00      1.00      1.00        20
    mungbean       1.00      1.00      1.00        20
   muskmelon       1.00      1.00      1.00        20
      orange       1.00      1.00      1.00        20
      papaya       1.00      1.00      1.00        20
  pigeonpeas       1.00      0.95      0.97        20
 pomegranate       1.00      1.00      1.00        20
        rice       1.00      0.75      0.86        20
  watermelon       1.00      1.00      1.00        20

    accuracy                           0.98       440
   macro avg       0.99      0.98      0.98       440
weighted avg       0.99      0.98      0.98       440

Accuracy por modelo:
Random Forest: 0.9955
KNN:           0.9773
SVM:           0.9841
```



Condiciones ingresadas:

	N	P	K	temperature	humidity	ph	rainfall
0	85	55	45	24.0	75.0	6.2	220.0

Cultivo recomendado por el modelo: rice