*# %% [markdown]*

"""

# Análise e Modelagem de Produtos Agrícolas

Este notebook tem como objetivo:

- Explorar uma base de dados contendo informações de solo e clima;

- Realizar uma análise descritiva (incluindo pelo menos cinco gráficos);

- Identificar o perfil ideal de condições para diferentes culturas;

- Desenvolver e comparar cinco modelos preditivos (Random Forest, SVM, KNN, Regressão Logística e Redes Neurais) que indiquem o melhor produto agrícola a ser cultivado.

"""

*# %% [code]*

*# Importação das bibliotecas necessárias*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%matplotlib** inline

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV

**from** sklearn.metrics **import** accuracy\_score, confusion\_matrix, classification\_report

*# Modelos:*

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.svm **import** SVC

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.neural\_network **import** MLPClassifier

In [28]:

*# %% [markdown]*

"""

## 1. Carregamento e Pré-processamento dos Dados

Nesta seção, vamos ler a base de dados, verificar o tipo de cada variável, identificar valores nulos e aplicar eventuais pré-processamentos necessários.

"""

*# %% [code]*

*# Carregar a base de dados*

df **=** pd**.**read\_csv("Atividade\_Cap\_14\_produtos\_agricolas.csv")

*# Visualizar as primeiras linhas*

print(df**.**head())

*# Informações gerais do dataset*

print(df**.**info())

*# Estatísticas descritivas*

print(df**.**describe())

*# Verificar valores nulos*

print("Valores nulos por coluna:")

print(df**.**isnull()**.**sum())

N P K temperature humidity ph rainfall label

0 90 42 43 20.879744 82.002744 6.502985 202.935536 rice

1 85 58 41 21.770462 80.319644 7.038096 226.655537 rice

2 60 55 44 23.004459 82.320763 7.840207 263.964248 rice

3 74 35 40 26.491096 80.158363 6.980401 242.864034 rice

4 78 42 42 20.130175 81.604873 7.628473 262.717340 rice

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2200 entries, 0 to 2199

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 N 2200 non-null int64

1 P 2200 non-null int64

2 K 2200 non-null int64

3 temperature 2200 non-null float64

4 humidity 2200 non-null float64

5 ph 2200 non-null float64

6 rainfall 2200 non-null float64

7 label 2200 non-null object

dtypes: float64(4), int64(3), object(1)

memory usage: 137.6+ KB

None

N P K temperature humidity \

count 2200.000000 2200.000000 2200.000000 2200.000000 2200.000000

mean 50.551818 53.362727 48.149091 25.616244 71.481779

std 36.917334 32.985883 50.647931 5.063749 22.263812

min 0.000000 5.000000 5.000000 8.825675 14.258040

25% 21.000000 28.000000 20.000000 22.769375 60.261953

50% 37.000000 51.000000 32.000000 25.598693 80.473146

75% 84.250000 68.000000 49.000000 28.561654 89.948771

max 140.000000 145.000000 205.000000 43.675493 99.981876

ph rainfall

count 2200.000000 2200.000000

mean 6.469480 103.463655

std 0.773938 54.958389

min 3.504752 20.211267

25% 5.971693 64.551686

50% 6.425045 94.867624

75% 6.923643 124.267508

max 9.935091 298.560117

Valores nulos por coluna:

N 0

P 0

K 0

temperature 0

humidity 0

ph 0

rainfall 0

label 0

dtype: int64

In [29]:

*# %% [markdown]*

"""

## 2. Análise Exploratória de Dados

Vamos visualizar as distribuições das variáveis e explorar as relações entre elas. Aqui você pode incluir, por exemplo, histogramas, boxplots, gráficos de dispersão e uma matriz de correlação.

"""

*# Carregar a base de dados*

df **=** pd**.**read\_csv("Atividade\_Cap\_14\_produtos\_agricolas.csv")

*# %% [code]*

*# Histogramas das variáveis numéricas*

df**.**hist(figsize**=**(12, 8))

plt**.**tight\_layout()

plt**.**show()

*# %% [code]*

*# Boxplots para visualizar outliers*

plt**.**figure(figsize**=**(12, 8))

sns**.**boxplot(data**=**df)

plt**.**xticks(rotation**=**45)

plt**.**title("Boxplots das variáveis")

plt**.**show()

*# %% [code]*

*# Heatmap de correlação*

*# plt.figure(figsize=(10, 6))*

*# sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")*

*# plt.title("Matriz de Correlação")*

*# plt.show()*

*# %% [code]*

*# Gráfico de dispersão exemplo: pH vs rainfall*

plt**.**figure(figsize**=**(12, 9))

sns**.**scatterplot(x**=**"ph", y**=**"rainfall", data**=**df, hue**=**"label")

plt**.**title("Dispersão: pH vs Precipitação")

plt**.**show()

*# %% [code]*

*# Distribuição da variável alvo "label"*

plt**.**figure(figsize**=**(12, 6))

sns**.**countplot(x**=**"label", data**=**df)

plt**.**title("Frequência dos Tipos de Cultura")

plt**.**xticks(rotation**=**45)

plt**.**show()

In [30]:

*# %% [markdown]*

"""

## 3. Definição do Perfil Ideal e Análise Comparativa

Nesta seção, você pode agrupar os dados por cultura (label) e calcular medidas como média, mediana e desvio padrão para cada variável. Em seguida, comparar os perfis entre três culturas escolhidas.

"""

*# %% [code]*

*# Agrupar por label e calcular médias*

perfil\_medio **=** df**.**groupby("label")**.**mean()

print("Perfil médio por cultura:")

print(perfil\_medio)

*# Você pode incluir gráficos de barras para comparar, por exemplo, os níveis de umidade, temperatura e nutrientes entre as culturas.*

perfil\_medio[['temperature', 'humidity', 'rainfall', 'P']]**.**plot(kind**=**'bar', figsize**=**(10,6))

plt**.**title("Comparação de Condições Climáticas e de Solo por Cultura")

plt**.**ylabel("Média")

plt**.**show()

Perfil médio por cultura:

N P K temperature humidity ph \

label

apple 20.80 134.22 199.89 22.630942 92.333383 5.929663

banana 100.23 82.01 50.05 27.376798 80.358123 5.983893

blackgram 40.02 67.47 19.24 29.973340 65.118426 7.133952

chickpea 40.09 67.79 79.92 18.872847 16.860439 7.336957

coconut 21.98 16.93 30.59 27.409892 94.844272 5.976562

coffee 101.20 28.74 29.94 25.540477 58.869846 6.790308

cotton 117.77 46.24 19.56 23.988958 79.843474 6.912675

grapes 23.18 132.53 200.11 23.849575 81.875228 6.025937

jute 78.40 46.86 39.99 24.958376 79.639864 6.732778

kidneybeans 20.75 67.54 20.05 20.115085 21.605357 5.749411

lentil 18.77 68.36 19.41 24.509052 64.804785 6.927932

maize 77.76 48.44 19.79 22.389204 65.092249 6.245190

mango 20.07 27.18 29.92 31.208770 50.156573 5.766373

mothbeans 21.44 48.01 20.23 28.194920 53.160418 6.831174

mungbean 20.99 47.28 19.87 28.525775 85.499975 6.723957

muskmelon 100.32 17.72 50.08 28.663066 92.342802 6.358805

orange 19.58 16.55 10.01 22.765725 92.170209 7.016957

papaya 49.88 59.05 50.04 33.723859 92.403388 6.741442

pigeonpeas 20.73 67.73 20.29 27.741762 48.061633 5.794175

pomegranate 18.87 18.75 40.21 21.837842 90.125504 6.429172

rice 79.89 47.58 39.87 23.689332 82.272822 6.425471

watermelon 99.42 17.00 50.22 25.591767 85.160375 6.495778

rainfall

label

apple 112.654779

banana 104.626980

blackgram 67.884151

chickpea 80.058977

coconut 175.686646

coffee 158.066295

cotton 80.398043

grapes 69.611829

jute 174.792798

kidneybeans 105.919778

lentil 45.680454

maize 84.766988

mango 94.704515

mothbeans 51.198487

mungbean 48.403601

muskmelon 24.689952

orange 110.474969

papaya 142.627839

pigeonpeas 149.457564

pomegranate 107.528442

rice 236.181114

watermelon 50.786219

In [31]:

*# %% [markdown]*

"""

## 4. Modelagem Preditiva

Nossa meta é desenvolver cinco modelos para prever o tipo de cultura. Primeiro, vamos separar as variáveis independentes (X) e a variável alvo (y).

"""

*# Carregar a base de dados*

df **=** pd**.**read\_csv("Atividade\_Cap\_14\_produtos\_agricolas.csv")

*# %% [code]*

*# Preparação dos dados para modelagem*

X **=** df**.**drop(columns**=**["label"])

y **=** df["label"]

*# Divisão dos dados em treino e teste*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

X, y, test\_size**=**0.2, random\_state**=**42

)

In [32]:

*# %% [markdown]*

"""

### 4.1 Modelo: Random Forest

"""

*# %% [code]*

rf\_model **=** RandomForestClassifier(n\_estimators**=**100, random\_state**=**42)

rf\_model**.**fit(X\_train, y\_train)

y\_pred\_rf **=** rf\_model**.**predict(X\_test)

accuracy\_rf **=** accuracy\_score(y\_test, y\_pred\_rf)

print(f"Acurácia Random Forest: {accuracy\_rf:.2f}")

print(classification\_report(y\_test, y\_pred\_rf))

Acurácia Random Forest: 0.99

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

In [33]:

*# %% [markdown]*

"""

### 4.2 Modelo: Support Vector Machine (SVM)

"""

*# %% [code]*

svm\_model **=** SVC(kernel**=**'linear')

svm\_model**.**fit(X\_train, y\_train)

y\_pred\_svm **=** svm\_model**.**predict(X\_test)

accuracy\_svm **=** accuracy\_score(y\_test, y\_pred\_svm)

print(f"Acurácia SVM: {accuracy\_svm:.2f}")

print(classification\_report(y\_test, y\_pred\_svm))

Acurácia SVM: 0.98

precision recall f1-score support

apple 1.00 1.00 1.00 23

banana 1.00 1.00 1.00 21

blackgram 0.95 1.00 0.98 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 0.94 1.00 0.97 17

grapes 1.00 1.00 1.00 14

jute 0.87 0.87 0.87 23

kidneybeans 1.00 1.00 1.00 20

lentil 1.00 1.00 1.00 11

maize 1.00 0.95 0.98 21

mango 0.95 1.00 0.97 19

mothbeans 1.00 1.00 1.00 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 1.00 0.98 23

pigeonpeas 1.00 0.91 0.95 23

pomegranate 1.00 1.00 1.00 23

rice 0.94 0.84 0.89 19

watermelon 1.00 1.00 1.00 19

accuracy 0.98 440

macro avg 0.98 0.98 0.98 440

weighted avg 0.98 0.98 0.98 440

In [34]:

*# %% [markdown]*

"""

### 4.3 Modelo: K-Nearest Neighbors (KNN)

"""

*# %% [code]*

knn\_model **=** KNeighborsClassifier(n\_neighbors**=**5)

knn\_model**.**fit(X\_train, y\_train)

y\_pred\_knn **=** knn\_model**.**predict(X\_test)

accuracy\_knn **=** accuracy\_score(y\_test, y\_pred\_knn)

print(f"Acurácia KNN: {accuracy\_knn:.2f}")

print(classification\_report(y\_test, y\_pred\_knn))

Acurácia KNN: 0.97

precision recall f1-score support

apple 1.00 1.00 1.00 23

banana 1.00 1.00 1.00 21

blackgram 0.95 1.00 0.98 20

chickpea 1.00 1.00 1.00 26

coconut 1.00 1.00 1.00 27

coffee 1.00 0.94 0.97 17

cotton 0.94 1.00 0.97 17

grapes 1.00 1.00 1.00 14

jute 0.76 0.96 0.85 23

kidneybeans 0.95 1.00 0.98 20

lentil 0.85 1.00 0.92 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 0.91 0.95 23

pomegranate 1.00 1.00 1.00 23

rice 0.93 0.68 0.79 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

In [35]:

*# %% [markdown]*

"""

### 4.4 Modelo: Regressão Logística

"""

*# %% [code]*

lr\_model **=** LogisticRegression(max\_iter**=**500)

lr\_model**.**fit(X\_train, y\_train)

y\_pred\_lr **=** lr\_model**.**predict(X\_test)

accuracy\_lr **=** accuracy\_score(y\_test, y\_pred\_lr)

print(f"Acurácia Regressão Logística: {accuracy\_lr:.2f}")

print(classification\_report(y\_test, y\_pred\_lr))

Acurácia Regressão Logística: 0.94

precision recall f1-score support

apple 1.00 1.00 1.00 23

banana 1.00 1.00 1.00 21

blackgram 0.75 0.75 0.75 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 0.80 0.94 0.86 17

grapes 1.00 1.00 1.00 14

jute 0.87 0.87 0.87 23

kidneybeans 1.00 1.00 1.00 20

lentil 0.71 0.91 0.80 11

maize 0.94 0.81 0.87 21

mango 1.00 1.00 1.00 19

mothbeans 0.87 0.83 0.85 24

mungbean 0.95 1.00 0.97 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 1.00 0.87 0.93 23

pomegranate 1.00 1.00 1.00 23

rice 0.89 0.84 0.86 19

watermelon 1.00 1.00 1.00 19

accuracy 0.94 440

macro avg 0.94 0.94 0.94 440

weighted avg 0.95 0.94 0.94 440

C:\Users\User\anaconda3\Lib\site-packages\sklearn\linear\_model\\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

https://scikit-learn.org/stable/modules/preprocessing.html

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

n\_iter\_i = \_check\_optimize\_result(

In [36]:

*# %% [markdown]*

"""

### 4.5 Modelo: Redes Neurais (MLP)

"""

*# %% [code]*

mlp\_model **=** MLPClassifier(hidden\_layer\_sizes**=**(50,), max\_iter**=**500, random\_state**=**42)

mlp\_model**.**fit(X\_train, y\_train)

y\_pred\_mlp **=** mlp\_model**.**predict(X\_test)

accuracy\_mlp **=** accuracy\_score(y\_test, y\_pred\_mlp)

print(f"Acurácia Redes Neurais: {accuracy\_mlp:.2f}")

print(classification\_report(y\_test, y\_pred\_mlp))

Acurácia Redes Neurais: 0.95

precision recall f1-score support

apple 1.00 1.00 1.00 23

banana 1.00 1.00 1.00 21

blackgram 0.89 0.80 0.84 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 0.80 0.94 0.86 17

grapes 1.00 1.00 1.00 14

jute 0.86 0.83 0.84 23

kidneybeans 0.95 1.00 0.98 20

lentil 0.77 0.91 0.83 11

maize 0.94 0.76 0.84 21

mango 1.00 1.00 1.00 19

mothbeans 0.88 0.96 0.92 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 0.96 0.98 23

pigeonpeas 1.00 0.87 0.93 23

pomegranate 1.00 1.00 1.00 23

rice 0.81 0.89 0.85 19

watermelon 1.00 1.00 1.00 19

accuracy 0.95 440

macro avg 0.95 0.95 0.95 440

weighted avg 0.95 0.95 0.95 440

In [37]:

*# %% [markdown]*

"""

## 5. Comparação dos Modelos e Conclusões

## Comparação de Acurácias:

Os modelos exibiram as seguintes acurácias:

• Random Forest: 99%

• SVM: 98%

• KNN: 97%

• Regressão Logística: 94%

• Redes Neurais (MLP): 95%

## Pontos Fortes e Limitações:

• Random Forest teve a melhor performance global, sendo robusto em diferentes classes. Seu desempenho é ótimo para capturar interações complexas

entre variáveis, mas pode ser mais lento e menos interpretável.

• SVM demonstrou alta precisão, especialmente em classes bem separadas. No entanto, o custo computacional pode ser elevado conforme o tamanho

do conjunto de dados cresce.

• KNN teve uma acurácia relativamente alta, mas apresenta desafios em termos de eficiência para conjuntos grandes, devido à necessidade de

calcular distâncias para cada nova predição.

• Regressão Logística foi o modelo com menor acurácia, possivelmente devido à linearidade dos dados. Além disso, o aviso de ConvergenceWarning

sugere que ajustes nos hiperparâmetros ou pré-processamento podem ser necessários.

• Redes Neurais (MLP) mostraram desempenho robusto, ficando acima da regressão logística, mas abaixo dos modelos de árvore. Dependem de um

bom ajuste de hiperparâmetros para evitar problemas como overfitting.

## Variáveis de Maior Impacto:

Embora os dados exatos das variáveis mais influentes não estejam explicitamente detalhados, modelos como Random Forest permitem análise de

importância das variáveis. Em geral, fatores como clima, tipo de solo e características nutricionais tendem a ter maior peso na predição de cultivos.

## Perfil Ideal e Relação com Diferentes Culturas

O perfil ideal identificado tende a favorecer cultivos específicos conforme suas necessidades ambientais. Algumas observações:

• Culturas como banana, coco e café mostram excelente classificação, sugerindo alta previsibilidade baseada nos fatores ambientais.

• Juta, lentilha e arroz apresentaram pequenas oscilações na precisão de alguns modelos, o que pode indicar maior sensibilidade a variações nos dados.

• O impacto dos fatores ambientais na previsão pode estar relacionado à regionalização da agricultura. Por exemplo, um modelo que capta padrões

climáticos distintos pode ser mais eficiente na predição de cultivos em diferentes regiões.

## Visualizações e Comentários Adicionais:

Uma análise gráfica pode revelar mais sobre o comportamento dos modelos

• Gráfico de importância das variáveis no Random Forest.

"""

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

*# Obter as importâncias das variáveis*

importances **=** rf\_model**.**feature\_importances\_

*# Criar um DataFrame para facilitar a visualização*

feature\_names **=** X\_train**.**columns *# Nome das variáveis*

importance\_df **=** pd**.**DataFrame({'Feature': feature\_names, 'Importance': importances})

*# Ordenar as variáveis pela importância*

importance\_df **=** importance\_df**.**sort\_values(by**=**'Importance', ascending**=False**)

*# Plotar o gráfico de barras*

plt**.**figure(figsize**=**(10, 6))

sns**.**barplot(x**=**'Importance', y**=**'Feature', data**=**importance\_df, palette**=**'viridis')

plt**.**xlabel('Importância')

plt**.**ylabel('Variáveis')

plt**.**title('Importância das Variáveis - Random Forest')

plt**.**show()

C:\Users\User\AppData\Local\Temp\ipykernel\_1348\2359822598.py:59: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x='Importance', y='Feature', data=importance\_df, palette='viridis')

In [38]:

*# %% [markdown]*

"""

# Conclusão

Neste notebook, exploramos a base de dados, realizamos análises descritivas e implementamos cinco algoritmos de Machine Learning para prever o tipo de cultura agrícola.

A partir dos resultados obtidos, podemos identificar quais condições de solo e clima favorecem cada cultura e quais métodos preditivos têm maior performance.

\*Sinta-se à vontade para ajustar os hiperparâmetros, realizar mais pré-processamentos ou adicionar outras visualizações conforme necessário.\*

"""

Out[38]:

'\n# Conclusão\n\nNeste notebook, exploramos a base de dados, realizamos análises descritivas e implementamos cinco algoritmos de Machine Learning para prever o tipo de cultura agrícola. \nA partir dos resultados obtidos, podemos identificar quais condições de solo e clima favorecem cada cultura e quais métodos preditivos têm maior performance.\n\n\*Sinta-se à vontade para ajustar os hiperparâmetros, realizar mais pré-processamentos ou adicionar outras visualizações conforme necessário.\* \n'

In [ ]:

In [ ]: