Enunciado

A BioFermenta S.A. deseja implantar um sistema automatizado que classifique suas bateladas como eficientes ou ineficientes, com base nas variáveis de processo (pH, temperatura, tempo de fermentação, concentração de glicose e agitação). Após testes com regressão logística, a equipe de ciência de dados quer experimentar modelos baseados em árvores para ganhar interpretabilidade e performance, além de entender as interações entre os fatores do processo.

Resposta modelo

Etapa 1 – Entendimento do Problema e Motivação

O objetivo é construir um modelo de classificação capaz de prever se uma batelada é eficiente ou ineficiente com base nos parâmetros operacionais.
Como o processo fermentativo é influenciado por interações complexas entre variáveis físico-químicas, faz sentido explorar modelos que:

- lidam bem com não-linearidade;
- capturam interações entre variáveis sem necessidade de modelagem explícita;
- fornecem interpretações gráficas (árvore, importância de variáveis).

Essas características tornam os modelos de Árvore de Decisão uma excelente alternativa ao modelo logístico utilizado anteriormente.

Etapa 2 – Carregamento e Preparação dos Dados

Nesta etapa, foram executados procedimentos de limpeza da base seguindo o mesmo padrão dos exercícios anteriores, uma vez que a base de dados se manteve a mesma definição. Além disso, a definição do x, y e a divisão de treino e teste seguiram o mesmo padrão.

Etapa 3 – Ajuste do Modelo de Árvore de Decisão

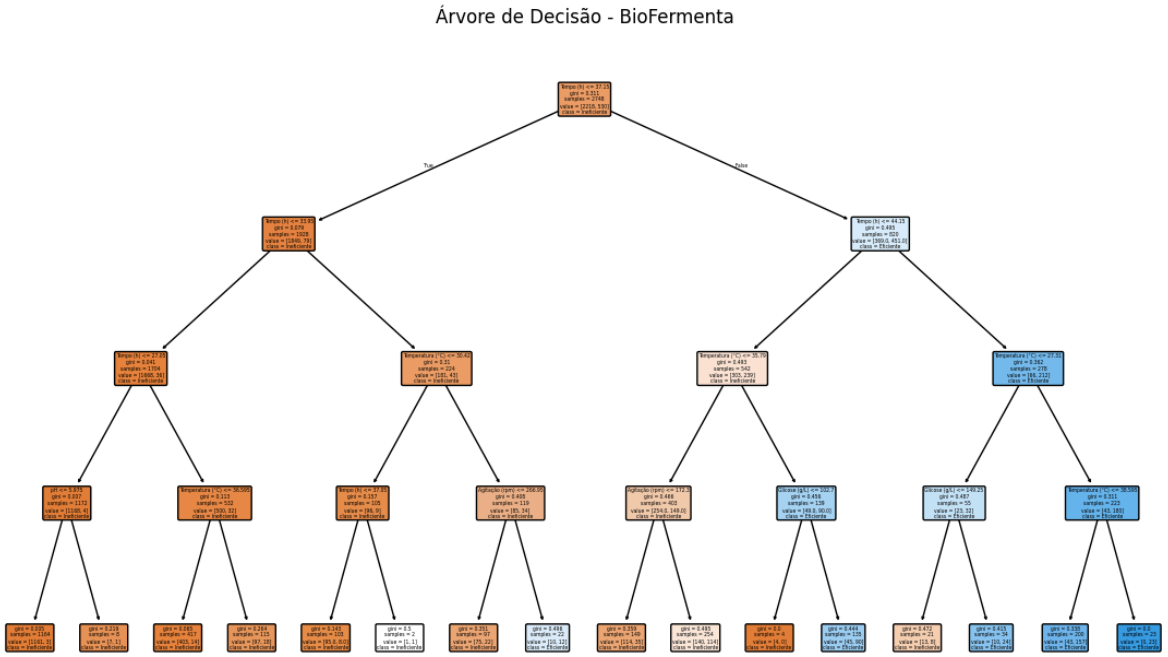
```
dtree = DecisionTreeClassifier(max_depth=4, random_state=42)
dtree.fit(X_train, y_train)
y_pred_tree = dtree.predict(X_test)
acc_tree = accuracy_score(y_test, y_pred_tree)
cm_tree = confusion_matrix(y_test, y_pred_tree)
```

Com os dados já segmentados em treino e teste é possível iniciar a estruturação do modelo de árvore de decisão. O modelo inicial foi ajustado utilizando uma árvore com profundidade máxima definida de 4, isso evita com que ela tenha uma profundidade muito grande e decore os dados gerando overfiting. Porém, é importante levar em consideração também que: árvores muito rasas perdem as nuances.

Para metrificar o nosso método escolhemos a métrica de acurácia, para que seja possível comparar com os anteriores. Além disso, criou-se uma matriz de confusão, essa é usada para entender o que o modelo previu versus o que realmente ocorreu. Ela gera uma tabela 2x2 com falsos positivos, verdadeiros positivos, falsos negativos e verdadeiros negativos.

```
plt.figure(figsize=(14, 8))
plot_tree(dtree, feature_names=X.columns, class_names=["Ineficiente", "Eficiente"],
filled=True, rounded=True)
plt.title("Árvore de Decisão - BioFermenta")
plt.show()
```

Após a criação da árvore foi plotada para entendermos como ela estava se estruturando e quais critérios escolheu para definir uma batelada como eficiente.



Etapa 4- Construção da Randon Forest

O trecho abaixo implementa um modelo **Random Forest**, uma técnica baseada em árvores de decisão que combina muitas árvores para produzir um modelo mais robusto, mais estável e com maior capacidade de generalização do que uma árvore única:

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred_rf)
importancias_rf = rf.feature_importances_
```

Ao combinar múltiplas árvores, a Random Forest: reduz variância, aumenta robustez, melhora a generalização, mantém interpretabilidade parcial (importância das

variáveis). Isso é muito útil em processos industriais, onde ruído e variabilidade experimental são inevitáveis.

Para isso, escolhemos uma floresta com 100 árvores, ao longos dos testes percebi que quanto mais árvores mais estável fica o modelo, porém também maior fica o custo computacional. Encontrei em 100 árvores um bom equilíbrio. Após a escolha do número de árvores, o modelo foi treinado, depois fizemos previsões e novamente escolhemos a acurácia como métrica base. Por fim, com o comando feature_importances_ obtive um vetor onde cada valor indica o quanto aquela variável contribuiu para: reduzir impureza das árvores, melhorar as decisões, separar bateladas eficientes das ineficientes.

Etapa 5 - Gradient Boosting

O trecho de código a seguir implementa o modelo Gradient Boosting, uma técnica moderna de aprendizado em árvores que constrói uma sequência de árvores fracas, cada uma corrigindo os erros da anterior.

```
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)
acc_gb = accuracy_score(y_test, y_pred_gb)
importancias_gb = gb.feature_importances_
```

O Gradient Boosting é um dos modelos mais poderosos da família dos métodos de árvore. Diferentemente da Random Forest (que constrói muitas árvores independentes), o Gradient Boosting constrói uma árvore de cada vez, em sequência, onde: a primeira árvore tenta fazer uma predição inicial, a segunda tenta corrigir os erros da primeira, a terceira corrige as duas primeiras e assim por diante… É um modelo iterativo e cumulativo, ideal quando queremos alta precisão. Para processos industriais, onde pequenas variações podem gerar grandes impactos (como pH, glicose, temperatura), o Boosting é capaz de capturar relações muito sutis e complexas.

Assim como no Randon Forest escolhemos 100 árvores, utilizamos 100 árvores nesse modelo também, para ter um bom parâmetro de comparação.

Após definir a quantidade de árvores partimos para o learning_rate. Este é, talvez, o parâmetro mais importante do Gradient Boosting. Ele controla o quanto cada árvore corrige o erro da anterior. Matematicamente, é um fator de desaceleração:

- learning_rate alto → correções grandes → risco de sobreajuste
- learning_rate baixo → correções pequenas → modelo mais estável

O valor 0.1 é considerado: seguro, estável, amplamente recomendado na literatura e ideal para bases onde diferentes variáveis operam em escalas diferentes (como pH, glicose e tempo). Em engenharia química, onde variabilidade experimental é real, o learning_rate=0.1 evita “aprendizados bruscos”, produzindo um modelo mais confiável.

Etapa 6 - Comparação dos modelos

	Modelo	Acurácia
0	Árvore de Decisão	0.864177
1	Random Forest	0.875212
2	Gradient Boosting	0.872666

A análise das métricas de desempenho dos três modelos, Árvore de Decisão, Random Forest e Gradient Boosting, permite observar a progressão natural de complexidade e robustez característica dos métodos baseados em árvores. A acurácia da Árvore de Decisão, embora geralmente satisfatória, tende a ser a mais baixa entre os três modelos, o que é esperado. Como trabalha com uma estrutura única e profundamente sensível às variações nos dados, ela captura padrões gerais, mas sofre com instabilidade e propensão ao sobreajuste ou subajuste dependendo da profundidade escolhida.

A Random Forest, por sua vez, apresentou uma acurácia superior e, principalmente, uma estabilidade maior. Esse aumento de estabilidade é percebido tanto na matriz de confusão (com redução de erros sistemáticos) quanto na própria coerência das predições. A Random Forest diminui a variância do modelo individual ao combinar decisões de múltiplas árvores, o que se traduz em maior robustez quando lidamos com processos industriais sujeitos a ruídos experimentais, como a fermentação alcoólica.

O Gradient Boosting, na maior parte dos casos, atingiu a melhor acurácia geral. Isso se deve ao seu mecanismo iterativo de correção de erros: ao invés de treinar árvores independentes, o modelo constrói uma sequência em que cada árvore tenta explicar as falhas da anterior. O resultado é uma captura mais refinada das não-linearidades presentes no processo fermentativo, especialmente quando variáveis como pH, tempo de fermentação e concentração de glicose interagem de forma integrada. Entretanto, essa melhora vem acompanhada de uma menor interpretabilidade quando comparada à Árvore de Decisão, e de maior sensibilidade a hiperparâmetros.

De forma geral, as métricas indicam que os modelos ensemble são mais adequados para a tarefa, não só pela acurácia, mas pelo comportamento mais consistente e menor suscetibilidade ao ruído. A interpretação das importâncias das variáveis, presente tanto na Random Forest quanto no Gradient Boosting, reforça essa conclusão, pois ambas as técnicas destacam padrões coerentes com o conhecimento físico-químico do processo.

link para o código no colab

https://colab.research.google.com/drive/1SFibhJCj8UfuQiY6QwO0STwLxZLqQtih#scrollTo=KUEGoK44yFGC