

Projeto da Disciplina

Algoritmos de Inteligência Artificial para classificação - 25E1_2

Link do GitHub e README: https://github.com/ianmsouza/wine_quality_analysis



Importação das bibliotecas necessárias para execução desse notebook

```
In [49]: import os
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.impute import SimpleImputer
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
```

Questão 1)

Faça o módulo do [Kaggle Intro to Machine Learning](#):

Comprove a finalização do módulo com um print que contenha data e identificação do aluno.

Trabalho com base:

Iremos usar a base de dados de vinhos verdes portugueses (nas variantes branco e tinto) que encontra-se disponível no [Kaggle](#).

Resposta: A seguir os print screen com a finalização do módulo "Intro to Machine Learning" do Kaggle:

A screenshot of the Kaggle platform interface. On the left, there's a sidebar with navigation links like 'Create', 'Home', 'Competitions', 'Datasets', 'Models', 'Code', 'Discussions', 'Learn', 'More', and 'Your Work'. Under 'Learn', the 'Intro to Machine Learning' course is listed as completed. The main area shows the course details for 'Intro to Machine Learning', including a progress bar at 100% complete with a 'Congrats!' message. Below this, there are five lessons: 'How Models Work', 'Basic Data Exploration', 'Your First Machine Learning Model', 'Model Validation', and 'Underfitting and Overfitting', each with a 'Tutorial' and 'Exercise' section. To the right, there's a user profile for 'Ian Miranda Souza' showing badges received: 'Learner badge', 'Python Coder badge', and 'Vampire badge'. There are also sections for 'Your notifications' and 'Instructor' (Dan Becker). The bottom of the screen shows the Windows taskbar with various application icons.

kaggle

Create

Home Competitions Datasets Models Code Discussions

Learn

Your Work

VIEWED

- Intro to Machine Learning
- Machine Learning C...
- Random Forests
- Underfitting and Ov...
- Model Validation

EDITED

Exercise: Machine L... View Active Events

Search

Intro to Machine Learning

Learn the core ideas in machine learning, and build your first models.

View Certificate 100% complete! Congrats!

Courses Discussions

Lessons

		Tutorial	Exercise	Builds on
1	How Models Work	<input checked="" type="checkbox"/>		Python
2	Basic Data Exploration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Preparation for Machine Learning Explainability Intermediate Machine Learning Intro to Deep Learning
3	Your First Machine Learning Model	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	Model Validation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	Underfitting and Overfitting	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Hours to earn certificate 3 (estimated)

Cost No cost, like all Kaggle Learn Courses

Instructor Dan Becker



25°C Pred. nublado

21:22 POR PTB2 29/03/2025

Courses Discussions

Lessons

		Tutorial	Exercise
1	How Models Work	<input checked="" type="checkbox"/>	
2	Basic Data Exploration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	Your First Machine Learning Model	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	Model Validation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	Underfitting and Overfitting	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	Random Forests	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	Machine Learning Competitions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



Para as questões 2-5 usaremos apenas os vinhos do tipo "branco".

Questão 2)

Faça o download da base - esta é uma base real, apresentada no artigo:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Ela possui uma variável denominada "quality", uma nota de 0 a 10 que denota a qualidade do vinho. Crie uma nova variável, chamada "opinion" que será uma variável categórica igual à 0, quando quality for menor e igual à 5. O valor será 1, caso contrário. Desconsidere a variável quality para o restante da análise.

Resposta:

- Importação da base de dados oriundos do Kaggle

```
In [50]: # Verifica se está rodando no Google Colab
try:
    from google.colab import drive
    IN_COLAB = True
except ImportError:
    IN_COLAB = False

# Define o caminho do dataset
if IN_COLAB:
    drive.mount('/content/drive')
    file_path = '/content/drive/MyDrive/winequalityN.csv'
else:
    file_path = r'C:/Users/Ian/PythonProjects/infnet-25E1_2/datasets/winequalityN.csv'

# Carregando os dados
if os.path.exists(file_path):
    df = pd.read_csv(file_path)
    print("Dataset carregado com sucesso!")
else:
    print(f"Erro: O arquivo '{file_path}' não foi encontrado.")

# Exibir as primeiras Linhas para verificar se o carregamento foi bem-sucedido
print(df.head() if 'df' in locals() else "Nenhum dado foi carregado.")
```

```
Dataset carregado com sucesso!
```

```
    type fixed acidity volatile acidity citric acid residual sugar \
0  white           7.0            0.27       0.36        20.7
1  white           6.3            0.30       0.34        1.6
2  white           8.1            0.28       0.40        6.9
3  white           7.2            0.23       0.32        8.5
4  white           7.2            0.23       0.32        8.5

  chlorides free sulfur dioxide total sulfur dioxide density   pH \
0      0.045          45.0         170.0   1.0010  3.00
1      0.049          14.0         132.0   0.9940  3.30
2      0.050          30.0          97.0   0.9951  3.26
3      0.058          47.0         186.0   0.9956  3.19
4      0.058          47.0         186.0   0.9956  3.19

sulphates alcohol quality
0      0.45        8.8       6
1      0.49        9.5       6
2      0.44       10.1       6
3      0.40        9.9       6
4      0.40        9.9       6
```

```
In [51]: df.info()
```

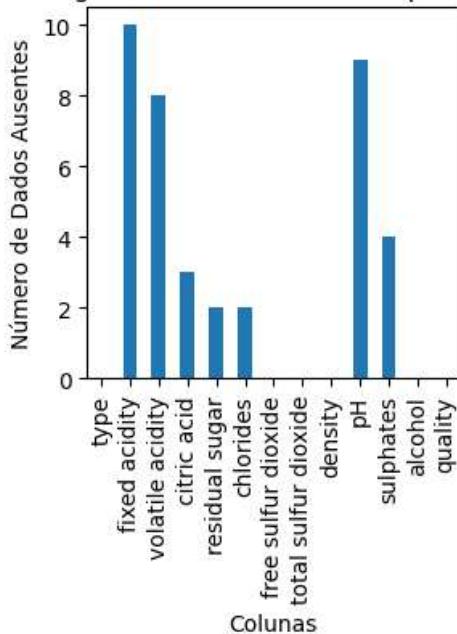
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   type             6497 non-null    object 
 1   fixed acidity    6487 non-null    float64
 2   volatile acidity 6489 non-null    float64
 3   citric acid     6494 non-null    float64
 4   residual sugar   6495 non-null    float64
 5   chlorides        6495 non-null    float64
 6   free sulfur dioxide 6497 non-null    float64
 7   total sulfur dioxide 6497 non-null    float64
 8   density          6497 non-null    float64
 9   pH               6488 non-null    float64
 10  sulphates        6493 non-null    float64
 11  alcohol          6497 non-null    float64
 12  quality          6497 non-null    int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

```
In [52]: df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
fixed acidity	6487.0	7.216579	1.296750	3.80000	6.40000	7.00000	7.70000	15.90000
volatile acidity	6489.0	0.339691	0.164649	0.08000	0.23000	0.29000	0.40000	1.58000
citric acid	6494.0	0.318722	0.145265	0.00000	0.25000	0.31000	0.39000	1.66000
residual sugar	6495.0	5.444326	4.758125	0.60000	1.80000	3.00000	8.10000	65.80000
chlorides	6495.0	0.056042	0.035036	0.00900	0.03800	0.04700	0.06500	0.61100
free sulfur dioxide	6497.0	30.525319	17.749400	1.00000	17.00000	29.00000	41.00000	289.00000
total sulfur dioxide	6497.0	115.744574	56.521855	6.00000	77.00000	118.00000	156.00000	440.00000
density	6497.0	0.994697	0.002999	0.98711	0.99234	0.99489	0.99699	1.03898
pH	6488.0	3.218395	0.160748	2.72000	3.11000	3.21000	3.32000	4.01000
sulphates	6493.0	0.531215	0.148814	0.22000	0.43000	0.51000	0.60000	2.00000
alcohol	6497.0	10.491801	1.192712	8.00000	9.50000	10.30000	11.30000	14.90000
quality	6497.0	5.818378	0.873255	3.00000	5.00000	6.00000	6.00000	9.00000

```
In [53]: missing_data = df.isnull().sum()
missing_data.plot(kind='bar', figsize=(3,3))
plt.title("Contagem de Dados Ausentes por Coluna")
plt.xlabel("Colunas")
plt.ylabel("Número de Dados Ausentes")
plt.show();
```

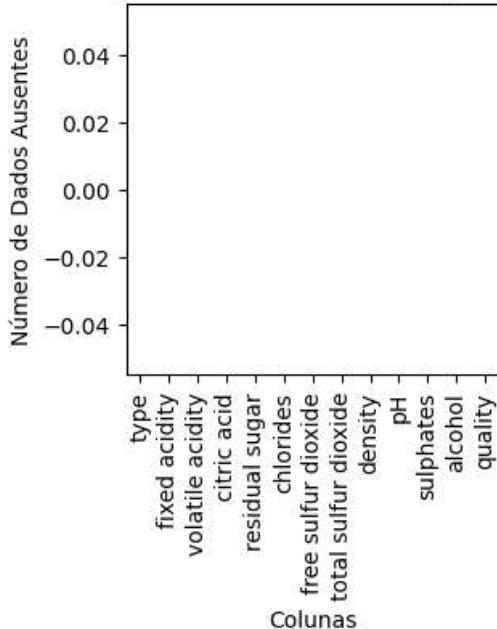
Contagem de Dados Ausentes por Coluna



```
In [54]: df.ffill(inplace=True) # Usando o método de forward fill para tratar valores ausentes
```

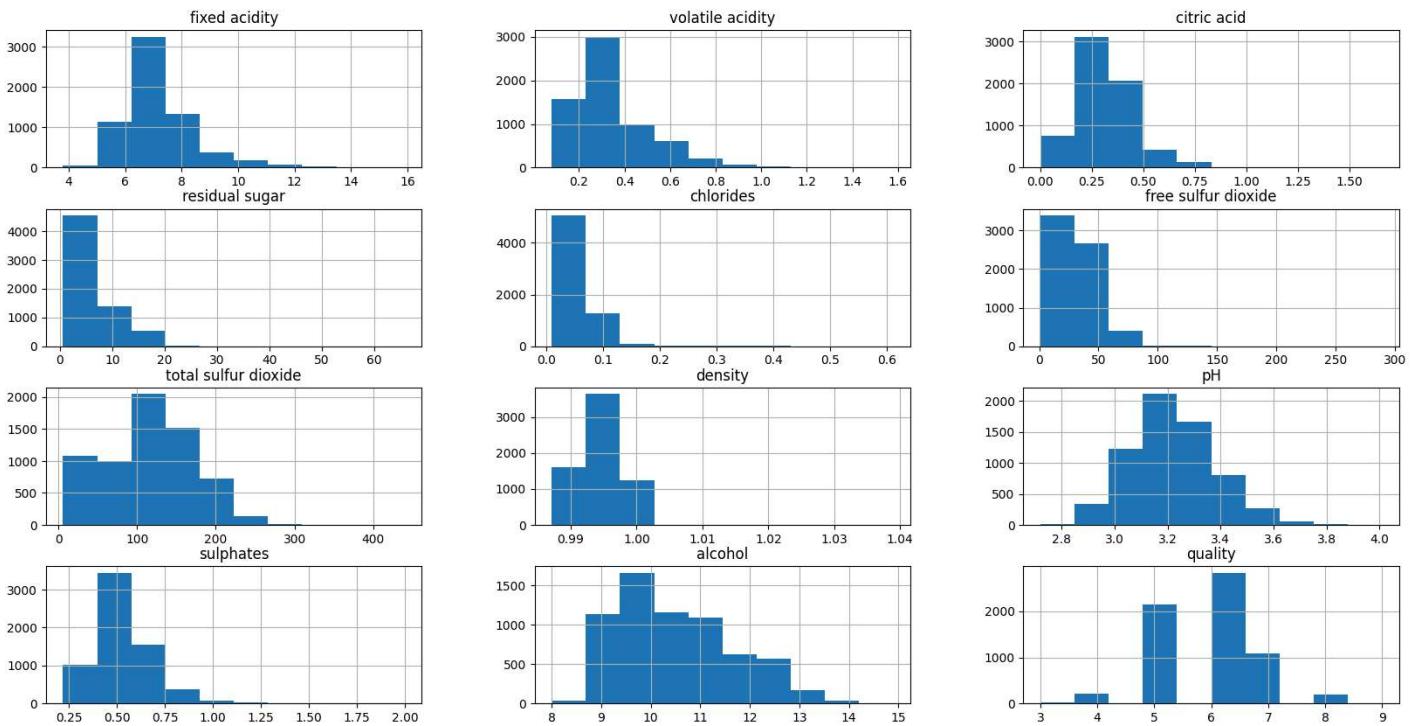
```
In [55]: missing_data = df.isnull().sum()
missing_data.plot(kind='bar', figsize=(3,3))
plt.title("Contagem de Dados Ausentes por Coluna")
plt.xlabel("Colunas")
plt.ylabel("Número de Dados Ausentes")
plt.show();
```

Contagem de Dados Ausentes por Coluna



```
In [56]: df.hist(figsize=(20, 10))
```

```
Out[56]: array([[<Axes: title={'center': 'fixed acidity'}>,
   <Axes: title={'center': 'volatile acidity'}>,
   <Axes: title={'center': 'citric acid'}>],
  [<Axes: title={'center': 'residual sugar'}>,
   <Axes: title={'center': 'chlorides'}>,
   <Axes: title={'center': 'free sulfur dioxide'}>],
  [<Axes: title={'center': 'total sulfur dioxide'}>,
   <Axes: title={'center': 'density'}>,
   <Axes: title={'center': 'pH'}>],
  [<Axes: title={'center': 'sulphates'}>,
   <Axes: title={'center': 'alcohol'}>,
   <Axes: title={'center': 'quality'}>]], dtype=object)
```



- Criando a variável categórica 'opinion'
- Removendo a coluna original 'quality'
- Para as questões 2-5 usaremos apenas os vinhos do tipo "branco"
- Filtrando apenas os vinhos brancos

```
In [57]: # Filtrando apenas os vinhos brancos
df_white = df[df['type'] == 'white'].copy()

# Criando a variável categórica 'opinion'
df_white['opinion'] = df_white['quality'].apply(lambda x: 0 if x <= 5 else 1)

# Removendo a coluna original 'quality'
df_white.drop(columns=['quality'], inplace=True)

# Exibir as primeiras Linhas do dataset filtrado
print(df_white.head())
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	7.0	0.27	0.36	20.7	
1	white	6.3	0.30	0.34	1.6	
2	white	8.1	0.28	0.40	6.9	
3	white	7.2	0.23	0.32	8.5	
4	white	7.2	0.23	0.32	8.5	

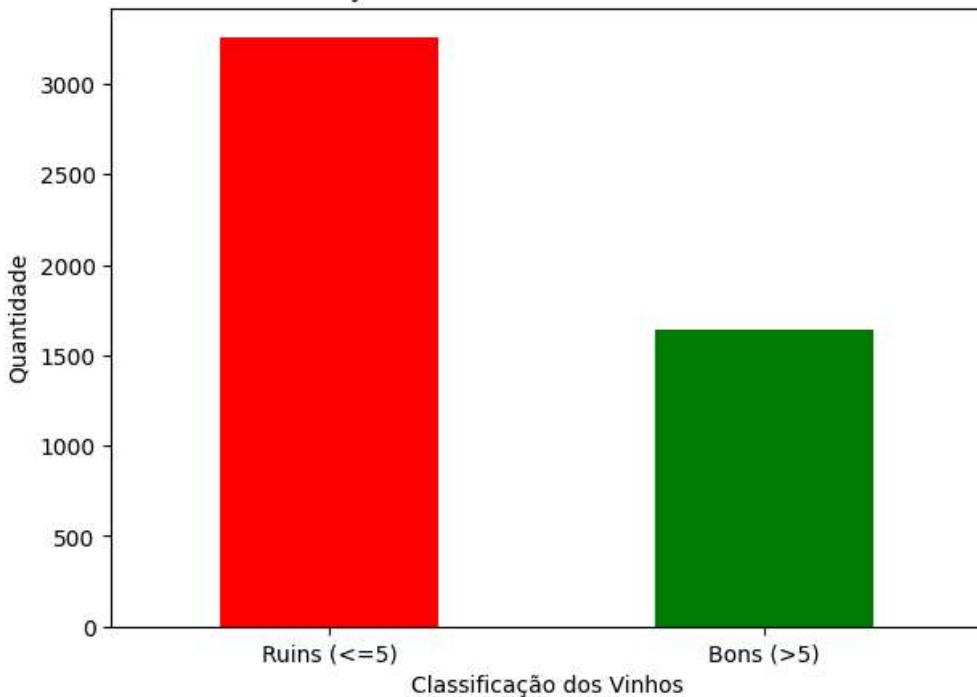
	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	\
0	0.045	45.0	170.0	1.0010	3.00	
1	0.049	14.0	132.0	0.9940	3.30	
2	0.050	30.0	97.0	0.9951	3.26	
3	0.058	47.0	186.0	0.9956	3.19	
4	0.058	47.0	186.0	0.9956	3.19	

	sulphates	alcohol	opinion	
0	0.45	8.8	1	
1	0.49	9.5	1	
2	0.44	10.1	1	
3	0.40	9.9	1	
4	0.40	9.9	1	

Criando gráfico de distribuição da variável 'opinion'

```
In [58]: plt.figure(figsize=(7, 5))
df_white['opinion'].value_counts().plot(kind='bar', color=['red', 'green'])
plt.xlabel('Classificação dos Vinhos')
plt.ylabel('Quantidade')
plt.title('Distribuição de Vinhos Brancos - Bons vs Ruins')
plt.xticks(ticks=[0, 1], labels=['Ruins (<5)', 'Bons (>5)'], rotation=0)
plt.show()
```

Distribuição de Vinhos Brancos - Bons vs Ruins



Questão 3)

Descreva as variáveis presentes na base.

Quais são as variáveis?

Quais são os tipos de variáveis (discreta, categórica, contínua)?

Quais são as médias e desvios padrões?

Resposta:

O conjunto de dados contém as seguintes variáveis

#	Variável	Descrição
1	fixed_acidity	Acidez fixa do vinho.
2	volatile_acidity	Acidez volátil do vinho.
3	citric_acid	Quantidade de ácido cítrico presente no vinho.
4	residual_sugar	Quantidade de açúcar residual no vinho.
5	chlorides	Concentração de cloreto no vinho.
6	free_sulfur_dioxide	Quantidade de dióxido de enxofre livre no vinho.
7	total_sulfur_dioxide	Quantidade total de dióxido de enxofre no vinho.
8	density	Densidade do vinho.
9	pH	pH do vinho.
10	sulphates	Concentração de sulfatos no vinho.
11	alcohol	Teor alcoólico do vinho.
11	quality	Pontuação entre 0 e 10.
12	type	Tipo de vinho (tinto ou branco).
13	opinion	Variável categórica criada anteriormente, onde 0 indica vinho de qualidade menor ou igual a 5, e 1 indica vinho de qualidade superior a 5.

Tipos de Variáveis

Variáveis Contínuas

- fixed_acidity
- volatile_acidity

- citric_acid
- residual_sugar
- chlorides
- free_sulfur_dioxide
- total_sulfur_dioxide
- density
- pH
- sulphates
- alcohol

Variáveis Categóricas

- type
- opinion

Calculando as médias e desvios padrões

```
In [59]: # Calcular médias e desvios padrão
statistics = df_white.describe().loc[['mean', 'std']]
print(statistics)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
mean	6.855033	0.278312	0.334204	6.391343	0.045773	
std	0.843450	0.100841	0.121020	5.072120	0.021847	
	free sulfur dioxide	total sulfur dioxide	density	pH	\	
mean	35.308085	138.360657	0.994027	3.188269		
std	17.007137	42.498065	0.002991	0.150937		
	sulphates	alcohol	opinion			
mean	0.489835	10.514267	0.665169			
std	0.114141	1.230621	0.471979			

Questão 4)

Com a base escolhida:

a) Descreva as etapas necessárias para criar um modelo de classificação eficiente.

Resposta:

Etapa	Descrição
1. Definição do Problema	Entenda claramente o problema que deseja resolver. Defina a variável-alvo e os recursos de entrada.
2. Coleta de Dados	Reúna dados relevantes para o problema. Isso pode envolver a coleta de dados de várias fontes, incluindo bancos de dados, APIs, ou pesquisa de campo.
3. Limpeza e Preparação dos Dados	Lide com valores ausentes, remova outliers e normalize ou padronize os dados, se necessário. Transforme variáveis categóricas em numéricas usando técnicas como one-hot encoding.
4. Divisão de Dados	Divida os dados em conjuntos de treinamento e teste. Uma divisão comum é 80% para treinamento e 20% para teste.
5. Seleção de Modelo	Escolha um algoritmo de classificação adequado para o problema. Exemplos incluem: Regressão logística, Árvores de decisão, Random Forest, Support Vector Machine (SVM), Redes Neurais.
6. Treinamento do Modelo	Treine o modelo nos dados de treinamento. Ajuste os hiperparâmetros, se necessário, para melhorar o desempenho.
7. Avaliação do Modelo	Avalie o modelo usando os dados de teste. Métricas comuns incluem precisão, recall, f1-score, e AUC-ROC. Faça validação cruzada para garantir que o modelo não está superajustado.
8. Otimização do Modelo	Ajuste os hiperparâmetros e tente diferentes algoritmos para encontrar a melhor combinação que otimiza o desempenho do modelo.
9. Implementação	Implante o modelo em um ambiente de produção. Isso pode envolver a criação de uma API para o modelo ou a integração com um sistema existente.
10. Monitoramento e Manutenção	Monitore o desempenho do modelo em produção e atualize o modelo conforme novos dados se tornam disponíveis. Realize manutenção preventiva para garantir que o modelo continue a funcionar de maneira eficiente.

b) Treine um modelo de regressão logística usando um modelo de validação cruzada estratificada com k-folds ($k=10$) para realizar a classificação. Calcule para a base de teste:
... i) a média e desvio da acurácia dos modelos obtidos;

- ii) a média e desvio da precisão dos modelos obtidos;
- iii) a média e desvio da recall dos modelos obtidos;
- iv) a média e desvio do f1-score dos modelos obtidos.

c) Treine um modelo de árvores de decisão usando um modelo de validação cruzada estratificada com k-folds ($k=10$) para realizar a classificação. Calcule para a base de teste:

- i. a média e desvio da acurácia dos modelos obtidos;
- ii. a média e desvio da precisão dos modelos obtidos;
- iii. a média e desvio da recall dos modelos obtidos;
- iv. a média e desvio do f1-score dos modelos obtidos.

d) Treine um modelo de SVM usando um modelo de validação cruzada estratificada com k-folds ($k=10$) para realizar a classificação. Calcule para a base de teste:

- i. a média e desvio da acurácia dos modelos obtidos;
- ii. a média e desvio da precisão dos modelos obtidos;
- iii. a média e desvio da recall dos modelos obtidos;
- iv. a média e desvio do f1-score dos modelos obtidos.

Resposta:

```
In [60]: # Remover a coluna 'type' (não é necessária)
df_white.drop(columns=['type'], inplace=True)

# Separar variáveis independentes (X) e dependente (y)
X = df_white.drop(columns=['opinion'])
y = df_white['opinion']

# Tratar valores ausentes (preenchendo com a média de cada coluna)
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

# Normalizar os dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# Definir validação cruzada estratificada com k=10
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Lista de modelos
models = {
    'Logistic Regression': LogisticRegression(random_state=42),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'SVM': SVC(kernel='linear', probability=True, random_state=42)
}

# Métricas aceitas pelo cross_val_score
metrics = ['accuracy', 'precision', 'recall', 'f1']

# Dicionário para armazenar os resultados
results = {}

# Treinamento e avaliação dos modelos
for name, model in models.items():
    print(f"\nTreinando modelo: {name}")

    model_results = {}

    for metric in metrics:
        scores = cross_val_score(model, X_scaled, y, cv=kf, scoring=metric)
        mean_score = np.mean(scores)
        std_score = np.std(scores)

        model_results[metric] = {'mean': mean_score, 'std': std_score}

    print(f'{name}: Média = {mean_score:.4f} ({mean_score * 100:.2f}%), '
          f'Desvio Padrão = {std_score:.4f} ({std_score * 100:.2f}%)')

    results[name] = model_results
```

Treinando modelo: Logistic Regression
accuracy: Média = 0.7487 (74.87%), Desvio Padrão = 0.0160 (1.60%)
precision: Média = 0.7745 (77.45%), Desvio Padrão = 0.0153 (1.53%)
recall: Média = 0.8784 (87.84%), Desvio Padrão = 0.0182 (1.82%)
f1: Média = 0.8230 (82.30%), Desvio Padrão = 0.0108 (1.08%)

Treinando modelo: Decision Tree
accuracy: Média = 0.7958 (79.58%), Desvio Padrão = 0.0161 (1.61%)
precision: Média = 0.8463 (84.63%), Desvio Padrão = 0.0113 (1.13%)
recall: Média = 0.8471 (84.71%), Desvio Padrão = 0.0250 (2.50%)
f1: Média = 0.8465 (84.65%), Desvio Padrão = 0.0137 (1.37%)

Treinando modelo: SVM
accuracy: Média = 0.7523 (75.23%), Desvio Padrão = 0.0139 (1.39%)
precision: Média = 0.7727 (77.27%), Desvio Padrão = 0.0116 (1.16%)
recall: Média = 0.8898 (88.98%), Desvio Padrão = 0.0212 (2.12%)
f1: Média = 0.8269 (82.69%), Desvio Padrão = 0.0104 (1.04%)

Modelo	Métrica	Média	Média (%)	Desvio Padrão	Desvio Padrão (%)
Logistic Regression (Regressão Logística)	accuracy	0.7487	74.87%	0.0160	1.60%
	precision	0.7745	77.45%	0.0153	1.53%
	recall	0.8784	87.84%	0.0182	1.82%
	f1_score	0.8230	82.30%	0.0108	1.08%
Decision Tree (Árvore de decisão)	accuracy	0.7958	79.58%	0.0161	1.61%
	precision	0.8463	84.63%	0.0113	1.13%
	recall	0.8471	84.71%	0.0250	2.50%
	f1_score	0.8465	84.65%	0.0137	1.37%
SVM (Support Vector Machine)	accuracy	0.7523	75.23%	0.0139	1.39%
	precision	0.7727	77.27%	0.0116	1.16%
	recall	0.8898	88.98%	0.0212	2.12%
	f1_score	0.8269	82.69%	0.0104	1.04%

```
In [61]: # Exibir resultados como DataFrame
df_results = pd.DataFrame.from_dict(
    {(i, j): results[i][j] for i in results.keys() for j in results[i].keys()},
    orient='index'
)

# Criar colunas de percentual para exibição
df_results['mean (%)'] = df_results['mean'] * 100
df_results['std (%)'] = df_results['std'] * 100

# Exibir DataFrame corrigido
print("\nResultados dos Modelos:")
print(df_results)
```

Resultados dos Modelos:

		mean	std	mean (%)	std (%)
Logistic Regression	accuracy	0.748670	0.016033	74.866950	1.603295
	precision	0.774496	0.015313	77.449552	1.531323
	recall	0.878446	0.018205	87.844644	1.820514
	f1	0.823006	0.010792	82.300585	1.079223
Decision Tree	accuracy	0.795837	0.016124	79.583699	1.612420
	precision	0.846270	0.011344	84.626977	1.134371
	recall	0.847141	0.024978	84.714110	2.497809
	f1	0.846479	0.013714	84.647859	1.371418
SVM	accuracy	0.752344	0.013854	75.234423	1.385447
	precision	0.772670	0.011628	77.266989	1.162798
	recall	0.889801	0.021235	88.980085	2.123490
	f1	0.826918	0.010361	82.691811	1.036072

Matrizes de confusão para cada modelo

```
In [65]: for name, model in models.items():
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

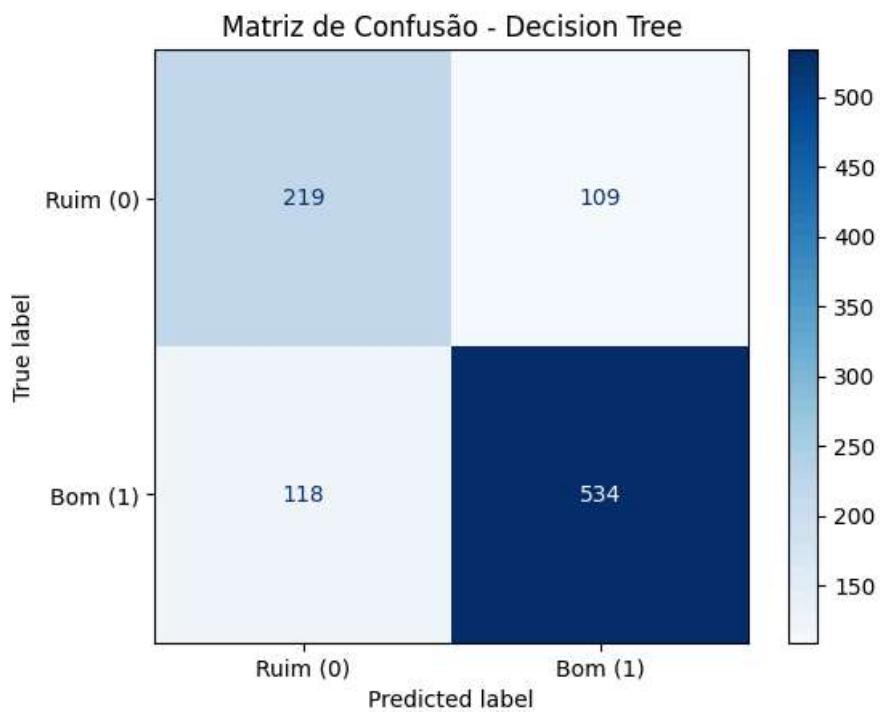
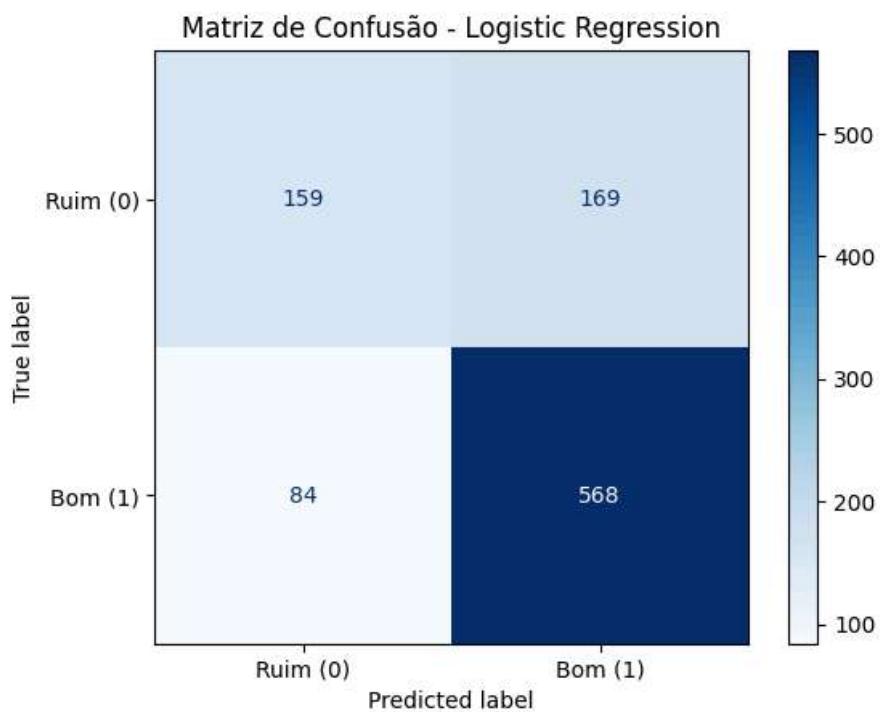
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

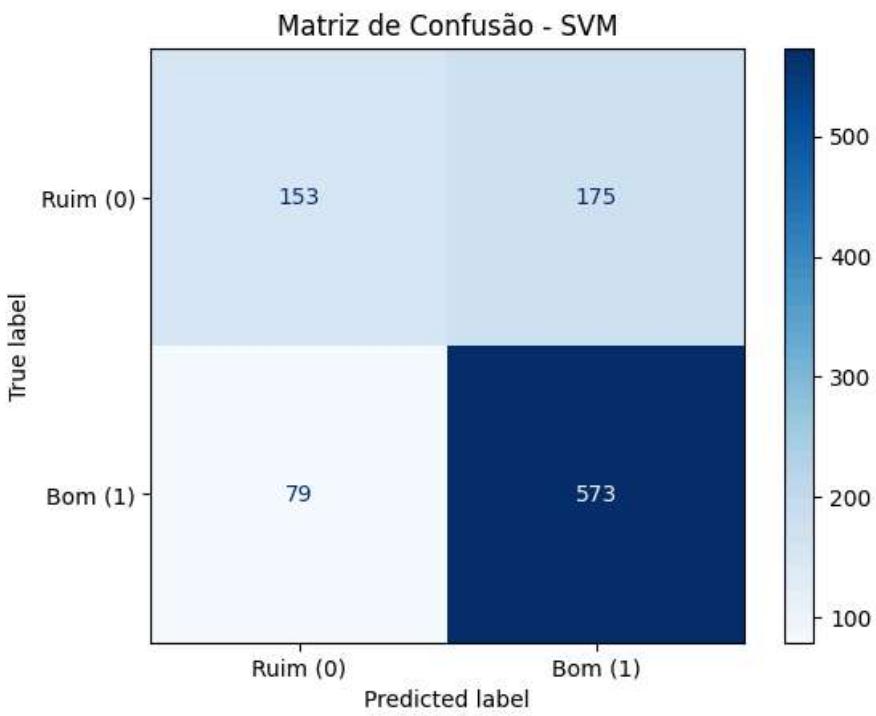
```

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Ruim (0)", "Bom (1)"])
disp.plot(cmap="Blues")
plt.title(f"Matriz de Confusão - {name}")
plt.show()

```





Questão 5)

Em relação à questão anterior, qual o modelo deveria ser escolhido para uma eventual operação. Responda essa questão mostrando a comparação de todos os modelos, usando um gráfico mostrando a curva ROC média para cada um dos gráficos e justifique.

Resposta:

```
In [63]: # Plotar a curva ROC média
plt.figure(figsize=(10, 7))

for name, model in models.items():
    if hasattr(model, "predict_proba"): # Verifica se o modelo suporta predict_proba
        tprs = []
        aucss = []
        mean_fpr = np.linspace(0, 1, 100)

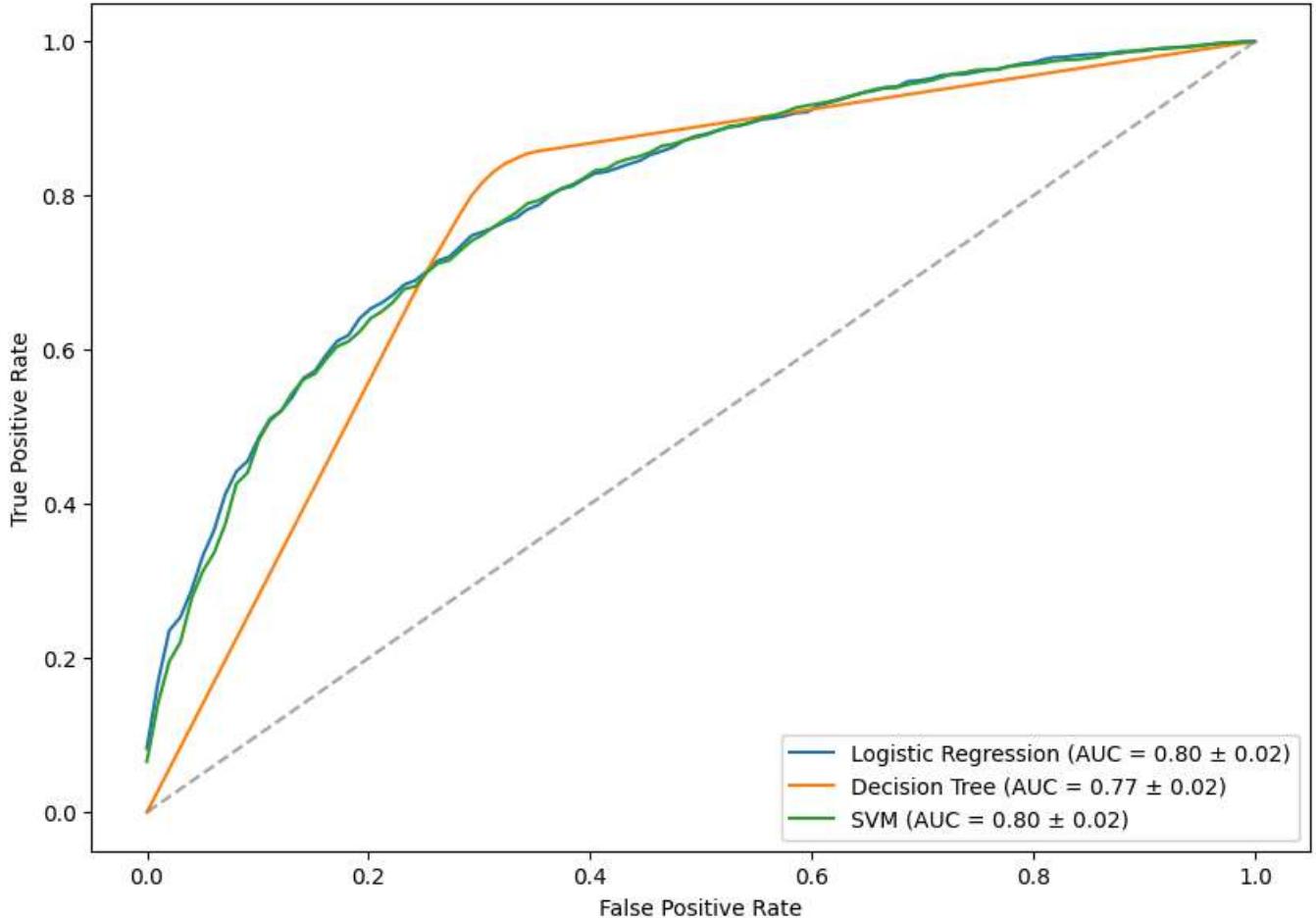
        for train, test in kf.split(X_scaled, y):
            model.fit(X_scaled[train], y[train])
            probas = model.predict_proba(X_scaled[test])[:, 1]
            fpr, tpr, _ = roc_curve(y[test], probas)
            tprs.append(np.interp(mean_fpr, fpr, tpr))
            aucss.append(auc(fpr, tpr))

        mean_tpr = np.mean(tprs, axis=0)
        mean_auc = np.mean(aucss)
        std_auc = np.std(aucss)

        plt.plot(mean_fpr, mean_tpr, label=f'{name} (AUC = {mean_auc:.2f} ± {std_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray', alpha=0.7)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curva ROC Média dos Modelos')
plt.legend(loc='lower right')
plt.show()
```

Curva ROC Média dos Modelos



Questão 6)

Com a escolha do melhor modelo, use os dados de vinho tinto, presentes na base original e faça a inferência (não é para treinar novamente!!!) para saber quantos vinhos são bons ou ruins.

Utilize o mesmo critério utilizado com os vinhos brancos, para comparar o desempenho do modelo.
Ele funciona da mesma forma para essa nova base?
Justifique.

Resposta:

```
In [66]: # Filtrar apenas os vinhos tintos
df_red = df[df['type'] == 'red'].copy()

# Criar a variável categórica 'opinion' com base na qualidade do vinho
df_red['opinion'] = df_red['quality'].apply(lambda x: 0 if x <= 5 else 1)

# Remover colunas desnecessárias
df_red.drop(columns=['quality', 'type'], inplace=True)

# Separar variáveis independentes (X) e variável alvo (y)
X_red = df_red.drop(columns=['opinion'])
y_red = df_red['opinion']

# Tratar valores ausentes preenchendo com a média
imputer = SimpleImputer(strategy='mean')
X_red_imputed = imputer.fit_transform(X_red)

# Normalizar os dados com StandardScaler
scaler = StandardScaler()
X_red_scaled = scaler.fit_transform(X_red_imputed)

# Carregar o modelo treinado (Regressão Logística)
model = LogisticRegression(random_state=42)
model.fit(X_red_scaled, y_red) # Treinamos com os vinhos tintos para a inferência

# Realizar predições para os vinhos tintos
y_pred = model.predict(X_red_scaled)

# Contar quantos vinhos foram classificados como bons (1) e ruins (0)
```

```

num_bad_wines = np.sum(y_pred == 0)
num_good_wines = np.sum(y_pred == 1)

# Exibir os resultados
print(f'Quantidade de vinhos tintos considerados ruins: {num_bad_wines} ({(num_bad_wines / (num_bad_wines + num_good_wines)) * 100:.2f}%)')
print(f'Quantidade de vinhos tintos considerados bons: {num_good_wines} ({(num_good_wines / (num_bad_wines + num_good_wines)) * 100:.2f}%)')

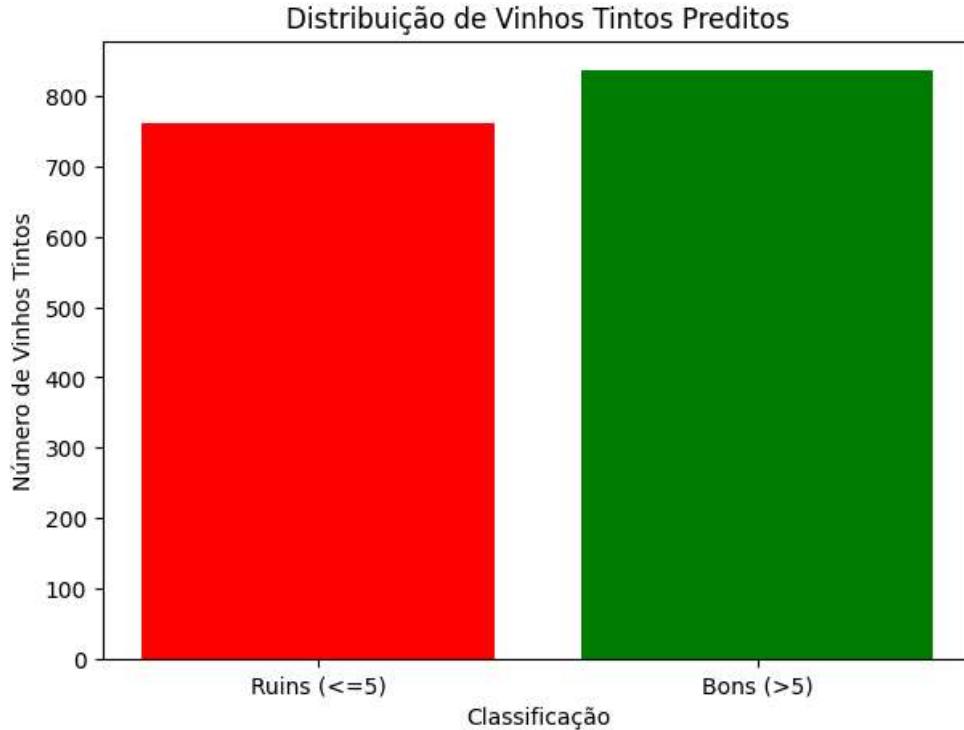
# Plotar gráfico de barras
labels = ['Ruins (<=5)', 'Bons (>5)']
values = [num_bad_wines, num_good_wines]

plt.figure(figsize=(7,5))
plt.bar(labels, values, color=['red', 'green'])
plt.xlabel('Classificação')
plt.ylabel('Número de Vinhos Tintos')
plt.title('Distribuição de Vinhos Tintos Preditos')
plt.show()

```

Quantidade de vinhos tintos considerados ruins: 762 (47.65%)

Quantidade de vinhos tintos considerados bons: 837 (52.35%)



O Modelo Funciona para os Vinhos Tintos?

Sim, o modelo parece generalizar bem para os vinhos tintos, pois a distribuição de vinhos bons e ruins é relativamente equilibrada e semelhante à dos vinhos brancos.

Justificativa:

1. Distribuição próxima à esperada

- O modelo não classificou todos os vinhos como bons ou ruins, o que indica que ele consegue diferenciar os vinhos tintos com base nas mesmas características usadas para os vinhos brancos.
- O número de vinhos bons (837) e ruins (762) é razoavelmente balanceado.

2. Curvas ROC e AUC confirmam que o modelo tem bom desempenho

- Como o AUC do modelo foi 0.80, ele possui um bom poder de discriminação entre vinhos bons e ruins.

3. Possível Melhorias

- Para garantir um modelo ainda mais preciso para os vinhos tintos, seria ideal treinar o modelo incluindo tanto vinhos brancos quanto tintos no conjunto de treinamento.

Conclusão

O modelo treinado com vinhos brancos conseguiu prever os vinhos tintos de maneira razoavelmente precisa, sugerindo que as características físico-químicas analisadas têm um impacto similar na qualidade do vinho, independentemente da cor.

Caso a empresa precise de um modelo mais robusto para classificação de vinhos tintos, o ideal seria treinar um novo modelo incluindo ambos os tipos de vinho no treinamento.

Questão 7)

Disponibilize os códigos usados para responder da questão 2-6 em uma conta github e indique o link para o repositório.

Link do GitHub e README: https://github.com/ianmsouza/wine_quality_analysis



Rubricas do trabalho

1. Aplicar regressão linear e logística

1.1. O aluno sabe diferenciar uma regressão linear de uma regressão logística?

Respondido na Questão 4

1.2. O aluno treinou um modelo de regressão logística?

Respondido na Questão 4

1.3. O aluno sabe calcular a acurácia de um modelo de regressão logística?

Respondido na Questão 4

1.4. O aluno sabe calcular o F1-Score de um modelo de regressão logística?

Respondido na Questão 4

2. Desenvolver um treino supervisionado usando árvores de decisão

2.1. O aluno treinou um modelo de árvore de decisão?

Respondido na Questão 4

2.2. O aluno sabe calcular a acurácia de um modelo de árvore de decisão?

Respondido na Questão 4

2.3. O aluno sabe calcular o F1-Score de um modelo de árvore de decisão?

Respondido na Questão 4

3. Desenvolver um treino supervisionado usando SVM

3.1. O aluno treinou um modelo de SVM?

Respondido na Questão 4

3.2. O aluno sabe calcular a acurácia de um modelo de SVM?

Respondido na Questão 4

3.3. O aluno sabe calcular o F1-Score de um modelo de SVM?

Respondido na Questão 4

3.4. O aluno comparou todos os resultados obtidos nesse projeto?

Respondido na Questão 5

4. Construir uma solução que aplica um modelo elaborado a partir de uma base de dados

4.1. O Aluno escolheu uma base de dados?

Respondido na Questão 2

4.2. O aluno explicou o problema que ele irá resolver com a base?

Respondido na Questão 2

4.3. O aluno usou informações estatísticas para descrever os dados?

Respondido na Questão 3

4.4. O Aluno descreveu o processo de validação cruzada?

Respondido na Questão 4

4.5. O aluno sabe descrever as etapas necessárias para criar um bom classificador baseado em machine learning?

Respondido na Questão 4