Projeto da Disciplina

Algoritmos de Inteligência Artificial para classificação - 25E1_2

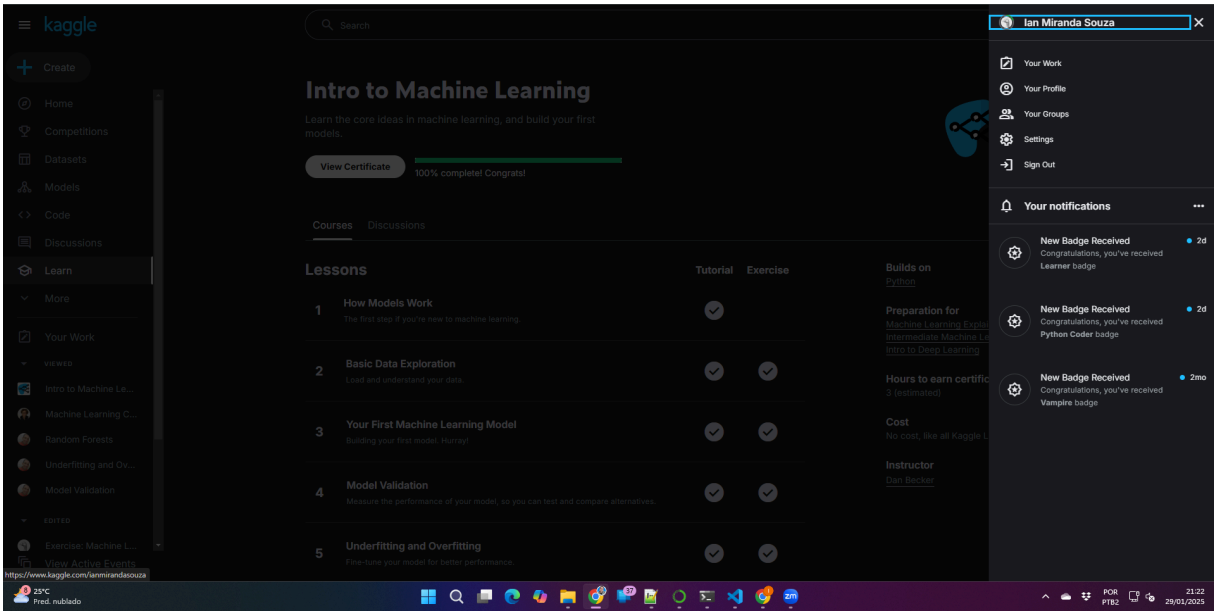
> Link do GitHub e README: https://github.com/ianmsouza/wine_quality_analysis

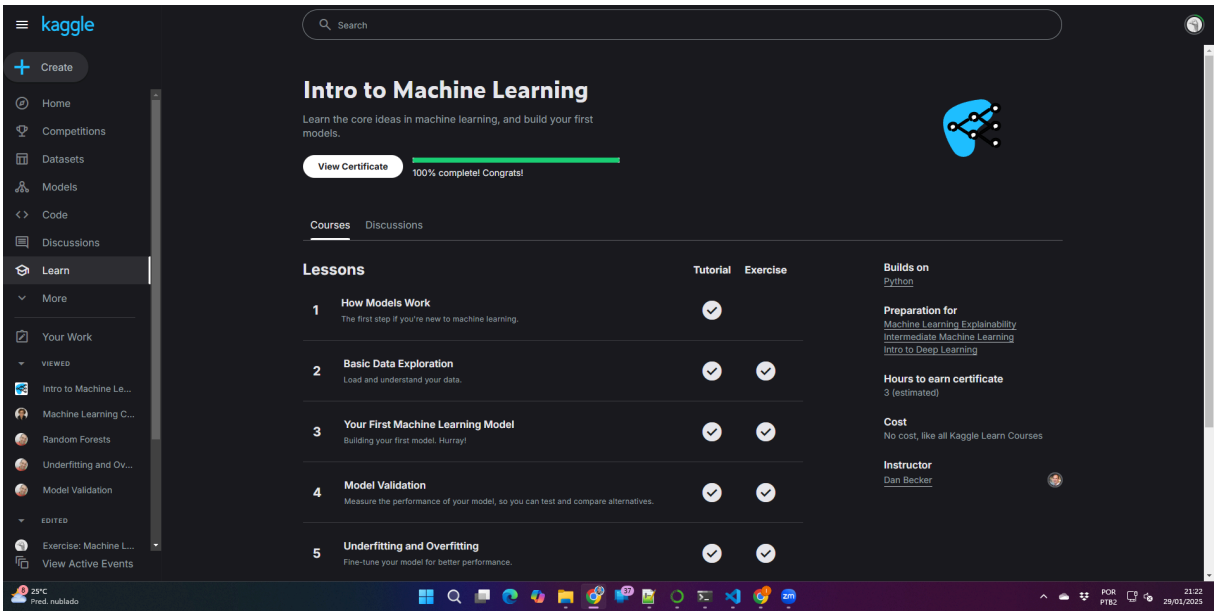
> Importação das bibliotecas necessárias para execução desse notebook

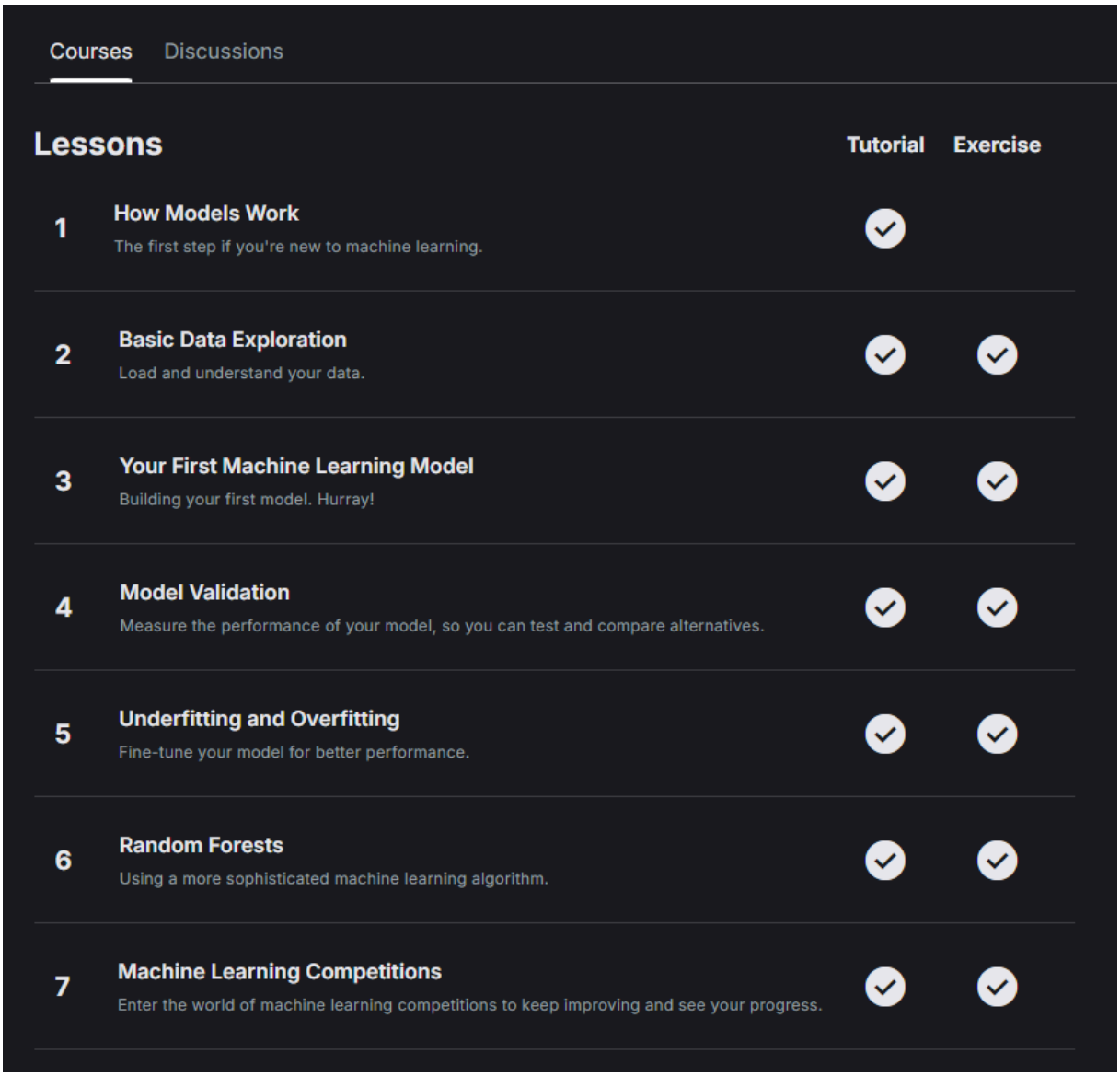
Questão 1)

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:




kaggle
Search
Create
Home
Competitions
Datasets
Models
Code
Discussions
Learn
More
Your Work
VIEWED
Intro to Machine Le...
Machine Learning C...
Random Forests
Underfitting and Ov...
Model Validation
EDITED
Exercise: Machine L...
View Active Events
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
1 How Models Work
The first step if you're new to machine learning.
2 Basic Data Exploration
Load and understand your data.
3 Your First Machine Learning Model
Building your first model. Hurray!
4 Model Validation
Measure the performance of your model, so you can test and compare alternatives.
5 Underfitting and Overfitting
Fine-tune your model for better performance.
Builds on
Python
Preparation for
Machine Learning Explainability
Intermediate Machine Learning
Intro to Deep Learning
Hours to earn certificate
3 (estimated)
Cost
No cost, like all Kaggle Learn Courses
Instructor
Dan Becker
25°C
Pred. nublado
POR
PTB2
21:22
29/01/2025


Courses
Discussions
Lessons
Tutorial
Exercise
1 How Models Work
The first step if you're new to machine learning.
2 Basic Data Exploration
Load and understand your data.
3 Your First Machine Learning Model
Building your first model. Hurray!
4 Model Validation
Measure the performance of your model, so you can test and compare alternatives.
5 Underfitting and Overfitting
Fine-tune your model for better performance.
6 Random Forests
Using a more sophisticated machine learning algorithm.
7 Machine Learning Competitions
Enter the world of machine learning competitions to keep improving and see your progress.

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

Questão 2)

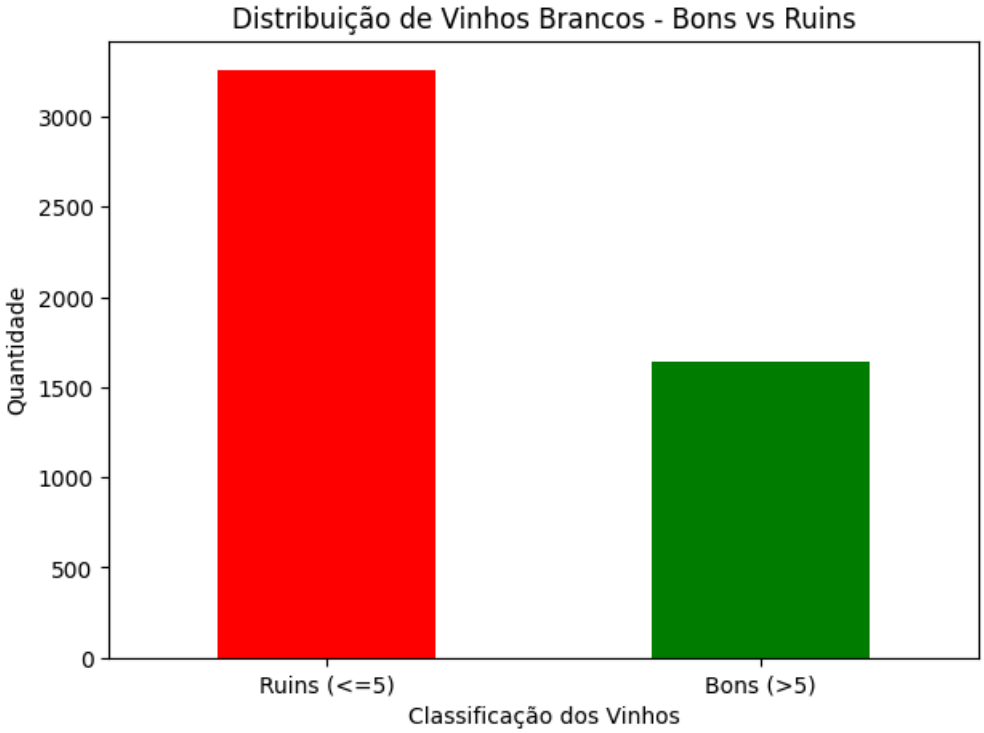
Faça o download da base - esta é uma base real, apresentada no artigo:
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
Ela possui uma variável denominada "quality", 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
> - 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 [18]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Carregando os dados
df = pd.read_csv(r'C:\Users\Ian\PythonProjects\infnet-25E1_2\datasets\winequalityN.csv', encoding='ISO-8859-1')

# 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())

# Salvando o novo dataset para análise posterior
df_white.to_csv(r'C:\Users\Ian\PythonProjects\infnet-25E1_2\datasets\winequality_white_filtered.csv', index=False)

# Criando gráfico de distribuição da variável 'opinion'
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()
```

```
    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
```



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 cloretos 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.

#	Variável	Descrição
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 [19]:
```
# Calcular médias e desvios padrão
statistics = df_white.describe().loc[['mean', 'std']]
print(statistics)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
mean       6.855532          0.278252     0.334250        6.393250   0.045778
std        0.843808          0.100811     0.120985        5.072275   0.021850

      free sulfur dioxide  total sulfur dioxide   density        pH  \
mean            35.308085            138.360657  0.994027  3.188203
std             17.007137             42.498065  0.002991  0.151014

      sulphates    alcohol   opinion
mean   0.489835  10.514267  0.665169
std    0.114147   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.

Etapa	Descrição
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.**

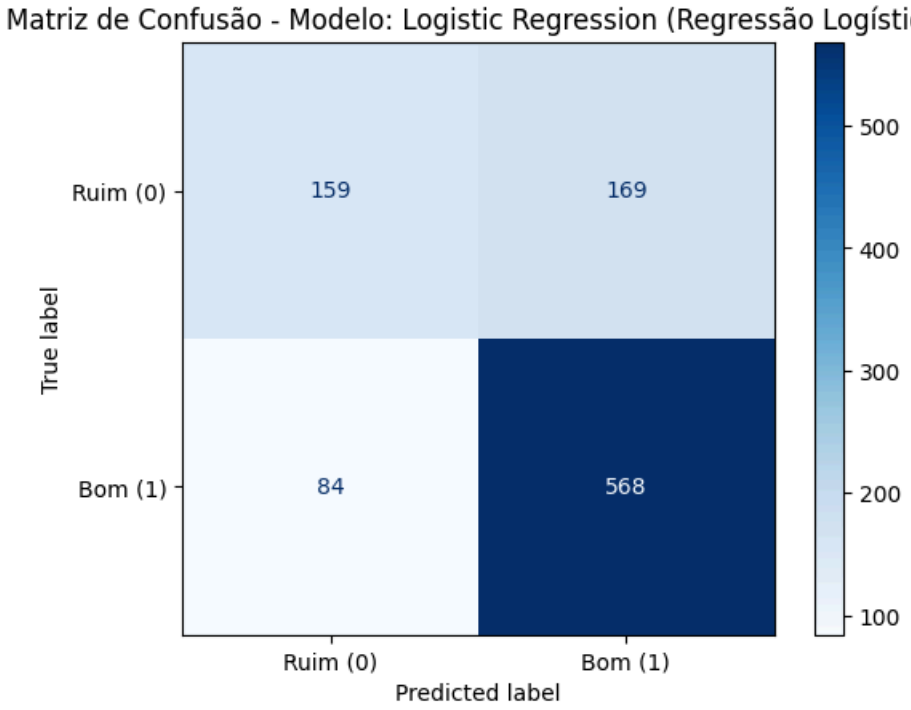
d) Treine um modelo de SVM usando um modelo de validação cruzada estratificada com k-folds (k=10) para realizar a classificação. Calcule para a base de teste:
- **i. a média e desvio da acurácia dos modelos obtidos;**
- **ii. a média e desvio da precisão dos modelos obtidos;**
- **iii. a média e desvio da recall dos modelos obtidos;**
- **iv. a média e desvio do f1-score dos modelos obtidos.**

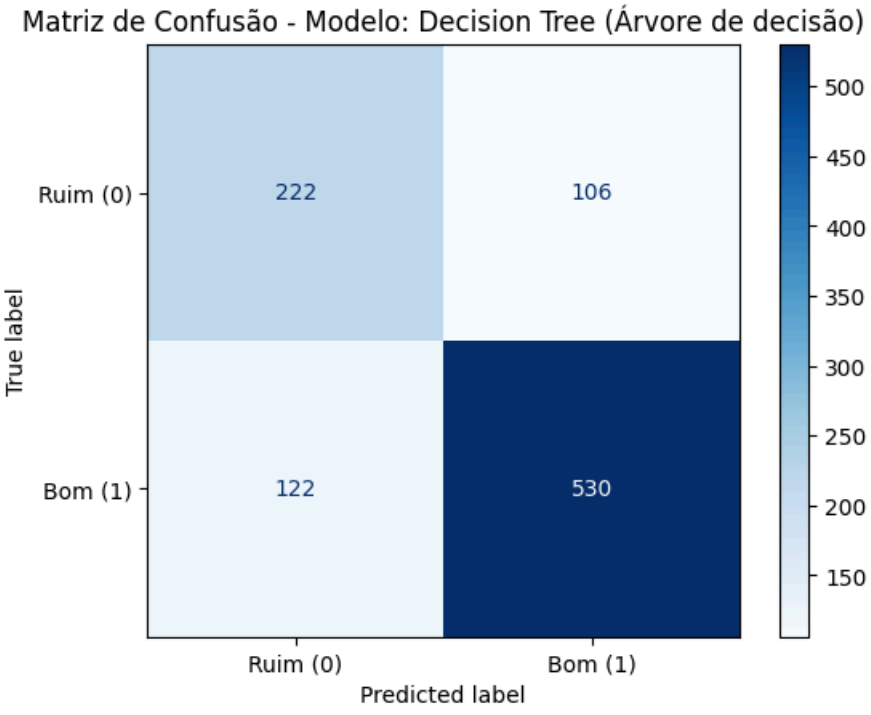
> Resposta:

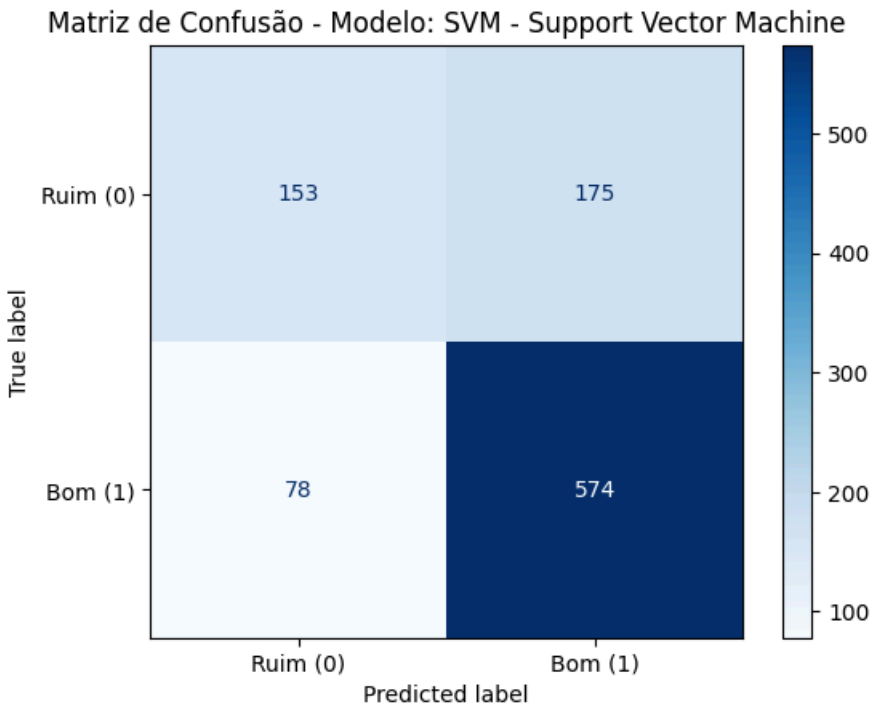
In [36]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.impute import SimpleImputer
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, Confu
import matplotlib.pyplot as plt

# Carregar os dados
df = pd.read_csv('C:/Users/Ian/PythonProjects/infnet-25E1_2/datasets/winequality_white_filtered.csv')

# Remover a coluna 'type' (não é necessária)
df.drop(columns=['type'], inplace=True)

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

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

# Normalizar os dados para melhorar a performance dos modelos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
```

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

# Lista de modelos
models = {
    'Logistic Regression (Regressão Logística)': LogisticRegression(random_state=42),
    'Decision Tree (Árvore de decisão)': DecisionTreeClassifier(random_state=42),
    'SVM - Support Vector Machine': SVC(kernel='linear', random_state=42)
}

# Métricas a serem avaliadas
scoring = {
    'accuracy': make_scorer(accuracy_score),
    'precision': make_scorer(precision_score),
    'recall': make_scorer(recall_score),
    'f1_score': make_scorer(f1_score)
}

# 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_name, metric in scoring.items():
        scores = cross_val_score(model, X_scaled, y, cv=kf, scoring=metric)
        model_results[metric_name] = {'mean': np.mean(scores), 'std': np.std(scores)}
        print(f"{metric_name}: Média = {np.mean(scores):.4f}, Desvio Padrão = {np.std(scores):.4f}")

    results[name] = model_results

# Gerar matrizes de confusão para cada modelo
for name, model in models.items():
    # Dividir os dados em treino e teste (80% treino, 20% teste)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

    # Treinar o modelo na base de treino
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Criar a matriz de confusão
    cm = confusion_matrix(y_test, y_pred)

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

# Converter os resultados para um DataFrame e salvar
df_results = pd.DataFrame.from_dict({(i, j): results[i][j] for i in results.keys() for j in results[i].keys()}, orient='i
df_results.to_csv('C:/Users/Ian/PythonProjects/infnet-25E1_2/datasets/model_results.csv')

print("\nResultados salvos em 'model_results.csv'!")
```

```
Treinando modelo: Logistic Regression (Regressão Logística)
accuracy: Média = 0.7495, Desvio Padrão = 0.0166
precision: Média = 0.7752, Desvio Padrão = 0.0154
recall: Média = 0.8788, Desvio Padrão = 0.0183
f1_score: Média = 0.8235, Desvio Padrão = 0.0112

Treinando modelo: Decision Tree (Árvore de decisão)
accuracy: Média = 0.7942, Desvio Padrão = 0.0247
precision: Média = 0.8481, Desvio Padrão = 0.0160
recall: Média = 0.8413, Desvio Padrão = 0.0300
f1_score: Média = 0.8445, Desvio Padrão = 0.0201

Treinando modelo: SVM - Support Vector Machine
accuracy: Média = 0.7528, Desvio Padrão = 0.0135
precision: Média = 0.7732, Desvio Padrão = 0.0112
recall: Média = 0.8895, Desvio Padrão = 0.0209
f1_score: Média = 0.8271, Desvio Padrão = 0.0101
```


Matriz de Confusão - Modelo: Logistic Regression (Regressão Logística)
Ruim (0)
Bom (1)
True label
159
169
84
568
Ruim (0)
Bom (1)
Predicted label
500
400
300
200
100


Matriz de Confusão - Modelo: Decision Tree (Árvore de decisão)
Ruim (0)
Bom (1)
True label
222
106
122
530
Ruim (0)
Bom (1)
Predicted label
500
450
400
350
300
250
200
150




```
Resultados salvos em 'model_results.csv'!
```

Modelo	Métrica	Média	Desvio Padrão
Logistic Regression (Regressão Logística)	accuracy	0.7495	0.0166
	precision	0.7752	0.0154
	recall	0.8788	0.0183
	f1_score	0.8235	0.0112
Decision Tree (Árvore de decisão)	accuracy	0.7942	0.0247
	precision	0.8481	0.0160
	recall	0.8413	0.0300
	f1_score	0.8445	0.0201
SVM (Support Vector Machine)	accuracy	0.7528	0.0135
	precision	0.7732	0.0112
	recall	0.8895	0.0209
	f1_score	0.8271	0.0101

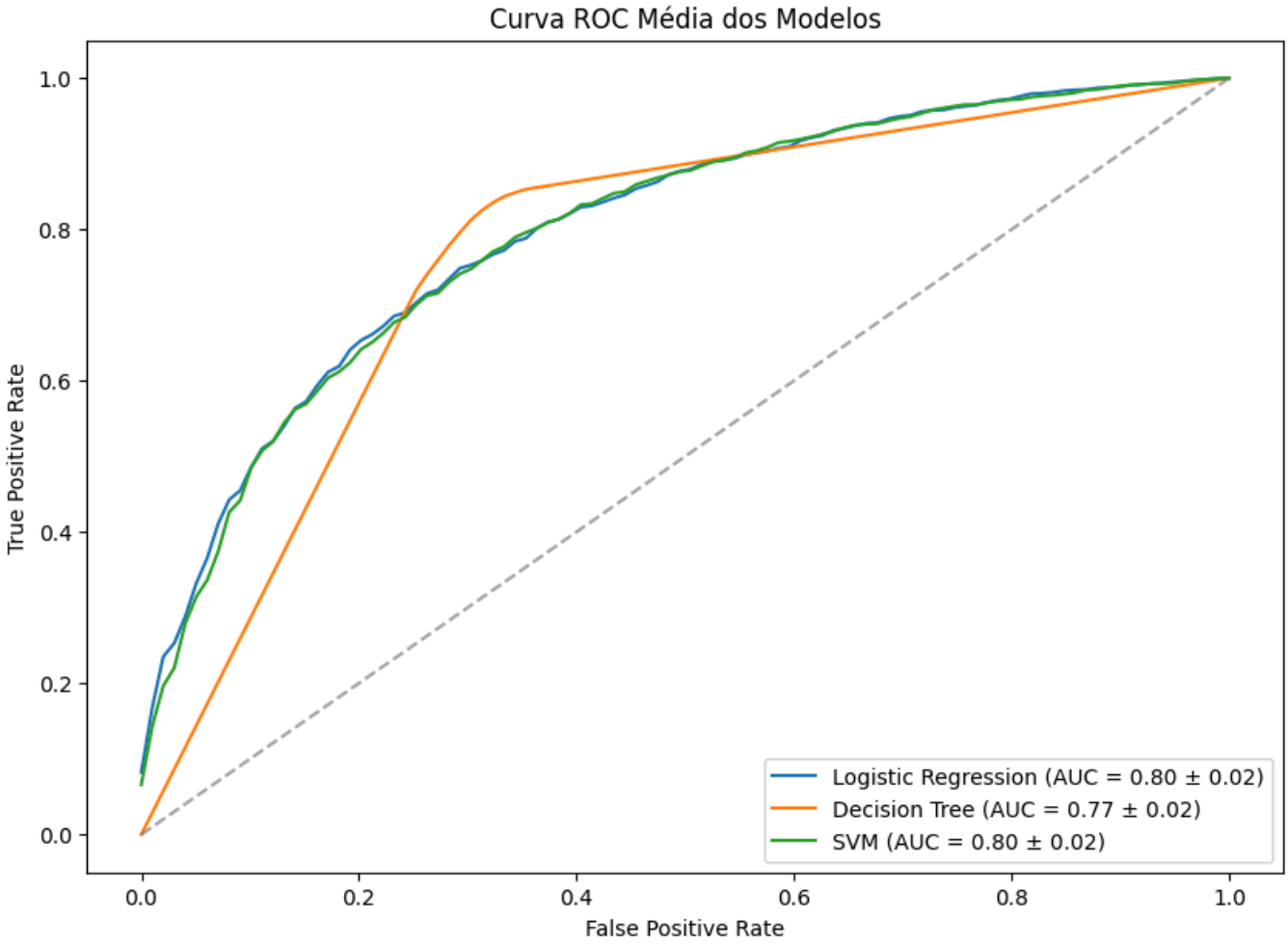
Questão 5)

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 [22]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

# Carregar os dados
df = pd.read_csv('C:/Users/Ian/PythonProjects/infnet-25E1_2/datasets/winequality_white_filtered.csv')
```

```
# Remover a coluna 'type'
df.drop(columns=['type'], inplace=True)

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

# Tratamento de valores ausentes
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

# Normalização dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

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

# Modelos a serem testados
models = {
    'Logistic Regression': LogisticRegression(random_state=42),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'SVM': SVC(kernel='linear', probability=True, random_state=42)
}

# Plotar a curva ROC média
plt.figure(figsize=(10, 7))

for name, model in models.items():
    tprs = []
    aucs = []
    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))
        aucs.append(auc(fpr, tpr))

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

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

# Configurar gráfico
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()
```



Questão 6)

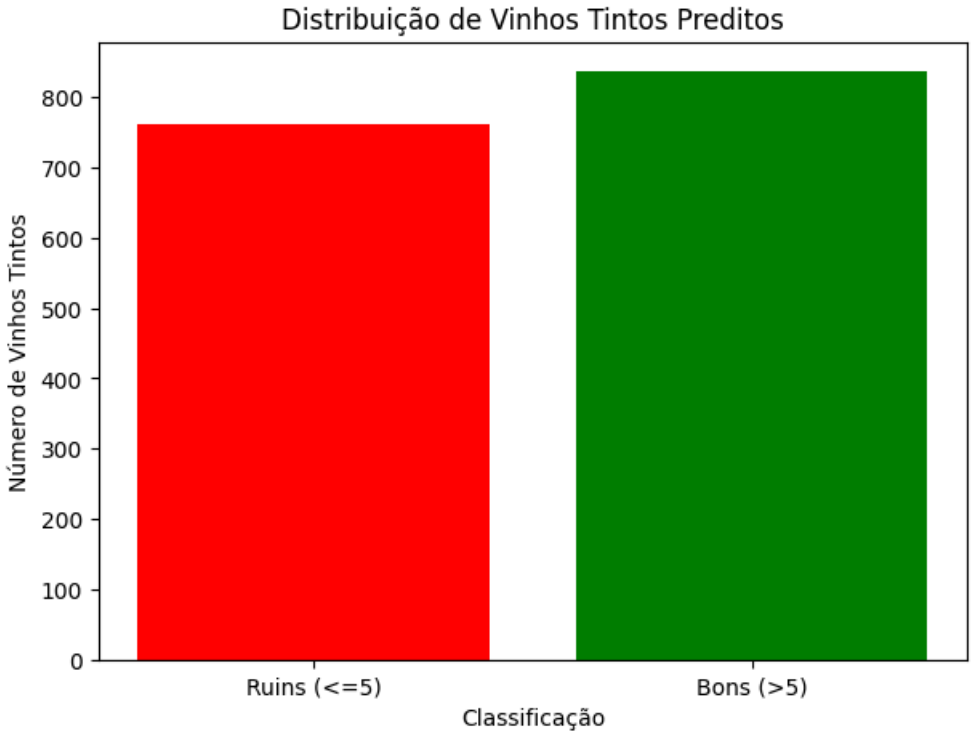
Com a escolha do melhor modelo, use os dados de vinho tinto, presentes na base original e faça a inferência (não é para treinar novamente!!!) para saber quantos vinhos são bons ou ruins.
Utilize o mesmo critério utilizado com os vinhos brancos, para comparar o desempenho do modelo.
Ele funciona da mesma forma para essa nova base?
Justifique.

> Resposta:

In [23]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

# Carregar os dados originais
df = pd.read_csv('C:/Users/Ian/PythonProjects/infnet-25E1_2/datasets/winequalityN.csv')

# 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}')
print(f'Quantidade de vinhos tintos considerados bons: {num_good_wines}')

# 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
Quantidade de vinhos tintos considerados bons: 837
```



> 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
>
> 
>