Projeto de Disciplina

Algoritmos de Inteligência Artificial para classificação [25E1_2]

Wilson Rodrigues Sampaio Melo

In [4]:
```
# instalação dos pacotes
!pip install pandas scikit-learn numpy matplotlib seaborn -q
```

```
[notice] A new release of pip is available: 23.2.1 -> 25.0.1
[notice] To update, run: pip3 install --upgrade pip
Reshimming asdf python...
```

In [22]:
```
# imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedKFold, cross_val_score, cross_val_
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
from sklearn.preprocessing import StandardScaler

# Para ignorar avisos desnecessários
import warnings
warnings.filterwarnings('ignore')

# para evitar notação científica no pandas
pd.options.display.float_format = '{:.0f}'.format
```

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:

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

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.

In [6]:
```
# Load data
data_path = "../data/winequalityN.csv"
wine_data = pd.read_csv(data_path)

# estatisticas da base de dados
wine_data.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
count	6487	6489	6494	6495	6495	6497	6497	6497	6488	
mean	7	0	0	5	0	31	116	1	3	
std	1	0	0	5	0	18	57	0	0	
min	4	0	0	1	0	1	6	1	3	
25%	6	0	0	2	0	17	77	1	3	
50%	7	0	0	3	0	29	118	1	3	
75%	8	0	0	8	0	41	156	1	3	
max	16	2	2	66	1	289	440	1	4	

In [7]:
```
# drop nulos e nan
print(wine_data.isna().sum())

# descatando os nan e nulls por entender que não representam
# uma fatia considerável da amostra e não deveria influenciar no resultado
wine_data.dropna(axis=0, inplace=True)
```

```
type                    0
fixed acidity          10
volatile acidity        8
citric acid             3
residual sugar          2
chlorides               2
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      9
sulphates               4
alcohol                 0
quality                 0
dtype: int64
```

In [8]:
```
# separar os dados apenas de vinhos brancos
white_wine_data = wine_data[wine_data['type'] == 'white'].copy()
red_wine_data = wine_data[wine_data['type'] == 'red'].copy()
```

In [9]:
```
# função auxiliar para criar a variável 'opinion'
# baseado na regra de negócio do enunciado baseado na
# na variável 'quality'
def create_opinion(df):
    # criar uma nova coluna chamada 'opinion'
    opinion = [0 if x <= 5 else 1 for x in df['quality']]
    df.loc[:, ('opinion')] = opinion
    return df
```

In [10]:
```
# cria a variável 'opinion' nas bases
white_wine_data = create_opinion(white_wine_data)
red_wine_data = create_opinion(red_wine_data)
```

```
# ver os dados
white_wine_data.loc[:,('type', 'quality', 'opinion')]
```

Out[10]:

	type	quality	opinion
0	white	6	1
1	white	6	1
2	white	6	1
3	white	6	1
4	white	6	1
...	...	...	...
4891	white	6	1
4893	white	6	1
4894	white	5	0
4896	white	7	1
4897	white	6	1

4870 rows × 3 columns

In [11]:
```
white_wine_data.columns
```

Out[11]:
```
Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
       'quality', 'opinion'],
      dtype='object')
```

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?

In [12]:
```
print("\nInformações dos dados dos vinhos brancos:")
print(white_wine_data.info())
```

```
Informações dos dados dos vinhos brancos:
<class 'pandas.core.frame.DataFrame'>
Index: 4870 entries, 0 to 4897
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   type                  4870 non-null   object
 1   fixed acidity         4870 non-null   float64
 2   volatile acidity      4870 non-null   float64
 3   citric acid           4870 non-null   float64
 4   residual sugar        4870 non-null   float64
 5   chlorides             4870 non-null   float64
 6   free sulfur dioxide   4870 non-null   float64
 7   total sulfur dioxide  4870 non-null   float64
 8   density               4870 non-null   float64
 9   pH                    4870 non-null   float64
 10  sulphates             4870 non-null   float64
 11  alcohol               4870 non-null   float64
 12  quality               4870 non-null   int64
 13  opinion               4870 non-null   int64
dtypes: float64(11), int64(2), object(1)
memory usage: 570.7+ KB
None
```

In [13]:
```
# Estatísticas descritivas para variáveis contínuas
desc_stats = white_wine_data.describe().T
print("\nEstatísticas descritivas:")
print(desc_stats[['mean', 'std']])
```

```
Estatísticas descritivas:
                       mean  std
fixed acidity             7    1
volatile acidity          0    0
citric acid               0    0
residual sugar            6    5
chlorides                 0    0
free sulfur dioxide      35   17
total sulfur dioxide    138   42
density                   1    0
pH                        3    0
sulphates                 0    0
alcohol                  11    1
quality                   6    1
opinion                   1    0
```

A base contém 14 variáveis, das quais a maioria são contínuas (medindo propriedades fisico-químicas) e duas são discretas/categóricas:

type (categorical) quality (discrete), que é transformada em opinion (discrete/categórica) para a análise de classificação.

4. Com a base escolhida:

```
- Descreva as etapas necessárias para criar um modelo de
classificação eficiente.

- 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:
```

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

- 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:
    - a média e desvio da acurácia dos modelos obtidos;
    - a média e desvio da precisão dos modelos obtidos;
    - a média e desvio da recall dos modelos obtidos;
    - a média e desvio do f1-score dos modelos obtidos.

- 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:
    - a média e desvio da acurácia dos modelos obtidos;
    - a média e desvio da precisão dos modelos obtidos;
    - a média e desvio da recall dos modelos obtidos;
    - a média e desvio do f1-score dos modelos obtidos.

In [14]:
```
# Separa as features (atributos fisico-químicos) e o target (opinion).
y = white_wine_data['opinion']
X = white_wine_data.drop(columns=['type', 'quality', 'opinion'], axis='columns',
```

In [15]:
```
# aplicar uma padronização
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [16]:
```
random_state = 42

# organização dos modelos para pipeline
modelos = {
    'Logistic Regression': LogisticRegression(random_state=random_state, solver=
    'Decision Tree': DecisionTreeClassifier(random_state=random_state),
    'SVM': SVC(random_state=random_state, probability=True)
}

# k-folds (k=10) para validação cruzada
folds = 10

# Cria um objeto para a validação cruzada estratificada
skf = StratifiedKFold(n_splits=folds, shuffle=True, random_state=random_state)
```

In [17]:
```
# Função auxiliar para calcular as métricas em cada fold e retornar média e desv
def avaliar_modelo(modelo, X, y):
    # Armazenar os resultados de cada métrica
    accuracy_list = []
    precision_list = []
    recall_list = []
    f1_list = []

    # Para armazenar as predições probabilísticas e verdadeiros para a curva ROC
    y_probs = np.array([])
    y_true_total = np.array([])

    for train_index, test_index in skf.split(X, y):
```

```
        # Divide os dados em treino e teste
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # Treina o modelo
        modelo.fit(X_train, y_train)

        # Predição das classes
        y_pred = modelo.predict(X_test)

        # Predição probabilística para a classe 1 (para ROC)
        y_prob = modelo.predict_proba(X_test)[:, 1]

        # Armazena os resultados para ROC
        y_true_total = np.concatenate([y_true_total, y_test])
        y_probs = np.concatenate([y_probs, y_prob])

        # Calcula as métricas
        accuracy_list.append(accuracy_score(y_test, y_pred))
        precision_list.append(precision_score(y_test, y_pred))
        recall_list.append(recall_score(y_test, y_pred))
        f1_list.append(f1_score(y_test, y_pred))

    # Calcula as médias e desvios padrões das métricas
    metrics = {
        'accuracy_mean': np.mean(accuracy_list),
        'accuracy_std': np.std(accuracy_list),
        'precision_mean': np.mean(precision_list),
        'precision_std': np.std(precision_list),
        'recall_mean': np.mean(recall_list),
        'recall_std': np.std(recall_list),
        'f1_mean': np.mean(f1_list),
        'f1_std': np.std(f1_list),
        'y_true': y_true_total,
        'y_probs': y_probs
    }

    return metrics
```

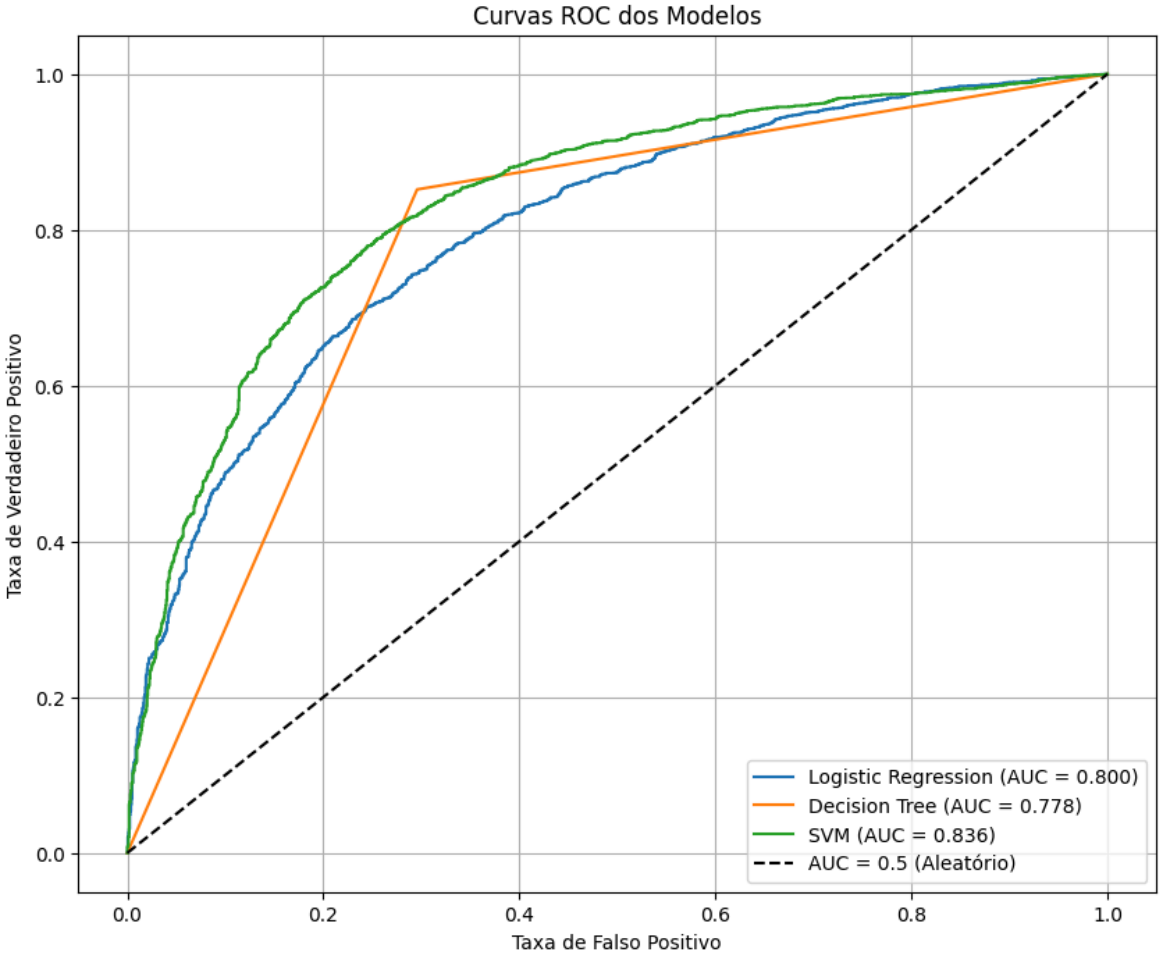
In [18]:

```
# Dicionário para armazenar os resultados de cada modelo
resultados = {}

# Itera sobre cada modelo e avalia
for nome, modelo in modelos.items():
    print(f"\nAvaliando o modelo: {nome}")
    # usando o X_scaled porque a padronização é recomendada para SVM e melhora a
    metrics = avaliar_modelo(modelo, X_scaled, y)
    resultados[nome] = metrics

    # Exibe as métricas calculadas
    print(f"Acurácia: {metrics['accuracy_mean']:.3f} ± {metrics['accuracy_std']:
    print(f"Precisão: {metrics['precision_mean']:.3f} ± {metrics['precision_std'
    print(f"Recall: {metrics['recall_mean']:.3f} ± {metrics['recall_std']:.3f}")
    print(f"F1-score: {metrics['f1_mean']:.3f} ± {metrics['f1_std']:.3f}")
```

```
Avaliando o modelo: Logistic Regression
Acurácia: 0.750 ± 0.018
Precisão: 0.775 ± 0.015
Recall: 0.879 ± 0.012
F1-score: 0.824 ± 0.012

Avaliando o modelo: Decision Tree
Acurácia: 0.802 ± 0.012
Precisão: 0.852 ± 0.014
Recall: 0.852 ± 0.014
F1-score: 0.852 ± 0.009

Avaliando o modelo: SVM
Acurácia: 0.787 ± 0.018
Precisão: 0.812 ± 0.014
Recall: 0.885 ± 0.012
F1-score: 0.847 ± 0.012
```

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.

In [23]:
```
plt.figure(figsize=(10, 8))

for nome, metrics in resultados.items():
    fpr, tpr, thresholds = roc_curve(metrics['y_true'], metrics['y_probs'])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{nome} (AUC = {roc_auc:.3f})')

plt.plot([0, 1], [0, 1], 'k--', label='AUC = 0.5 (Aleatório)')
plt.xlabel('Taxa de Falso Positivo')
plt.ylabel('Taxa de Verdadeiro Positivo')
plt.title('Curvas ROC dos Modelos')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



In [20]:
```
# Seleciona o modelo com melhor F1-score médio
melhor_modelo_nome = max(resultados, key=lambda nome: resultados[nome]['f1_mean'
print(f"\nMelhor modelo escolhido: {melhor_modelo_nome}")

melhor_modelo = modelos[melhor_modelo_nome]
melhor_modelo.fit(X_scaled, y)
```

```
Melhor modelo escolhido: Decision Tree
```

Out[20]:

DecisionTreeClassifier

```
DecisionTreeClassifier(random_state=42)
```

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.

In [21]:
```
# Preparação dos dados dos vinhos tintos
X_red = red_wine_data.drop(columns=['type', 'quality', 'opinion'], axis=1)
y_red = red_wine_data['opinion']

# Aplica o mesmo tratamento de padrnização padronização dos dados
X_red_scaled = scaler.transform(X_red)

# Realiza a predição com o melhor modelo
if melhor_modelo_nome == 'Linear Regression':
```

```
    # Para regressão linear, converte a saída contínua para 0 ou 1
    y_red_cont = melhor_modelo.predict(X_red_scaled)
    y_red_pred = (y_red_cont >= 0.5).astype(int)
else:
    y_red_pred = melhor_modelo.predict(X_red_scaled)

# Calcula as métricas na base de vinhos tintos
accuracy_red = accuracy_score(y_red, y_red_pred)
precision_red = precision_score(y_red, y_red_pred)
recall_red = recall_score(y_red, y_red_pred)
f1_red = f1_score(y_red, y_red_pred)

print("\nResultados de inferência para os vinhos tintos usando o melhor modelo t
print(f"Acurácia: {accuracy_red:.3f}")
print(f"Precisão: {precision_red:.3f}")
print(f"Recall: {recall_red:.3f}")
print(f"F1-score: {f1_red:.3f}")
```

```
Resultados de inferência para os vinhos tintos usando o melhor modelo treinado:
Acurácia: 0.537
Precisão: 0.592
Recall: 0.428
F1-score: 0.497
```

O desempenho do melhor modelo de vinhos brancos não performou bem na base de vinhos tintos. Penso que era algo esperado pela distinção de tipos e propriedades, uma vez que a base de treino não tinha nenhum vinho tinto o modelo fica naturalmente desbalanceado o que prejudica o desempenho da inferência.

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

https://github.com/bakudas/pos-infnet-classificacao