Aula06 - Assícrona - Atividade Prática 01

Implementação de modelos e verificação de eficiência da estratégia adotada em problemas de classificação

Esta atividade corresponde à nossa primeira atividade **valorizada** do curso de *Machine Learning*.

Instruções:

1 - Reunam-se em grupos de **no máximo 3 pessoas**;

2 - A entrega deve ser realizada via CANVAS através de um artigo no formato PDF gerado a partir do *notebook*. Para tal fim, os grupos poderão utilizar o *Jupyter Notebook* ou o *Google Drive* para implementação do que foi solicitado.

3 - O período de disponibilidade para recebimento da atividade será de **24/08/23 às 17h, até 31/08/23, às 17h**.

4 - No cabeçalho do documento devem constar o **nome completo e o RA de TODOS os membros do grupo**.

Práticas a serem desenvolvidas:

1 - Criação de um *dataset* sintético para avaliação da tarefa de classificação;

2 - Divisão de dados em treinamento e teste;

3 - Criação e aplicação de um modelo de *Machine Learning*;

4 - Verificação da eficiência do modelo na tarefa de classificação através da acurácia.

Bibliotecas a serem utilzadas:

In [29]:
```
# Importando o modulo para criarmos datasets sinteticos
from sklearn.datasets import make_classification

# Importando bibliotecas de plotagem e ajustes de tamanho de figuras
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab

params = {'legend.fontsize': 20,
          'figure.figsize': (15,5),
         'axes.labelsize': 20,
         'axes.titlesize': 20,
         'xtick.labelsize': 20,
         'ytick.labelsize': 20}

# Importando a função de divisão dos dados em treinamento e teste
from sklearn.model_selection import train_test_split

# Importando o modelo do Decision Tree
from sklearn.tree import DecisionTreeClassifier

# Importando a métrica de avaliação do modelo
from sklearn.metrics import accuracy_score
```

Passo 1 - Criação de um *dataset* genérico

O Python propcia testes para averiguação de eficiência de algoritmos de classificação através do uso do comando `make_classification`. Este comando está incluído no módulo `datasets` que existe dentro de uma biblioteca do Python, chamada `sklearn`. A documentação desta biblioteca está disponível no link: https://scikit-learn.org/stable/.

A função `make_classification` possui os seguintes parâmetros:

- `n_samples`: Indica o número de linhas que existirão dentro do banco de dados a ser criado;
- `n_features`: Indica o número de colunas que existirão no *dataset*;
- `n_informative`: Número de colunas que serão úteis para o processo de tomada de decisão. Ou seja, o número de colunas que serão mais relevantes dentro do banco de dados;
- `n_redundant`: Número de colunas que são redundantes dentro do banco de dados. utilizado para criar uma maior complexidade ao problema;
- `n_classes`: Número de classes diferentes às quais os padrões criados pertencem;
- `random_state`: Uma vez que os dados criados são aleatórios, prender a semente faz com que os resultados alcançados sejam reprodutíveis quando o código for colocado para rodar em outro momento, ou outro computador.

Dessa forma, vamos utilizar a função e atribuir os dados a duas variáveis, `X` e `y`. A variável `X` receberá os dados, as colunas e linhas, que correspondem aos dados das colunas que contém dados "medidos", enquanto `y`, corresponderá à variável *target*. Ou seja, as classes existentes no problema.

Criaremos o banco de dados a partir das linhas de código indicadas abaixo:

In [10]:
```
# Gera dados aleatórios
X, y = make_classification(n_samples = 1000, n_features = 5,
                           n_informative = 3, n_redundant = 1,
                           n_classes = 3, random_state = 42)
```

Vamos verificar as dimensões de linhas e colunas da variável `X`:

In [11]:
```
print(X.shape)
```

```
(1000, 5)
```

Através do método `shape`, conseguimos verificar que `X` tem 1000 linhas e 5 colunas. Estes foram exatamente os valores passados como parâmetros para `n_samples` e `n_features`, na função `make_classification`.

Vamos verificar as primeiras linhas do banco de dados, através da linha de comando abaixo:

In [12]:
```
print(X[:5,:])
```

```
[[-2.52469981  0.44395622  1.52290093 -0.74224801  2.6846391 ]
 [ 2.00753252 -1.46309596 -0.88517443  1.17754472 -0.20972932]
 [ 1.32100607 -2.4018553  -1.42626134  1.29163523 -1.24303485]
 [-1.19770619 -0.76353166  0.47551907 -1.10995284  1.44762553]
 [ 1.36709886 -0.06586396 -0.38047272 -0.67419056 -0.34760868]]
```

Este comando de escrita na tela, `print(X[:5,:])`, tem como parâmetro principal a variável `X`, que contém o banco de dados. Ao utilizarmos os colchetes, estamos indicando que não queremos ver o banco de dados inteiro, mas sim uma parte específica dele. Deesa forma, ao utilizarmos `:5`, estamos indicando ao Python que queremos ver as primeiras 5 linhas existentes na matriz do banco de dados, enquanto o comando `:` após a virgula indica que pedimos ao código para mostrar todas as colunas existentes, limitadas pelo número de linhas já indicando antes da vírgula.

Vamos verificar também a variável `y`. Neste caso, vamos pedir ao código para que sejam mostradas as 10 primeiras amostras que compõem a variável em questão. Para isso, utilizaremos a seguinte linha:

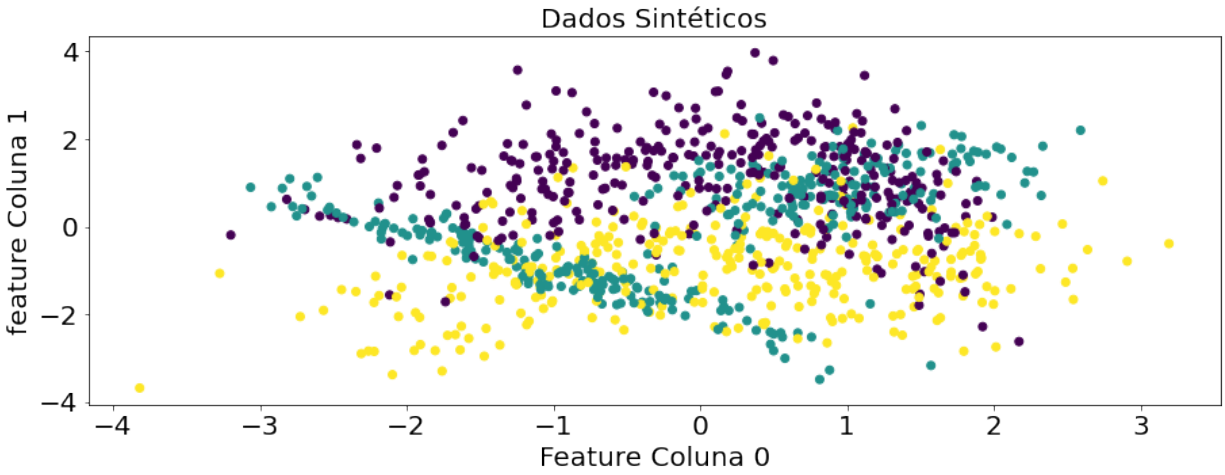
In [13]:
```
print(y[:10])
```

```
[1 2 2 1 2 1 2 0 2 0]
```

Como pode ser observado, três valores diferentes são mostrados: `0`, `1` e `2`. Tal fato se deve ao número de classes definido inicialmente como parâmetro da função `make_classification`.

Vamos analisar visualmente os dados gerados através da biblioteca `matplotlib`. Para isso, vamos considerar **arbitrariamente** as colunas 0 e 1 da matrix `X`.

In [14]:
```
# Visualizando os dados graficamente
pylab.rcParams.update(params)
fig, ax = plt.subplots()
ax.scatter(X[:,0], X[:,1], c = y)
ax.set_xlabel('Feature Coluna 0')
ax.set_ylabel('feature Coluna 1')
ax.set_title('Dados Sintéticos')
plt.show()
```



Na Figura acima, cada uma das cores representa uma classe diferente no problema. É possível perceber que os padrões estão "embaralhados", o que dificulta o processo de separação entre as classes e, consequentemente, da classificação.

Passo 2 - Divisão dos dados em Dados de Treinamento e Dados de Teste

Para realizar esta divisão, utilizaremos a função `train_test_split`, disponível na biblioteca `sklearn`, comentada na Aula 04, síncrona, do dia 22/08/23. Portanto, utilizaremos a seguinte linha de código:

Neste caso, estaremos utilizando a proporção de 80% dos dados destinados à etapa de treinamento e 20% dos dados destinados à etapa de teste.

In [19]:
```
# Divisao dos dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
                        test_size = 0.2, random_state = 42)
```

Podemos verificar as dimensões das variáveis para assegurar que a divisão tenha sido feita corretamente. Para isso, utilizaremos:

Para os dados que envolvem o processo de treinamento `X_train` e `y_train`:

In [20]:
```
print("As dimensões de X_train:", X_train.shape)
print("As dimensões de y_train:", y_train.shape)
```

```
As dimensões de X_train: (800, 5)
As dimensões de y_train: (800,)
```

Das variáveis da etapa de teste `X_test` e `y_test`:

In [21]:
```
print("As dimensões de X_test:", X_test.shape)
print("As dimensões de y_test:", y_test.shape)
```

```
As dimensões de X_test: (200, 5)
As dimensões de y_test: (200,)
```

Com estas informações, podemos ver que a divisão foi feita corretamente.

Passo 3 - Criação e aplicação de um modelo de *Machine Learning*

Neste primeiro problema, utilizaremos um dos algoritmos clássicos mais famosos do aprendizado de máquina, a **Árvore de Decisão**, ou, do inglês ***Decision Tree***. A Árvore de Decisão também se encontra disponívelna biblioteca `sklearn` e pode ser utilizada por meio da função `DecisionTreeClassifier`, que é responsável por criar o modelo do algoritmo para ser utilizado no problema.

Podemos criar o modelo utilizando a seguinte linha de código:

In [24]:
```
# Criando o modelo do Decision Tree
clf_dt = DecisionTreeClassifier(random_state = 42)
```

Com o modelo criado e armazenado na variável `clf_dt`, podemos aplicar o método `.fit` para realizar o treinamento do algoritmo *Decision Tree* para que o mesmo aprenda sobre os modelos existentes no banco de dados. Faremos este procedimento utilizando a seguinte linha de código:

In [25]:
```
# Treinamento
clf_dt.fit(X_train,y_train)
```

Out[25]:
```
DecisionTreeClassifier(random_state=42)
```

Agora, a variável `clf_dt` armazena também as informações que foram obtidas durante o período de treinamento. E assim podemos ver o comportamento do algoritmo quando exposto a dados desconhecidos por ele até o momento, os dados de teste. Para isso, utilizaremos o método `.predict`.

In [26]:
```
# Teste
y_pred = clf_dt.predict(X_test)
```

A variável `y_pred` armazena todas as predições do algoritmo classificador. Ela possui a mesma dimensão (tamanho) da variável `y_test`, como ser visto abaixo:

In [27]:
```
print(y_pred.shape)
```

```
(200,)
```

Isto se deve ao fato de que para cada padrão presente no conjunto de dados de texte, o modelo realizou uma predição indicando à qual classe aquele padrão pertence e o armazenou na variável `y_pred`. Podemos verificar a eficiência do modelo ao utilizar a métrica da acurácia, disponível também na biblioteca `sklearn`.

In [30]:
```
# Avaliando o modelo
acc_dt = accuracy_score(y_test,y_pred)
print("Resultado da classificação:", round(acc_dt,2)*100)
```

```
Resultado da classificação: 87.0
```

Exercícios

Questão 1) Reproduza os processos indicados no exemplo anterior alterando os seguintes parâmtros na função `make_classification`:

- `n_samples`: 2000;
- `n_features`: 10;
- `n_informative`: 3;
- `n_redundant`: 2;
- `n_classes`: 3;
- `random_state`: 42.

Indique a porcentagem de acurácia do modelo criado.

Questão 2) Faça as seguintes alterações e verifique os resultados:

- Nas funções `make_classification`, `train_test_split` e `DecisionTreeClassifier` mude o `random_state` para 0;
- Mude a proporção da divisão dos dados de treinamento e teste para 70% e 30%, respectivamente.

Indique a porcentagem de acurácia do modelo.

Questão 3) Explique as diferenças de resultados obtidos nas questões 1 e 2.