SOFTEX/UNIFEI - Curso de Introdução à Inteligência Artificial

3. Aprendizado de Máquina

Notebook 3 - Aprendizado Supervisionado

Autores:

Rafael Frinhani
Rodrigo Lima

Tópico: Processo de Machine Learning - Desenvolvimento do Modelo

Objetivo: Este notebook mostra o processo de Desenvolvimento do Modelo de *Machine Learning*, mais especificamente a aplicação de algoritmos de aprendizado Supervisionado para a tarefa de classificação e regressão.

Organização do Notebook: Os exemplos a seguir consideram um conjunto de dados biológicos bem conhecido na literatura. Será aplicado os modelos de classificação Naive Bayes e Árvore de Decisão, e o modelo de Regressão Logística.

+ Código + Texto

Conjunto de Dados

O conjunto de dados utilizado neste notebook é chamado Iris, cujos arquivos e demais informações constam no UCI Machine Learning Repository. O cenário do problema associado ao conjunto de dados é o de uma classificação taxonomica de três espécies da flores Iris (Íris-Setosa, Íris-Versicolor e Íris-Virginica), que embora sejam visualmente semelhantes possuem, possuem diferenças anatômicas em relação a medidas de comprimento e largura das sépalas e das pétalas. O conjunto de dados possui um total de 150 flores de íris, sendo 50 amostras para cada uma das espécies.

Fonte: Wikipedia (Flores de Iris)

O conjunto de dados pode ser importado diretamente do repositório da UCI, sua carga é feita em um *DataFrame* Pandas, seguida da visualização dos primeiros registros conforme a seguir:

```
import pandas as pd
import numpy as np
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=None)
df.head()
```

Para melhorar a leitura do conjunto de dados as colunas dos atributos são renomeadas:

```
df.rename(columns = {0:'comp_sepala',
                     1:'larg_sepala',
                     2:'comp_petala',
                     3:'larg_petala',
                     4:'classe'}, inplace = True)
df
```

CLASSIFICAÇÃO

Modelo - Naive Bayes

Definidos os atributos que serão utilizados, o objetivo é desenvolver um modelo de classificação com o método Naive Bayes. Esse método é de funcionamento relativamente simples, não requer o ajuste de muitos parâmetros, possuindo desempenho comparável com métodos como Árvore de Decisão e Redes Neurais em determinados casos.

Preparação do Conjunto de Dados

Por se tratar de um aprendizado supervisionado, o objetivo é a identificação de uma classe previamente informada considerando um conjunto de treinamento e um de teste. Para isso, o conjunto é dividido em um conjunto de amostras denominado *X*, e um de rótulos denominado *y*. No conjunto *X* foi considerado apenas os atributos comprimento da pétala (comp_petala) e largura da pétala (larg_petala).

```
# Divisão do conjunto de dados X (atributos) e y (classes)
X = df[['comp_petala',
        'larg_petala']]
y = df['classe']
print("Conjunto X (atributos):",'\n' , X)
print("Conjunto Y (classes):",'\n', y)
```

O conjunto *Y* armazena os rótulos das classes no formato nominal (textual), mas muitos algoritmos de aprendizado de máquina são modelos computacionais que manipulam informações numéricas. O comando a seguir transforma os rótulos de classe do tipo nominal para numérico conforme o seguinte mapeamento: 0 = Iris-setosa; 1 = Iris-versicolor; 2 = Iris-virginica.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y
```

Divisão do Conjunto de Dados

Uma vez que os dados estão no formato adequado, o conjunto de dados é dividido em conjuntos de treinamento e de teste. A divisão do conjunto possibilita validar se o modelo é capaz de executar com qualidade a classificação dos dados. No caso deste exemplo o conjunto de treinamento corresponde a 80% do conjunto de dados original e o conjunto de testes 20%.

```
from sklearn.model_selection import train_test_split

X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, train_size=0.8, random_state=42) # divisão do conjunto de dados X e y em tr
print("Quantidade de amostras de treinamento:", X_treino.shape[0])
print("Quantidade de amostras de teste:", X_teste.shape[0])
```

Treinamento e Avaliação do Modelo

Conforme o processo de Machine Learning, é necessário treinar um modelo de classificação para identificar as classes do problema.

No exemplo a foi utilizado o algoritmo Naïve Bayes na variante Gaussiana. Foi considerada a biblioteca Scikit-Learn e o método `fit()`, que aprende as amostras do conjunto de treinamento com os respectivos rótulos de classe. Em seguida, é possível medir o acerto do modelo na fase de treinamento através do método `score()`.

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB() # Método Naive Bayes Gaussiano
nb.fit(X_treino.values, y_treino) # Treinamento do Modelo
nb.score(X_treino.values, y_treino) # Avaliação do acerto do treinamento
```

O valor retornado indica que o modelo é capaz de encontrar corretamente a classe para 95% das amostras do conjunto de treinamento.

Após o treinamento, é possível utilizar o modelo para prever os valores de classe para as amostras de teste. O objetivo é verificar o quanto o modelo acerta considerando amostras até então desconhecidas para ele. Para isso, é utilizado o método `predict()`, que retorna os valores previstos para os objetos do conjunto de teste:

```
from sklearn.metrics import accuracy_score
y_previsto = nb.predict(X_teste.values) # Previsão das classes dos objetos do conjunto de teste
y_previsto # Resultado da previsão
```

A métrica acurácia do modelo considerando o conjunto de teste pode ser obtida pelo método `accuracy_score`. A pontuação obtida mostra que o modelo tem um acerto de 100% para o conjunto de teste.

```
accuracy_score(y_teste, y_previsto) # Valor de acurácia do modelo
```

A análise da qualidade do modelo pode ser feita por uma matriz de confusão. Ela é capaz de informar quantas amostras do conjunto foram corretamente associadas ao rótulo real:

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sn

y_previsto = nb.predict(X_teste)
cm = confusion_matrix(y_teste, y_previsto)

plt.figure(figsize = (8,6))

sn.heatmap(cm,
           annot=True,
           cmap= 'winter_r',
           xticklabels = ['Iris-setosa','Iris-versicolor','Iris-virginica'],
           yticklabels = ['Iris-setosa','Iris-versicolor','Iris-virginica'],
           cbar=False)
plt.xlabel('Classe prevista')
plt.ylabel('Classe real')
plt.title('Matriz de confusão - Naïve Bayes')
```

Através da matriz de confusão nota-se que o modelo classifica corretamente todas as amostras da espécie Iris-Setosa, mas incorretamente três amostras em cada uma das classes Iris-Versicolor e Iris-Virgínica.

Isso mostra que, apesar de obter uma boa acurácia, o modelo não consegue acertar totalmente. Para nosso exemplo isso não é exatamente um problema, mas em aplicações reais, errar a identificação de uma determinada classe pode não ser tolerado para a finalidade pretendida. Nesses casos, pode ser necessário reavaliar o modelo que foi empregado, afim de encontrar um que se ajuste melhor aos dados, ou ainda buscar por melhores representações do dado para o contexto.

Modelo - Decision Tree

Um modelo alternativo pode ser obtido através de uma Árvore de Decisão. Os procedimentos iniciais são semelhantes ao do Naive Bayes, de modo que as seguintes passos são executados:

1. Entrada de Dados
2. Preparação do Conjunto de Dados
3. Divisão do Conjunto de Dados

Treinamento e Avaliação do Modelo

Após a execução destes passos é realizado o Treinamento e Avaliação do Modelo. O comando a seguir faz o treinamento do modelo.

```
# Importe o classificador DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

# Crie uma instância do classificador com profundidade máxima de 3
clf = DecisionTreeClassifier(max_depth=3, random_state=42)

# Treine o modelo
clf.fit(X_treino, y_treino)
```

A visualização da árvore gerada pode ser obtida pelo código a seguir:

```
from sklearn import tree
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=['comp_petala','larg_petala'],
                                class_names=['Iris-setosa','Iris-versicolor','Iris-virginica'],filled=True,
                                 rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("Gini")
graph
```

A avaliação do modelo baseado na Árvore de Decisão pode ser feita pelo comando abaixo:

```
clf.score(X_teste, y_teste)
```

A avaliação do modelo por cross-validation (validação cruzada) pode ser feita conforme a seguir:

```
# Importe o método cross_val_score
from sklearn.model_selection import cross_val_score

# Instância o classificador Árvore de Decisão
clf = tree.DecisionTreeClassifier(max_depth=3, random_state=42)

# Obtenha as pontuações da validação cruzada
scores = cross_val_score(clf, X_treino, y_treino, cv=5)

# Exiba o valor médio e o desvio padrão das pontuações obtidas
print("%0.2f de acurácia com um desvio padrão de %0.2f" % (scores.mean(), scores.std()))
```

Avaliação por Matriz de Confusão:

```
from sklearn.metrics import confusion_matrix
import seaborn as sn
import matplotlib.pyplot as plt

# Instância o classificador Árvore de Decisão
clf = tree.DecisionTreeClassifier(max_depth=3, random_state=42)
clf.fit(X_treino, y_treino)

y_previsto = clf.predict(X_teste)
cm = confusion_matrix(y_teste, y_previsto)

plt.figure(figsize = (8,6))

sn.heatmap(cm,
           annot=True,
           cmap= 'winter_r',
           xticklabels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
           yticklabels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
           cbar=False)
plt.xlabel('Classe Prevista')
plt.ylabel('Classe Real')
plt.title('Matriz de confusão - Decision Tree')
```

Um relatório simples do resultado da classificação pode ser obtido conforme a seguir:

```
from sklearn.metrics import classification_report

print(classification_report(y_teste, y_previsto, target_names=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']))
```

REGRESSÃO

Primeiramente tratamos o arquivo com os dados: após a leitura e visualização de algumas informações, renomeamos as colunas de acordo com o comprimento e a largura da **sépala** e o comprimento e a largura da **pétala**.

```
df.head(5)
```

```
# Informações do arquivo lido
df.info()
```

```
# Renomeando as colunas
df.rename(columns = {0:'SepalLengthCm',
                     1:'SepalWidthCm',
                     2:'PetalLengthCm',
                     3:'PetalWidthCm',
                     4:'Species'}, inplace = True)
df
```

Podemos verificar a quantidade de tipos de Iris presentes nos dados. Essa classificação é baseada no comprimento/largura da sépala e no comprimento/largura da pétala.

```
# Verificando a quantidade de tipos: "setosa", "versicolor", "virginica"
df['Species'].value_counts()
```

O comando abaixo atribui um número inteiro para cada tipo de Iris: 0 para "setosa", 1 para "versicolor" e 2 para "virginica". As alterações são feitas na coluna 'Species'.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
df.head(150)
```

Vamos extrair da tabela apenas os tipos 0 "setosa" e 1 "versicolor" para construir o modelo de Regressão Logística.
Como temos 50 informações de cada tipo, vamos extrair 100 dados referentes ao comprimento/largura de sépala.

```
X = df[['SepalLengthCm', 'SepalWidthCm']].values[0:100]
Y = df['Species'][0:100]
```

Dos 100 dados extraídos (50 do tipo 0 e 50 do tipo 1), escolhemos 30 aleatoriamente para formar o conjunto teste.
Os 70 dados restantes são utilizados para construir o modelo.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30,random_state=13)
print("Quantidade de amostras de treinamento:", X_train.shape[0])
print("Quantidade de amostras de teste:", X_test.shape[0])
```

O modelo de Regressão Logística é construído e sua precisão é testada.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
```

```
print("Accuracy: ", model.score(X_test, Y_test) * 100)
```

Com os coeficientes do modelo de Regressão Logística obtido, podemos construir um gráfico em que visualizamos a fronteira de separação entre os dois tipos de Iris (setosa ou versicolor).

```
theta = np.ravel(model.coef_)
print(theta)
```

```
intercept = np.ravel(model.intercept_)
print(intercept)
```

```
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt

tipo0 = (np.ravel(Y) == 1).reshape(100,1)
tipo1 = (np.ravel(Y) == 0).reshape(100,1)

ax = sns.scatterplot(x = X[tipo0[:, 0], 0],
                     y = X[tipo0[:, 0], 1],
                     marker = "^",
                     color='green',
                     s=60)
sns.scatterplot(x = X[tipo1[:, 0], 0],
                y = X[tipo1[:, 0], 1],
                     marker = "X",
                     color='red',
                     s=60)
ax.legend(['Setosa', 'Versicolor'])
ax.set(xlabel="SepalLength", ylabel="SepalWidth")

x_boundary = np.array([np.min(X[:, 0]), np.max(X[:, 0])])
y_boundary = -(intercept + theta[0] * x_boundary) / theta[1]

sns.lineplot(x=x_boundary, y=y_boundary, color="blue")
plt.show();
```

Utilizando o modelo construído para classificar um tipo de Iris com comprimento e largura da sépala igual a 5.2cm e 2.6cm, respectivamente.

```
def logistic_function(x):
    return 1 / (1 + np.exp(-x))
```

```
teste = np.array([5.2, 2.6])
teste = np.append(np.ones(1), teste)
coef_otimo = np.append(intercept, theta)
probabilidade = logistic_function(teste.dot(coef_otimo))
print("Se o comprimento e a largura da sépala é 5.2 e 2.6, respectivamente, a probabilidade de ser do tipo setosa é",
      np.round(probabilidade, 2)*100, "%")
```

Referências

- McKINNEY, Wes. "Python para análise de dados: Tratamento de dados com Pandas, NumPy e IPython". Editora Novatec, 2018.
- KLOSTERMAN, Stephen. "Projetos de Ciência de Dados com Python: Abordagem de estudo de caso para a criação de projetos de ciência de dados bem sucedidos usando Python, Pandas e Scikit-Learn. Editora Novatec, 1ª Edição, 2020.
- GÉRON, Aurélien. "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow". O'Reilly Media, Inc., 2022.
- Biblioteca Pandas Python
- Biblioteca Pandas-Profile Python
- Biblioteca Sciki-Learn - Naive Bayes Gaussian
- Biblioteca Sciki-Learn - Decision Trees