Lista de exercícios - PCS3438

Gustavo Freitas de Sá Oliveira - 11261062
Roberta Boaventura Andrade - 11260832

05/12/2022

1 Introdução

Os programas desenvolvidos para a resolução dos problemas descritos foram desenvolvidos em Python. Foram utilizadas as bibliotecas: pandas (leitura e tratamento de dados), numpy (operações matemáticas) e sklearn (modelos de inteligência artificial e métodos de validação).

2 Questão 1

Considerando os dados presentes no arquivo class01.csv, treine o algoritmo Naive Bayes Gaussiano utilizando a metodologia de validação cruzada hold-out (utilize para treino as 350 primeiras linhas e para validação as demais). Qual o valor da acurácia a base de treino? Qual o valor da acurácia na base de validação?

```
# Validação hold-out
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
    train_size = 350, shuffle = False)

# Modelo Naive Bayes Gaussiano
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(x_train, y_train)

acuracia_treino = nb.score(x_train, y_train)
acuracia_validacao = nb.score(x_test, y_test)
```

O código apresentado acima lê a base de dados e inicializa um objeto *GaussianNB*, que recebe os dados como parâmetros nas funções *fit* e *score*.

Essas funções são responsáveis por treinar e avaliar o algoritmo, respectivamente.

O método de validação hold-out é implementado com a função *train_test_split*, que separa a base em duas porções para treino e validação.

Saída observada:

```
Acurácia na base de treino: 0.76
Acurácia na base de validação: 0.6276923076923077
```

Nesse exemplo, percebe-se que a acurácia na base de treino foi maior do na base de validação. Esse fenômeno pode estar relacionado com o *overfitting*, refletindo uma "memorização" dos dados pelo modelo.

3 Questão 2

Considerando os dados presentes no arquivo class02.csv, treine o algoritmo 10-Nearest Neighbors (KNN com $k = 10$ e distância Euclidiana), utilizando a metodologia de validação cruzada k-fold com 5 folds. Considere que a primeira pasta de validação seja formada pelas primeiras 20% linhas do arquivo, que a segunda pasta de validação seja formada pelas 20% linhas seguintes, e assim por diante, até atingir a última pasta, formada pelas 20% linhas finais da base. Qual a acurácia média para a base de validação?

```
# Validação K-fold
from sklearn.model_selection import KFold
kf = KFold(5)

# Modelo KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(10)

acuracia_treino, acuracia_validacao = 0, 0
for train, test in kf.split(x):
    x_train, x_test = x[train], x[test]
    y_train, y_test = y[train], y[test]

    knn.fit(x_train, y_train)

    acuracia_treino += knn.score(x_train, y_train)
    acuracia_validacao += knn.score(x_test, y_test)

acuracia_treino /= kf.get_n_splits(x)
acuracia_validacao /= kf.get_n_splits(x)
```

O código apresentado acima lê a base de dados e inicializa um objeto *KNeighborsClassifier*, que recebe os dados como parâmetros nas funções *fit*

e *score*. Essas funções são responsáveis por treinar e avaliar o algoritmo, respectivamente.

O método de validação k-fold é implementado com o objeto *Kfold*, com parâmetro 5. Esse método divide a base em partes iguais, utilizando uma de cada vez para validação, e as demais para treino.

Saída observada:

```
Acurácia na base de treino: 0.8661666666666668
Acurácia na base de validação: 0.8386666666666667
```

Assim como no exemplo anterior, percebe-se uma maior acurácia na base de treino. Isso pode estar também relacionado com o fenômeno anteriormente descrito.

4 Questão 3

Considerando os dados presentes no arquivo reg01.csv, obtenha um modelo de regressão linear com regularização L1 (LASSO com $\alpha = 1$) utilizando a metodologia Leave-One-out. Qual o valor médio do Root Mean Squared Error (RMSE) para a base de treino e para a base de validação?

```
# Validação Leave-One-Out
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()

# Modelo LASSO
from sklearn.linear_model import Lasso
l = Lasso(1)

rmse_treino, rmse_validacao = 0, 0
from sklearn.metrics import mean_squared_error
for train, test in loo.split(x):
    x_train, x_test = x[train], x[test]
    y_train, y_test = y[train], y[test]

    l.fit(x_train, y_train)

    rmse_treino += np.sqrt(mean_squared_error(y_train, l.predict
        (x_train)))
    rmse_validacao += np.sqrt(mean_squared_error(y_test, l.
        predict(x_test)))

rmse_treino /= loo.get_n_splits(x)
rmse_validacao /= loo.get_n_splits(x)
```

O código apresentado acima lê a base de dados e inicializa um objeto *Lasso*, que recebe os dados como parâmetros nas funções *fit* e *predict*. Essas funções são responsáveis por treinar e avaliar o algoritmo, respectivamente.

O método de validação Leave-One-Out é implementado com o objeto *LeaveOneOut*. Esse método é semelhante ao k-fold anteriormente exemplificado, mas aqui o número de folds é igual ao número de exemplos na base de dados.

A obtenção do RMSE é feita retirando a raiz quadrada do erro médio quadrado das predições do modelo.

Saída observada:

```
RMSE na base de treino: 19.220259837710355
RMSE na base de validação: 15.465218791702428
```

O *RMSE* foi maior na base de validação que na base de treino que na base de validação. A escolha do método de validação (Leave-One-Out) pode ter influenciado nesse fenômeno, uma vez que, nesse método, as validações acontecem para cada linha da base de dados.

5 Questão 4

Considerando os dados do arquivo reg02.csv, treine árvores de regressão, sem realizar podas, utilizando a metodologia de validação cruzada k-fold com $k = 5$. Qual o valor do Mean Absolute Error (MAE) para a base de treino? Qual o valor médio do MAE para a base de validação?

```
# Validação K-fold
from sklearn.model_selection import KFold
kf = KFold(5)

# Modelo árvore de regressão
from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor(criterion = 'absolute_error')

mae_treino, mae_validacao = 0, 0
from sklearn.metrics import mean_absolute_error
for train, test in kf.split(x):
    x_train, x_test = x[train], x[test]
    y_train, y_test = y[train], y[test]

    tree.fit(x_train, y_train)

    mae_treino += mean_absolute_error(y_train, tree.predict(
        x_train))
```

```
    mae_validacao += mean_absolute_error(y_test, tree.predict(
        x_test))

mae_treino /= kf.get_n_splits(x)
mae_validacao /= kf.get_n_splits(x)
```

O código apresentado acima lê a base de dados e inicializa um objeto *DecisionTreeRegressor*, que recebe os dados como parâmetros nas funções *fit* e *predict*. Essas funções são responsáveis por treinar e avaliar o algoritmo, respectivamente.

O método de validação k-fold é o mesmo mostrado anteriormente.

A obtenção do MAE é feita retirando o erro absoluto médio das predições do modelo.

Saída observada:

```
MAE na base de treino: 0.0
MAE na base de validação: 43.22051929803169
```

Nesse exemplo, o *MAE* foi nulo na base de treino. Esse resultado é esperado em virtude do tipo de modelo escolhido (árvore de regressão). O *MAE* na base de validação, entretanto, foi diferente de zero.